Quasi-Adiabatic Logic Arrays for Silicon and Beyond-Silicon Energy-Efficient ICs

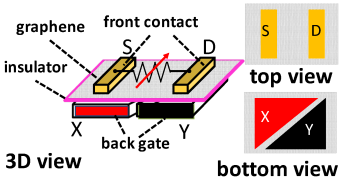
Quasi-Adiabatic Logic Arrays for Silicon and Beyond-Silicon Energy-Efficient ICs

Valerio Tenace, Andrea Calimera, *Member, IEEE,*
Enrico Macii, *Fellow, IEEE,* Massimo Poncino, *Senior Member, IEEE*

***Abstract*—This work describes a novel integration strategy that aims at bringing adiabatic computation to large scale integration. The proposed design solution, built upon logic primitives packed into regular arrays, the *Quasi-Adiabatic-Logic-Arrays* (QALAs), is well suited not just for today's silicon transistors but also for emerging devices. QALAs are indeed a viable path towards the integration of ultra-low power ICs using devices with a gapless energy spectrum, graphene p-n junctions in particular. The design of QALA circuits is supported by a dedicated RTL-to-device design framework whose kernel consists of a new area/delay-driven logic synthesis&optimization engine.**
Simulation results conducted on several benchmarks mapped both onto silicon (using 40nm MOSFETs) and graphene (using electrostatically controlled p-n junctions) show the proposed methodology provides adiabatic logic circuits with a power-delay-product (PDP) that is one order of magnitude smaller than that of state-of-the-art adiabatic circuits implemented using pass-transistor logic (PTL) and conventional logic synthesis flows.

I. Introduction

Adiabatic logic aims at mimicking an adiabatic (i.e., without energy exchange) charging process in digital circuits. Although regarded as a mostly theoretical computation style, research on the topic has been constantly active over the years, providing several demonstrations of working implementations [1], [2]. The interest in adiabatic circuits recently increased with the raising of emerging devices, such as Nanoelectromechanicals switches (NEMs) [3] and graphene p-n junctions [4], proven to be good technological vehicles for adiabatic computing. These devices naturally implement passive resistors through which a load capacitance can be charged (ideally) at zero-energy consumption. This feature is particularly suited for the next generation of nano-computers and nano-machines [5].
Despite the high energy efficiency, adiabatic logic faced severe limitations in reaching large-scale integration. Two are the main reasons: first, the difficulty in logic pipelining, primarily due to complex clocked-power schemes; second, the lack of computer-aided design tools able to cope with modern circuits complexity.
A practical solution to overcome these issues is presented in this paper. It consists of a new design and integration strategy based on regular arrays, the *Quasi-Adiabatic Logic Arrays* (QALAs), in which logic primitives are placed and connected in a way that enables Boolean logic functions of being evaluated adiabatically; complex digital ICs are assembled using multiple QALAs. The internal computation of a QALA is adiabatic while interface to/from primary inputs/outputs and/or other QALAs follows a conventional synchronous digital protocol. This allows large scale integration of *Globally-Static Locally-Adiabatic* circuits.
The key characteristic of a QALA lies in its internal topology, that is, a forest-of-trees rather than the standard tree structure adopted in classical adiabatic Pass-Transistor Logic (PTL) [2]. Such a forest topology allows deeper optimization during logic synthesis and hence a smarter use of emerging devices with high expressive power, graphene p-n junctions in particular.
Needless to say, CAD tools that can efficiently tackle the synthesis of PTL or, even worse, QALA circuits do not exist yet. Indeed, while the logic synthesis evolved by following the growth of semi-custom CMOS circuits, little efforts have been made for PTL. It is not a coincidence that most previous works focus on PTL circuits built for specific arithmetic functions [6], [7] or handcrafted basic Boolean logic gates [8], [9]. The only existing synthesis tools for PTL rely on Binary-Decision-Diagrams (BDDs) [10], which, as will shown later in the text, do not fit the characteristics of QALAs. Hence, as additional contribution, this work briefly describes a novel top-down synthesis flow built upon the logic optimizer firstly introduced in [11].
It is worth emphasizing that the aim of this work is not to demonstrate the efficiency of adiabatic computing, nor the superior characteristics of graphene; instead it gives a viable path and the related tools to pursuing ultra-low energy computation within silicon and beyond-silicon ICs.
Simulation results show the new QALAs architectures obtained with the proposed tool are smaller in size and lower in depth if compared to PTL adiabatic circuits generated using classical synthesis engines. This reflects into circuit implementations with a lower power-delay products.



S	X	Y	$R_{S\text{-}D}$	D
evaluation signal	0	0	Low	1
	0	1	High	0
	1	0	High	0
	1	1	Low	1

Fig. 1. Graphene p-n junction.

II. Background and Previous Works

A. Graphene p-n Junction

In the landscape of emerging devices, graphene p-n junctions stand out for their similarity to silicon transmission gates (TGs), yet with a higher expressive power. A graphene p-n junction (Figure 1) is a four-terminal device with two control pins (X and Y) implementing the electrostatic doping, one input pin (S) to which an input evaluation signal is fed, and one output pin (D) that collects the evaluation signal [12]. The control pins are back-gates isolated from the graphene sheet by a thin layer of oxide, while the input/output pins are front metal-to-graphene contacts. When the back gates are fed with control signals having the same "polarity", i.e., 0/0 or 1/1, the junction is in the ON-state (low R_{S-D}) and the input evaluation signal can propagate through the junction; control signals having opposite "polarity", i.e., 0/1 or 1/0, turn the junction OFF (high R_{S-D}) thereby preventing the propagation of the evaluation signal [12].
From a functional viewpoint, a p-n junction resembles a 2-input silicon MOS TG with an embedded EXNOR functionality (refer to the table in Figure 1); however its electrical behavior shows substantial differences. During the switching transients, a TG suffers a gradual ripple that follows the rise of the output voltage; this is the main source of voltage drop along a chain of TGs. Instead, a graphene p-n junction acts

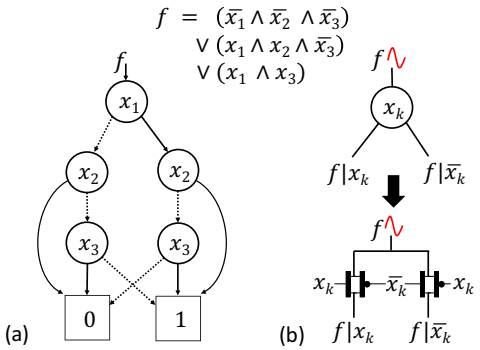


Fig. 2. A Binary Decision Diagram (a) and the MOSFET implementation of a generic node k (b).

as voltage-controlled passive resistor whose resistance stays constant during switching, regardless the S-to-D voltage [13]. This guarantees a smooth propagation of the input evaluation signal, with huge benefits on propagation delay and signal integrity. Such a characteristic makes graphene p-n junctions a good candidate for pass-logic [14], [11]. Several demonstrations of working graphene p-n junctions were described in previous works, e.g., [15], [16]. This work leverages the electrical Verilog-A model developed in [17], which describes the physical implementation of the device firstly presented in [12].

B. Tree-Based Pass-Transistor Logic (PTL)

The *pass-transistor logic* (PTL) [8] has been introduced as an answer to the growing complexity of CMOS circuits. As a matter of fact, PTL implements logic functions with a lower transistor count, smaller parasitic capacitance and hence better performance [10]. Even modern CMOS libraries make use of PTL for some logic gates, e.g., flip-flops and multiplexers. Also, PTL circuits can be used to implement logic adiabatic operations [2].

The few existing tools for the logic synthesis of PTL circuits rely on Binary Decision Diagrams (BDDs) [18], [10] (Figure 2-a). Each node in a BDD (Figure 2-b) represents the Shannon's expansion of a Boolean function f w.r.t. an input variable x_k. Outgoing edges represent the positive and negative co-factors, whereas the incoming edge represents an evaluation signal that propagates throughout the BDD network, from root to sink. The BDD nodes work as multiplexers whose circuit implementation consists of two TGs (Figure 2-b). Hence, the logic synthesis of PTL circuits encompasses the construction of a BDD and its 1-to-1 mapping onto a tree of multiplexers. Even though BDDs are a natural representation for PTL circuits, their use reflects into a "pyramidal" structure which, as will shown later in the text, does contrast with the regular architecture of QALAs. This imposes a departure from classical BDD-based logic synthesis. Moreover, algorithms used for static multi-level CMOS circuits provide circuit implementations that are not suited for adiabatic computation [19]. Finally, BDD decomposition can't exploit the EXNOR functionality made available by graphene p-n junctions. Only a very recent version of BDDs, the Biconditional-BDDs [20], fixes this issue. However, even BBDDs are trees that do not fit well with the structure of QALAs.

III. Quasi-Adiabatic Logic Arrays (QALA)

A. Internal Architecture

The QALA architecture, depicted in Figure 3, implements a multi-input/single-output combinational logic function. An evaluation signal, referred as the "clocked-power", adiabatically propagates through an array of resistive logic primitives, called f-switches, whose ON/OFF state is controlled by external logic primary inputs. Series connections of f-switches make the *logic paths*, while parallel logic paths form the *logic array*. Notice that logic paths are in mutual exclusion, namely, only one path is active under a certain input pattern; this avoids sneak paths. Mutual exclusion is managed at the synthesis stage, during logic decomposition. At the output of the logic array, a Schmitt-trigger digitizes the clocked-power signal, whereas a Flip-Flop samples and stores the squared QALA's output.

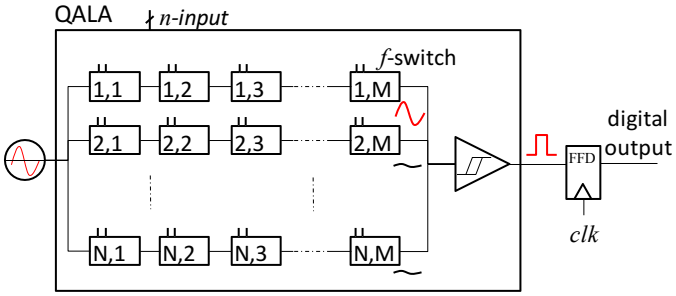


Fig. 3. NxM Quasi-Adiabatic Logic Array (QALA) architecture.

B. Modular QALAs and Timing Issues

Figure 4 shows the abstract view of a Globally-Static Locally-Adiabatic (GSLA) circuit, which consists of multiple QALAs working in parallel. The sampling at the output of each QALA is driven by a global clock signal clk. Such a clock is obtained from the sinusoidal waveform fed at the input of the QALAs delayed in time by t_d; multiple QALAs share the same sine wave. As for the QALA's output, the clk signal is squared by a Schmitt-trigger having the same threshold voltage V_{th}.

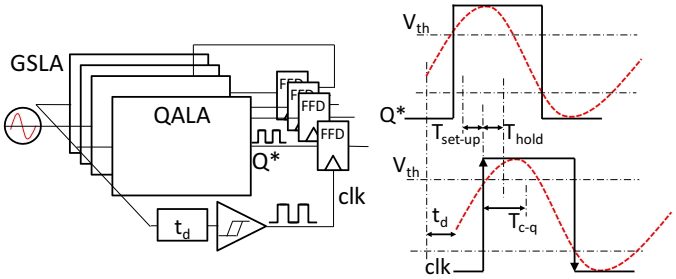
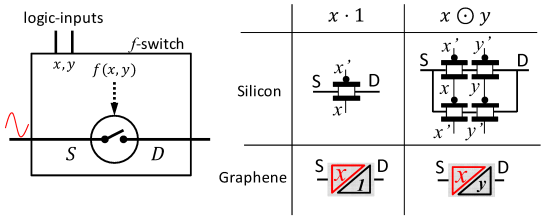


Fig. 4. Globally-Static Locally-Adiabatic circuits and timing issues.

Both the t_d and the V_{th} values are selected such that the set-up/hold constraints of the flip-flops are satisfied. Notice that for backward signals the set-up time is satisfied by construction ($T_{c-q} < T_{hold}$), while the primary inputs of each QALA are updated at the falling edge of the clk. As an alternative solution, the delay shift t_d can be applied at the output of the Schmitt-trigger, i.e., on the digital clock clk. Moreover, a tunable delay unit can be used to compensate skews in the clock signal (this issue is out of the scope of this work).

C. QALA Logic Primitives

The logic primitive of a QALA is the f-switch (Figure 5). Its abstract view consists of a switch controlled by a 2-input Boolean function $f(x, y)$. Different embodiments of such function are possible depending on the technology in use:



Material	Physical device	Logic Primitive	$f(x,y)$	# physical devices
Silicon	MOSFET	TG	$x \cdot 1$	2
			$x \odot y$	8
Graphene	p-n junction	XNOR PG	$x \cdot 1$	1
			$x \odot y$	1

Fig. 5. f-switch for QALAs: abstract view and physical implementation using Silicon MOSFETs and Graphene p-n junctions.

the *identity* function $f(x,y)$=$x \cdot 1$, and the *EXNOR* function $f(x,y)$=$x \odot y$.

As shown in Figure 5, the two primitives can be implemented using both silicon and graphene. With silicon, the *identity* function requires only two transistors, while the *EXNOR* needs eight transistors. With graphene, both functions can be realized using a single p-n junction. This suggests how different technologies have preferential primitives, i.e., those with higher expressive power (identity for silicon, EXNOR for graphene). A smart logic synthesizer must support multiple primitives, as the one proposed in this work does.

D. Design Flow for QALA Architectures

Figure 6 describes the design flow for QALA circuits. After a first step of standard multi-level logic synthesis, the generated netlist is partitioned into multiple single-output disjoint logic cones. Each cone is f-decomposed and then optimized by means of Boolean transformations for redundancy removal [11], [21]. Finally, depending on the technology in use (silicon or graphene) and the kind of function f used during decomposition (identity or EXNOR), a dedicated algorithm maps each logic cone into a QALA.

The synthesis&optimization engine used in the flow consists of a modified version of the *Gemini* tool introduced in [11]. Although a discussion of the algorithmic details is out of the scope of this work, it is important to underline that *Gemini* implements an *One-Pass Synthesis* (OPS) algorithm, where logic synthesis and technology mapping are performed simultaneously on the same data-structure. The latter, called *Pass-Diagram* (PD), substantially differs from BDDs, as it naturally represents forest topologies rather than trees. PDs do match with the QALA requirements, thereby enabling deeper optimization and faster convergence; this results into more efficient circuit implementations which count of less devices, have faster paths, and guarantee more energy efficiency. Indeed, by leveraging an appropriate forest-based abstraction model, the *Gemini* tool generates a compact representation of the Boolean function.

E. Power Modeling

The dynamic power consumption of a QALA is the sum of two main contributions: (i) P_{in}, which is the power consumed by primary inputs when charging/discharging the gate capacitance of the f-switches; (ii) P_{eval}, which is the power consumed due to the propagation of the evaluation signal through the f-switches. The former term (P_{in}) is similar to the input power consumed by CMOS gates; the latter (P_{eval}) follows the adiabatic charging principle.

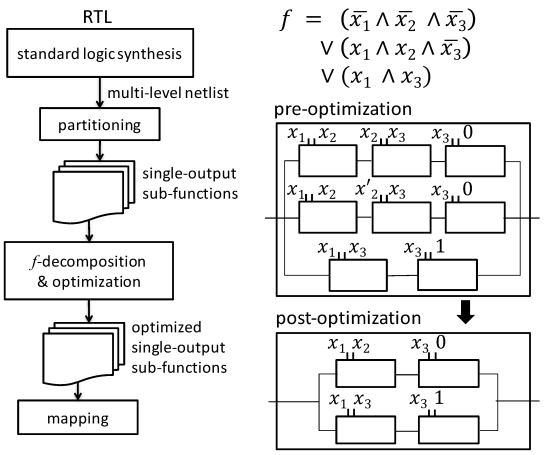


Fig. 6. Implemented top-down design framework and abstract view of a QALAs pre- and post-optimization (f-decomposition $x \odot y$).

From an electrical viewpoint, an array of f-switch primitives reduces to an equivalent resistor R_{eq} [17]. The R_{eq} is simply calculated as series/parallel connections of the in-to-out resistance of the f-switches; each of them can assume the value R_{ON}, i.e., the in-to-out resistance when the the f-switch is in the ON state, or R_{OFF}, i.e., the in-to-out resistance when the the f-switch is in the OFF state, depending on the actual input-pattern. It is worth noticing that the evaluation signal is allowed to propagate to the output *iff* there exists at least one path having all its f-switches turned ON.

For the sake of simplicity, and without loss of generality, let us approximate the rising phase of the sinusoidal evaluation signal as a simple ramp having a rise/fall-time T_{rf}.

Under such condition, the power consumed across R_{eq} can be calculated as $P_{eval}(t) = R_{eq} i^2_{C_l}(t)$, with i_{C_l} the current injected into C_l, the load capacitance attached at the output of the array. The average value is given by:

$$P_{eval} = \frac{1}{T_{rf}} \int_0^{T_{rf}} R_{eq} i^2_{C_l}(t) dt = \frac{R_{eq}}{T^2_{rf}} C^2_l V^2_{dd} \qquad (1)$$

As one can observe, the larger the T_{rf}, the smaller the P_{eval}. In particular, T_{rf}=$2RC$ is the break-even point at which the power consumed equals that obtained using a step evaluation signal. Ideally (i.e., $T_{rf} \rightarrow \infty$) the evaluation phase completes at zero-power, namely, adiabatically. Notice that SPICE-level simulations can be used to estimate R_{eq} and $i_{C_l}(t)$ under different input patterns.

IV. Simulation Results

A. Taxonomy of Existing Adiabatic Synthesis Flows

Figure 7 pictorially describes the taxonomy of the adiabatic logic implementations discussed in this work, that is, those for which logic synthesis tools are currently available. The criteria used for the classification are: (i) the circuit topology, i.e., forest (the one proposed in this work with QALAs) and tree (the Pass-Transistor Logic); (ii) the function implemented by the f-switches, i.e., identity ($f(x,y) = x \cdot 1$) and EXNOR ($f(x,y) = x \odot y$); (iii) the material used to implement the logic primitives, i.e., silicon (Si) and graphene (G).

The leaves, numbered from ① to ⑧, correspond to different circuit implementations, each obtained using a specific logic synthesis tools: *Gemini* for forest topologies, *CUDD* [22] and *BBDD* [20] for tree-based topologies depending on the logic primitive. For the sake of clarity, Figure 8 shows the abstract

	QALA (Forest)					Tree				
	① Transistors	② P-N	③ Transistors	④ P-N	Depth	⑤ Transistors	⑥ P-N	⑦ Transistors	⑧ P-N	Depth
apex1	3478	1739	11960	1495	15	113344	56672	257360	32170	45
bigkey	69770	34885	279080	34885	8	24680	12340	3546080	443260	486
c8	508	254	2032	254	10	544	272	9984	1248	28
duke2	1518	759	5864	733	14	3892	1946	20496	2562	22
k2	14228	7114	53392	6674	15	113344	56672	101648	12706	15
misex1	244	122	936	117	5	164	82	1136	142	8
misex2	376	188	1472	184	12	544	272	5696	712	25
s13207.1	206752	103376	788688	98586	17	2706704	1353352	23210720	2901340	700
sao2	846	423	3072	384	10	620	310	1472	184	10
5mod5	70	35	136	17	5	52	26	128	16	5
frg1	1584	792	6336	792	14	67268	33634	6544	818	28
hamming	1776	888	4160	520	6	420	210	1040	130	7
s510	970	485	3616	452	8	76304	38152	10176	1272	25
s1488	3268	1634	12296	1537	10	4064	2032	12816	1602	14
Total	**305388**	**152694**	**1173040**	**146630**	**-**	**3111944**	**1555972**	**27185296**	**3398162**	**-**

TABLE I
SYNTHESIS RESULTS FOR QALA AND TREE-BASED CIRCUITS. COLUMNS **TRANSISTORS** AND **P-N** REPORT THE TOTAL NUMBER OF DEVICES FOR SILICON AND GRAPHENE CIRCUITS RESPECTIVELY; COLUMN **DEPTH** REPORTS THE MAXIMUM DEPTH OF THE OBTAINED LOGIC STRUCTURE.

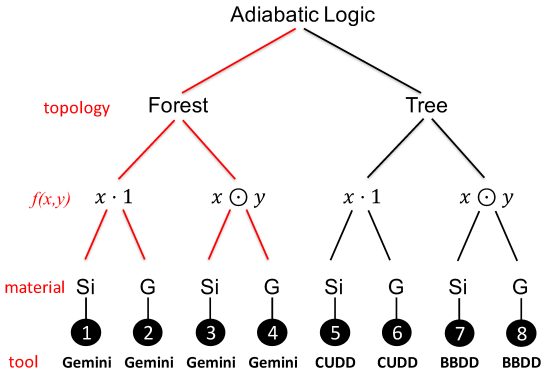
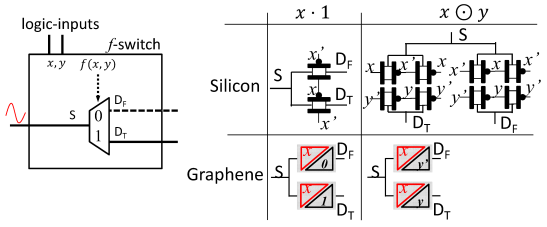


Fig. 7. Taxonomy of adiabatic logic styles.



Material	Physical device	Logic Primitive	$f(x,y)$	# physical devices
Silicon	MOSFET	TG	$x \cdot 1$	4
			$x \odot y$	16
Graphene	p-n junction	XNOR PG	$x \cdot 1$	2
			$x \odot y$	2

Fig. 8. mux-switch for decision diagrams: abstract view and physical implementation using Silicon MOSFETs and Graphene p-n junctions

view and the physical implementation of the logic primitives (mux-switch) adopted in tree-based architectures.

B. Experimental Setup

The final goal of this section is to give a fair comparison between forest-based QALA architectures and state-of-the-art tree-based adiabatic circuits, and demonstrate that QALAs obtained with the proposed synthesis flow show lower PDP than PTL circuits synthesized with standard BDD packages [22], [20]. Experiments were conducted on a set of open source benchmarks [23], and figures of merit were quantified by means of accurate SPICE simulations. Such simulations make use of card models provided with a commercial 40nm bulk technology for silicon, while for graphene we resort to the Verilog-A model described in [17]. Frequency and slew-rate of the inputs are compliant with the timing issues pictorially described in Section III.

C. Logic Synthesis Outcome

Table I collects the synthesis results. Following the taxonomy of Figure 7, the two master columns, i.e., **QALA** and **Tree**, report, for each benchmark, the total number of devices (either MOSFET transistors or graphene p-n junctions) and the depth of the resulting circuit (i.e., the number of devices on the longest path). As reported in Table I, the QALA circuits have (i) a lower logic depth and (ii) a lower number of devices (both for Silicon and Graphene). In particular, QALAs show a logic depth 9.5X shallower than that of tree networks, resulting into almost 10X faster paths. These huge savings are mainly due to: (i) a forest-like topology that grows in parallel rather than vertically as it happens for tree-based PTL circuits; (ii) a more efficient decomposition algorithm that can play with different logic primitives depending on the technology in use (*identity* for Silicon and *EXNOR* for Graphene). Such a result underlines the orthogonality of the QALA style and the optimization engine we propose. In other words, the QALA does not work better than PTL cause it leverages a specific characteristic of Graphene. On the contrary, its internal structure and the synthesis engine have intrinsic features that make them outperforming state-of-the-art PTL, both for Silicon and Graphene.

D. Power-Delay Analysis

Figure 9 and Figure 10 compare different implementations, from ① to ⑧, in terms of power-delay-product (PDP) vs. transition time of the input/evaluation signal (T_{rf}). Notice that PDP is averaged over the benchmarks reported in Table I. Figure 9 collects the results for silicon implementations (①, ③, ⑤, ⑦), while Figure 10 for graphene (②, ④, ⑥, ⑧). QALAs outperform PTL implementations at any T_{rf} and for both Silicon (e.g., ① and ③ vs. ⑤ and ⑦ in Figure 9) and Graphene (② and ④ vs. ⑥ and ⑧ in Figure 10). PDP savings achieved with QALAs are more than 1 order of magnitude (e.g., ① vs. ⑤ and ③ vs. ⑦). Concerning the logic primitive adopted, when considering silicon, the use of $f = x \cdot 1$ allows a 50% lower PDP w.r.t. $f = x \odot 1$ (① vs. ③ in Figure 9), while using $f = x \odot y$ works better for Graphene,

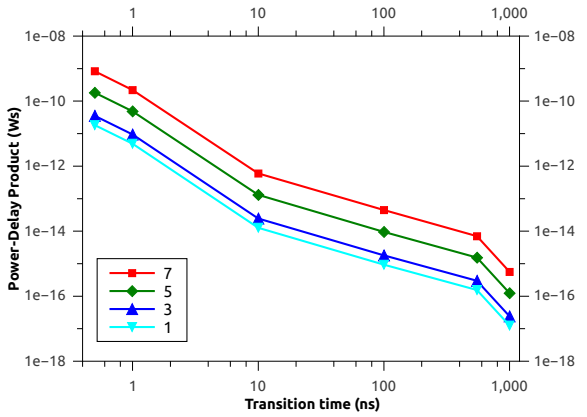


Fig. 9. Silicon adiabatic logic circuits: QALAs (①,③) vs. tree-based (⑤,⑦) implementations. QALAs outperform tree structures for any T_r. The $f = x \cdot 1$ decomposition ① shows slower PDP than $f = x \odot 1$ decomposition ③.

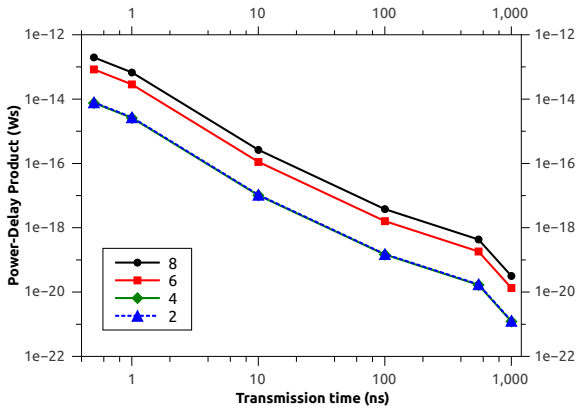


Fig. 10. Graphene adiabatic logic circuits: QALAs, (②,④) vs. tree-based (⑥, ⑧) implementations. Graphene QALAs outperform tree structures for any T_{rf}. The $f = x \cdot 1$ decomposition ② shows lower PDP than $f = x \odot 1$ decomposition ④.

7% lower PDP w.r.t. $f = x \cdot 1$ (② vs. ④ in Figure 10). Finally, a technology comparison is given in Figure 11 which shows a comparison between Silicon QALAs and Graphene QALAs (① vs. ②) obtained on top of the same *identity* primitive. Since graphene QALAs are more compact (2X less devices on the total count) and faster (roughly 2 orders of magnitude), their PDP is orders of magnitude smaller than silicon counterparts. This result highlights once again how the proposed design and integration strategy maximizes the potentials of emerging graphene devices. As an additional piece of information to support the proposed adiabatic logic, we conducted a quantitative comparison between QALAs (① and ②) and standard silicon CMOS circuits mapped onto a 40nm technology using a commercial logic synthesis tool. Experimental results reveal that there exists a break-even operating frequency below which QALAs show lower energy consumption than the CMOS circuits, i.e., 80MHz (for Silicon ①) and 770MHz (for Graphene ②). Applications that run at a speed slower than the break-even frequencies, might take genuine advantage from the QALA style.

V. Conclusions

Enabling ultra-low power computation represents the key for the development of future ceaseless-connected wireless devices in nano-computers. This works introduced a novel *Quasi-Adiabatic Logic Array* (QALA) architecture that effectively leverages the adiabatic charging principle for energy efficient computation. The proposed solution guarantees substantial improvement over existing adiabatic circuits techniques, in terms of area occupation (8X less devices) and power/delay (1 order of magnitude lower PDP).

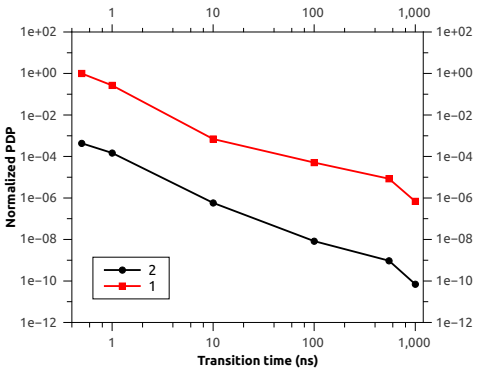


Fig. 11. Silicon QALAs ① vs Graphene QALAs ②

References

[1] D. J. Frank, "Comparison of high speed voltage-scaled conventional and adiabatic circuits," in *Low Power Electronics and Design, 1996., International Symposium on*. IEEE, 1996, pp. 377–380.
[2] V. G. Oklobdžija *et al.*, "Pass-transistor adiabatic logic using single power-clock supply," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 44, no. 10, pp. 842–846, 1997.
[3] S. Houri *et al.*, "Limits of cmos technology and interest of nems relays for adiabatic logic applications," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 62, no. 6, pp. 1546–1554, 2015.
[4] S. Miryala *et al.*, "Ultra low-power computation via graphene-based adiabatic logic gates," in *Digital System Design (DSD), 2014 17th Euromicro Conference on*. IEEE, 2014, pp. 365–371.
[5] I. F. Akyildiz and J. M. Jornet, "The internet of nano-things," *Wireless Communications, IEEE*, vol. 17, no. 6, pp. 58–63, 2010.
[6] M. Suzuki *et al.*, "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.
[7] J. D. Lee *et al.*, "Application of dynamic pass-transistor logic to an 8-bit multiplier," *Journal-Korean Physical Society*, vol. 38, no. 3, pp. 220–223, 2001.
[8] 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, Jul 1997.
[9] T.-Y. Wu *et al.*, "Low-leakage and low-power implementation of high-speed 65nm logic gates," in *EDSSC'08*, Dec 2008, pp. 1–4.
[10] R. S. Shelar and S. S. Sapatnekar, "Bdd decomposition for delay oriented pass transistor logic synthesis," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 13, no. 8, pp. 957–970, 2005.
[11] V. Tenace *et al.*, "One-pass logic synthesis for graphene-based Pass-XNOR logic circuits," in *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*. IEEE, 2015, pp. 1–6.
[12] S. Tanachutiwat *et al.*, "Reconfigurable multi-function logic based on graphene pn junctions," in *Design Automation Conference (DAC), 2010 47th ACM/IEEE*. IEEE, 2010, pp. 883–888.
[13] S. Miryala *et al.*, "Ultra-low power circuits using graphene p–n junctions and adiabatic computing," *Microprocessors and Microsystems*, vol. 39, no. 8, pp. 962–972, 2015.
[14] V. Tenace *et al.*, "Pass-XNOR logic: a new logic style for PN junction based graphene circuits," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2014, pp. 1–4.
[15] H.-Y. Chiu *et al.*, "Controllable pn junction formation in monolayer graphene using electrostatic substrate engineering," *Nano letters*, vol. 10, no. 11, pp. 4634–4639, 2010.
[16] H.-C. Cheng *et al.*, "High-quality graphene p- n junctions via resist-free fabrication and solution-based noncovalent functionalization," *Acs Nano*, vol. 5, no. 3, pp. 2051–2059, 2011.
[17] S. Miryala *et al.*, "A Verilog-A model for reconfigurable logic gates based on graphene pn-junctions," in *DATE Conference*, 2013, pp. 877–880.
[18] V. Bertacco *et al.*, "Decision diagrams and pass transistor logic synthesis," in *Int'l Workshop on Logic Synth*, vol. 168, 1997.
[19] I. Hänninen *et al.*, "Adiabatic CMOS: Limits of Reversible Energy Recovery and First Steps for Design Automation," in *Transactions on Computational Science XXIV*. Springer, 2014, pp. 1–20.
[20] "Biconditional Binary Decision Diagrams Software Package." Available at http://lsi.epfl.ch/BBDD, 2016.
[21] V. Tenace *et al.*, "Graphene-PLA (GPLA): a Compact and Ultra-Low Power Logic Array Architecture," in *Proceedings of the 26th edition on Great Lakes Symposium on VLSI*. ACM, 2016, pp. 145–150.
[22] F. Somenzi, "Cudd: Cu decision diagram package release 2.3. 0," *University of Colorado at Boulder*, 1998.
[23] "Collection of digital design benchmarks," http://goo.gl/6fOVfN, 2016.