Adapting Computer Arithmetic Structures to Sustainable Supercomputing in Low-Power, Majority-Logic Nanotechnologies

Ghassem Jaberipur, Behrooz Parhami, *Life Fellow, IEEE*, and Dariush Abedi

Abstract—Petascale supercomputers are already pushing power boundaries that can be supplied or dissipated cost-effectively; greater challenges await us in the era of exascale machines. We are thus motivated to study methods of reducing the energy cost of arithmetic operations, which can be substantial in numerically intensive applications. Additionally, being both a widely-used operation in itself and an important building block for synthesizing other arithmetic operations, has received much attention in this regard. Circuit and energy costs of fast adders are dominated by their fast carry networks. The availability of simple and energy-efficient majority function in certain emerging nanotechnologies (such as quantum-dot cellular automata, single-electron tunneling, tunneling phase logic, magnetic tunnel junction, nanoscale bar magnets, and memristors) has motivated our work to reformulate the carry recurrence in terms of fully-utilized majority elements, with all three inputs usefully employed. We compare our novel designs and resulting circuits to prior proposals based on 3-input majority elements in quantum-dot cellular automata, demonstrating advantages in both speed and circuit complexity. We also show that the performance and cost advantages carry over to at least one other emerging, energy-efficient technology, single-electron tunneling, raising hopes for achieving similar benefits with other technologies, which we review very briefly.

Index Terms—High-speed arithmetic, performance analysis and design aids, cellular arrays and automata, algorithms implemented in hardware, logic design styles, low-power design

1 Introduction

One of the key challenges of exascale computing is reigning in energy requirements of millions of nodes, both within the nodes and in internode communication. A second major challenge is dealing with the very real possibility that some components in the extremely complex system will malfunction, with the effects spreading and infecting the entire system. In this paper, we focus on the intranode energy consumption implied by high-speed computer arithmetic.

1.1 Sustainability and Energy Efficiency

According to generalized Amdahl's law, improvements in performance, energy consumption, or any other architectural attribute of a computer system will be limited if not applied across the board to all important subsystems and/or functions. Therefore, even though the contribution of arithmetic circuitry to energy consumption in modern supercomputers is estimated to be only 10-15 percent, given the rates at which energy efficiency of various subsystems, such as memory and communication (both inter-processor and inter-system) are improving, it won't be long before the arithmetic units are the main consumers of energy, if nothing is done to improve those as well.

It is well-known that faster circuits tend to consume more energy and that it could be advantageous to use a larger number of lower-performance circuits, operating in parallel, to reduce energy requirements. This is an attractive method when the application has a great deal of parallelism that can be exploited without undue overhead in scheduling, load-balancing, and communication. Reducing the energy requirements at the circuit or technology level is attractive, whether or not we use the aforementioned massive parallelization strategy.

Another approach to reducing energy requirements is to sacrifice precision, a technique that is the focus of the rapidly expanding field of approximate computing [1]. However, this trade-off is possible only when the computations performed are error-resilient (e.g., when convergence occurs, regardless of the error, if the latter is within a reasonable range). We explore the alternative of moving to completely new implementation technologies that are energy-efficient by nature.

Thus far, energy-efficiency has been the main attribute of concern when discussing sustainability. The parts and material from which computing devices are built and their proper handling at the end of their useful life is another

- *G. Jaberipur is with the Departement of Computer Sceince and Engineering, Shahid Beheshti University, Tehran 19839-63113, Iran, and the School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran. E-mail: jaberipur@sbu.ac.ir.*
- *B. Parhami is with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106. E-mail: parhami@ece.ucsb.edu.*
- *D. Abedi is with the Departement of Computer Sceince and Engineering, Shahid Beheshti University, Tehran 19839-63113, Iran. E-mail: d_abedi@sbu.ac.ir.*

Manuscript received 20 Apr. 2017; revised 5 Feb. 2018; accepted 19 Feb. 2018. Date of publication 1 Mar. 2018; date of current version 6 Dec. 2018. (Corresponding author: Ghassem Jaberipur.) Recommended for acceptance by A. Louri. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TSUSC.2018.2811181

aspect that has received less attention. Interestingly, the type of atomic-level computation envisaged by the new technologies discussed in this paper, and their biological brethren, are expected to contribute to this aspect of sustainability, although here we focus on their energy efficiency.

1.2 The Quest for Faster Addition

Speeding up the addition operation and the attendant carry computation was among early challenges faced by designers of electronic computers [2]. Mechanisms for carry anticipation were entertained even in the era of mechanical computing. For example, Charles Babbage fiddled with the idea of reducing the propagation penalty of ripple-carry addition [3]. The notion of carry-lookahead addition, whose modern inception dates back to five decades ago [4], is still being refined, both theoretically and practically. Theoretical refinements consist of new parallel-prefix formulations. Practical improvements consist of extension and fine-tuning for use with emerging technologies, as well as to accommodate newer optimization criteria, of which VLSI layout area and energy efficiency are most notable.

Two-way and multi-way combining of carry-generation and carry-propagation signals for blocks of input operands give rise to numerous circuit designs and implementation alternatives for carry generation networks (CGN). Two-way combining leads to the least complex carry operator block, but uses both more of such blocks and a larger number of levels in the carry network's critical delay path. According to an interesting taxonomy for parallel-prefix CGNs [5], alternative designs for a k-bit adder entail choices of values for circuit and layout parameters f, t, and l, where gate fanout is $2^f + 1$, number of wire tracks is 2^t, and number of circuit-block levels is $\lceil \log k \rceil + l$. Designs within this taxonomy, which use AND and OR gates in implementing the carry operator, offer tradeoffs in area, speed, and power.

1.3 Adaptation to Technological Changes

While the theory of fast addition is well-established, changes in technology necessitate a reassessment of strategies for carry network implementation from time to time, even though the logical functions to be realized remain the same.

Use of multiplexers and other types of building block [6] further expand the available options and make it possible to take advantage of the capabilities and efficiencies offered by new implementation technologies. In effect, each new technology brings with it more/less efficient realizations of certain building blocks, thus shifting the optimal design point. Whereas naive per-gate mapping of an existing design to a new technology may lead to improvements, it seldom results in an optimal design.

One promising new technology, with broad computational potential as well as non-computational applications, is quantum-dot cellular automata (QCA) which necessitates a fundamental reassessment of how we perform arithmetic [7], [8]. Other technologies with properties and potential similar to QCA also exist, with notable examples being single-electron integrated circuits [9], computational use of nanomagnets [10], molecular computing [11], [12], and memristors [13]. QCA has been used in the design (manually or via design tools) of full-adder blocks and simple adders [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], but there is no indication whether the resulting designs are optimal or just feasible ones. In parallel, majority-gate-based design methodologies have been studied for developing QCA circuits with up to 3 input variables (e.g., [21], [22]).

The aforementioned technologies, and a few others reviewed briefly in Section 2, allow efficient realization of majority gates, so a question of interest, addressed in this paper, is whether it is possible to formulate the carry computation directly in terms of majority logic, rather than trivially translate existing designs by letting partially utilized majority elements perform AND as $\mathcal{M}(0, x, y)$ and OR as $\mathcal{M}(1, x, y)$, where $\mathcal{M}$ denotes the 3-way majority function.

Majority elements, as well as accompanying techniques for synthesizing logic functions using such gates, have a long history. An n-input majority gate is a special case of a threshold circuit, with unit-weight inputs and threshold value of $\lceil \frac{n+1}{2} \rceil$. During its 70-year history, threshold logic has been revisited from time to time in connection with emerging technologies [23]. Early interest revolved around neural networks and neuronlike computational elements [24]. Subsequent designs were realized in several technologies and entailed capacitance- and inductance-based solutions, among others [25].

The rest of this paper is organized as follows. A brief survey of new $\mathcal{M}$-based (majority-friendly) technologies is provided in Section 2. The previous relevant works are discussed in Sections 3 (Naïve mapping) and 4 (QCA fast adders). Section 5 elaborates on details of the new carry generation scheme, whose actual utilization in designing majority based parallel prefix adders is offered in Section 6. QCA realization of the proposed adders, their evaluations and comparison with the relevant previous works are found in Sections 7 and 8, respectively. Section 9 discusses the extension of proposed scheme to other technologies. Closing remarks appear in Section 10.

2 New (Majority-Friendly) Technologies for Atomic-Scale Computation

The end of Moore's-Law scaling of integrated circuits will likely occur in the early 2020s [26] at a feature size of a few nanometers, which is only a decimal order-of-magnitude larger than the size of a silicon atom. Beyond this point, we enter the realm of atomic-scale computation, where physical effects such as quantum tunneling [27] should be accommodated or even exploited to achieve fast, reproducible, and reliable computation.

As noted in Section 1, majority function with equally weighted inputs is a special case of threshold logic, which allows arbitrary input weights and threshold value. CMOS realization of a majority gate with three Boolean inputs a, b, and c yields $\mathcal{M}(a, b, c) = (a + b)\, c + ab$. Clearly, $\mathcal{M}$ elements can be realized in other technologies via direct replacement of the AND and OR pairs in the expression above with their equivalents in the target technology. However, $\mathcal{M}$ is more attractive in some new technologies, where it can be realized far more efficiently.

The 3-input majority function can also be defined by the arithmetic expression $\mathcal{M}(a, b, c) = \lfloor \frac{a+b+c+1}{3} \rfloor$, and it can be viewed as the median function. The median

Fig. 1. Three QCA cell configurations.

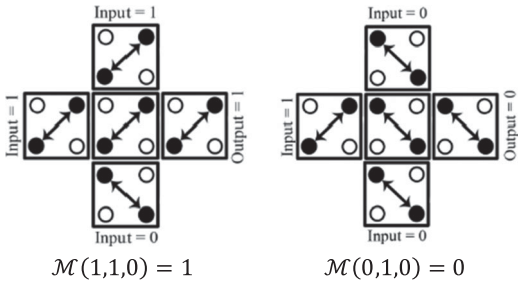


Fig. 2. QCA $\mathcal{M}$ gate with two input sets.

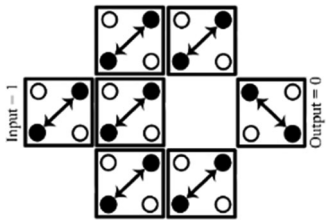


Fig. 3. A robust QCA Inverter.

interpretation of the majority function allows us to use the axiomatically defined median algebra [28] to prove new results and derive various relationships.

2.1 Quantum-dot Cellular Automata (QCA)

The basic QCA cell contains four electron place-holders or "dots," within which two injected electrons can assume the slash or the backslash configuration (Fig. 1, from [19]). QCA realization of the $\mathcal{M}$ gate is depicted in Fig. 2, and that of an inverter in Fig. 3. In fact, these two gates constitute a complete logic set, since both AND and OR functions can be expressed in terms of majority gate. For example, direct QCA realization of a 7-gate full adder (FA) entails the use of 7 partially utilized $\mathcal{M}$ gates, while, based on what we will present later, the same functionality can be realized via 3 fully utilized $\mathcal{M}$ gates and 2 inverters [29].

To capture the benefits of QCA realization of basic arithmetic circuits over their CMOS counterparts, figures of merit of the common CMOS FAs and the corresponding QCA realization are compiled in Table 1, where the contents have been obtained as explained below.

Regarding the delay evaluation, since QCA simulation engines do not provide for the working frequency, we estimated the frequency of a QCA FA, as realized in [19], as follows. The clock rate of QCA circuits, in general, is known to be around 1-2 THz [30], provided that number of cells in one zone is at most 18. On the other hand, operation of the FA of [19] takes only one clock cycle, where number of QCA cells in one clock zone is at most 16. Therefore, its frequency is estimated to be 1THz, in the worst case, whose latency (i.e., 1 ps) is far more than that of the typical 45 nm CMOS realization (i.e., 50000 ps) [31].

As for the area consumption, that of the QCA FA is 0.046 μm^2, while 94 μm^2 (i.e., over X2000) is reported [31] for the aforementioned CMOS technology.

TABLE 1
Figures of Merits of an FA Realized in CMOS and QCA

FA	Area (μm^2)	Delay (ps)	Power (pW)
QCA 18 nm	0.046	1	0.4
CMOS 45 nm [31]	94	50000	113

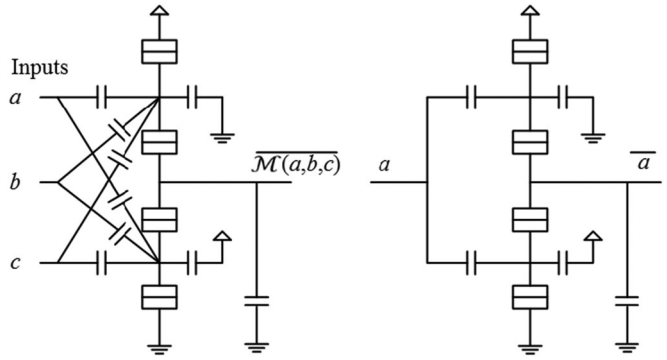


Fig. 4. SET circuits for Min (left) and inverter (right) [9].

Finally, power dissipation of the QCA FA of [19] that is evaluated by the QCAPro (Temperature $= 2$k, kink tunneling energy levels $= 0.5$k) [32] is less than 0.01 percent of the corresponding CMOS measure.

2.2 Single-Electron Tunneling (SET)

The ultimates in compactness and energy efficiency are offered by single-carrier electronics, which allows for controlled transfer of individual electrons, using the single-electron tunneling phenomenon. Using this technology for computation requires a demonstration of feasible logic gates, which has been done successfully for minority elements [9]. Fig. 4 shows the minority circuit along with an inverter that is needed to make the set universal.

2.3 Other Technologies

Tunneling phase logic (TPL): In TPL, several capacitively coupled inputs feed a load capacitance, which, under the right conditions, can realize the 3-input minority function [33], [34].

Magnetic tunnel junction (MTJ): One of the new spintronics technologies, MTJ is based on devices with two ferromagnetic thin-film layers, free (capable of changing magnetization) and fixed (not easily changeable). The resulting low or high resistance of the junction allows the representation of a bit [35], [36]. Realization of Majority gate in MTJ logic is reported in [35].

Nano-scale bar magnets (NBM): The use of NBMs as computational elements [10] was pursued as a way of overcoming some difficulties with QCA. Key benefits of computing with nanomagnets are their extreme energy efficiency and latch-free pipelining, due to the built-in non-volatile storage capability. However, this approach is no match for silicon-based technologies in terms of computation speed.

Memristors: Memristors [13] and memristive devices are basically resistors with varying resistance, which depends on the history of the device. The three schemes currently available for memristor-based majority logic realization are programmable majority logic arrays (PMLAs) [37], charge-sharing threshold gates (CSTGs) [38], and current-mirror threshold gates (CMTGs) [38].

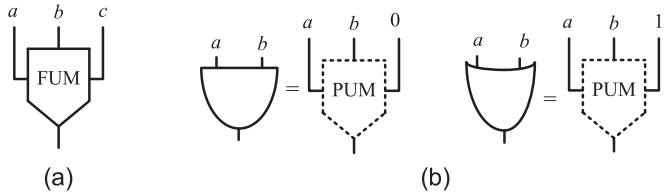


Fig. 5. (a) An FUM and (b) two PUMs.

DNA: Biological embodiments of the majority function [12] form a basis for neural computation in human and animal brains.

It appears that majority (or 2-out-of-3 agreement), extending both OR (1-out-of-2) and AND (2-out-of-2) functions of standard gates, is a capability that arises rather naturally in many different domains, so we can expect additional new technologies (biologically-based or otherwise) to support its efficient realization.

3 Naïve Mapping of CGNs to Majority Logic

Given that $a \wedge b = \mathcal{M}(a,\ b,\ 0)$ and $a \vee b = \mathcal{M}(a, b, 1)$, equivalents of the basic AND and OR gates can be realized via partially-utilized majority (PUM) gates, where one of the inputs is either 0 (for AND) or 1 (for OR), as depicted by Fig. 5(a). However, a fully-utilized majority (FUM) gate refers to an $\mathcal{M}$ gate with no constant input, as is shown by Fig. 5(b). For example, the 7-gate realization of a full adder (FA) of Fig. 6(a) can be realized with 7 PUM gates, as shown in Fig. 6(b). This naïve design has been significantly improved by Wang et al. [29] to that of Fig. 6(c), which uses only three FUM gates, and no PUMs.

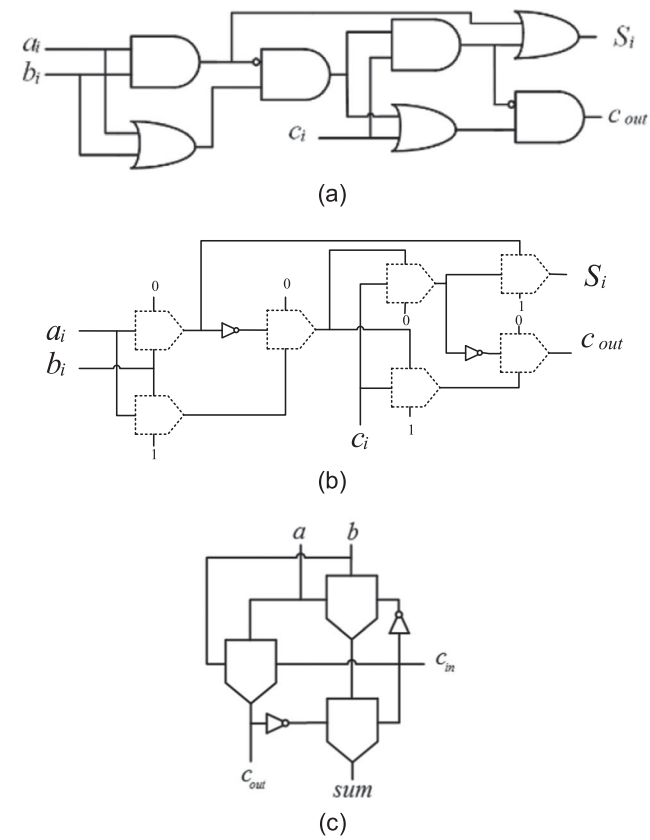


Fig. 6. (a) Seven-gate full adder, (b) its naïve equivalent PUM circuit, and (c) three-FUM full adder.

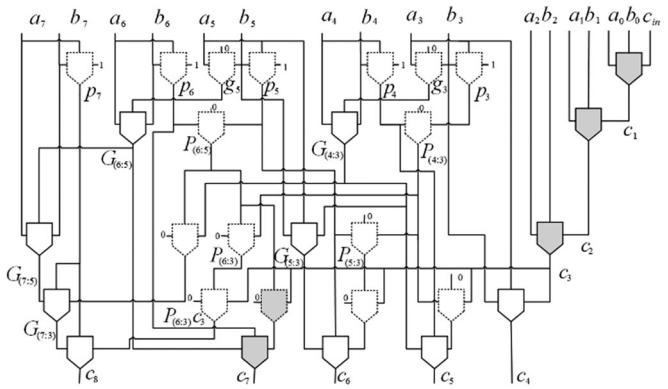


Fig. 7. Five-level 8-bit parallel CGN [17]; dashed $\mathcal{M}$ gates are partially-utilized, shaded ones define the CDP.

Similar efforts for incorporating as many FUMs as possible in constructing majority-based parallel-prefix adders has resulted in more efficient designs than a hypothetical design based on naïve mapping (see Section 4), where the AND and OR gates are directly replaced by PUMs. As an example of the naïve design, we note that an 8-bit Ladner-Fischer (LF) all-PUM parallel-prefix CGN would be composed of 38 PUMs.

4 Prior Work on QCA Fast Addition

Full adders (FAs), as fundamental arithmetic cells, have been the subject of extensive attention by researchers and design engineers for optimized realization in CMOS and recently in new emerging technologies. For example, a breakthrough design of QCA full adders has been proposed [29], where the FA cell is composed of only 3 majority gates and 2 inverters (see Fig. 3).

Regarding mixed PUM/FUM realization of parallel prefix CGNs, the first attempt known to us [16] incorporates 15 PUMs and 8 FUMs. The critical delay path (CDP) of this design travels through 6 majority gates, twice as many levels as that of the standard LF adder. A second work by the same authors [17], shown in Fig. 7, provides a 5-level 8-bit parallel prefix CGN with 16 PUMs and 13 FUMs. However, the one level reduction of CDP at the cost of 62.5 percent more FUMs (with no decrease in PUM count) is hard to justify. Sridharan et al [20] have subsequently offered a 16-bit version of the same design, whose figures of merit will be provided in Section 8 (see Tables 1 and 2).

In a recent development [39], a 2-bit carry generation QCA circuit halves the carry propagation delay at the cost of four additional majority elements. That is, two FUM and

TABLE 2
Delay (Clock Zone) and Cost (Majority Gate)

Adder	n	Delay (CZ)	PUM	FUM	Total M
New	8	5	0	16	16
Ref. [43]	8	6	0	20	20
Ref. [15]-LF	8	9	28	7	35
Ref. [17]	8	8	9	10	19
New	16	9	0	44	44
Ref. [43]	16	10	0	52	52
Ref. [15]-LF	16	15	75	23	98
[20]-hybrid	16	11	31	23	54

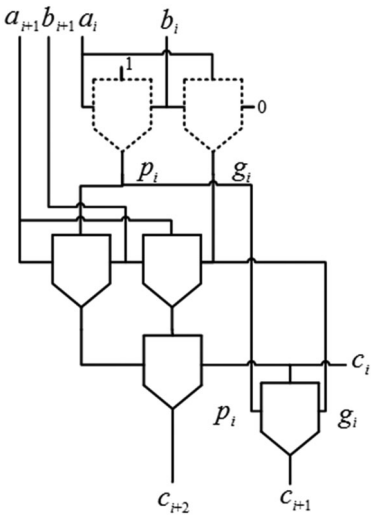


Fig. 8. Two-bit $\mathcal{M}$-based carry generation [39].

two PUM gates, as depicted in Fig. 8. Besides adders, designs for other arithmetic circuits, such as multipliers and dividers, have been proposed for QCA technology [14], [40], [41], [42]. We will not review such designs, as they have no direct bearing on the work reported here.

5 A New Direct Mapping of CGNs to Majority Logic

An all FUM parallel prefix CGN was offered in [43], whose design was based on a special carry genenration scheme that easily allows for all-FUM realizations. For example, the 8-bit LF CGN therein contains 20 FUMs, and no PUMs at all. The basis for the aformentioned carry generation is reproduced here from [43], for ease of reference.

The standard and low-cost 2-bit FUM carry generation is described by Eqn. (1), where there are two $\mathcal{M}$ gates along the CDP. However, it can be made to travel through only one $\mathcal{M}$ gate by using Eqn. (2) (reproduced from [39]), at the cost of three more (i.e., 5 total) $\mathcal{M}$ gates.

$$c_{i+2} = \mathcal{M}(a_{i+1}, b_{i+1}, \mathcal{M}(a_i, b_i, c_i)) \tag{1}$$

$$\begin{aligned} p_i &= \mathcal{M}(a_i, b_i, 1) \\ g_i &= \mathcal{M}(a_i, b_i, 0) \\ c_{i+2} &= \mathcal{M}(\mathcal{M}(a_{i+1}, b_{i+1}, p_i),\ \mathcal{M}(a_{i+1}, b_{i+1}, g_i), c_i). \end{aligned} \tag{2}$$

In the remainder of this section, we present a new compromise design, which achieves a (CDP, $\mathcal{M}$-count) of (1, 3), compared with (2, 2) and (1, 5) of the just-mentioned previously presented designs. We aim to define c_{i+2} in terms of c_i via a single majority gate whose other two inputs depend only on the main addition inputs a_{i+1}, b_{i+1}, a_i, and b_i. Eqn. (3) represents the desired expression, where $A_{i+1:i}$ and $B_{i+1:i}$ represent the contribution of input pairs $(a_{i+1},\ a_i)$ and $(b_{i+1},\ b_i)$ in carry generation. More formally, these radix-4-like variables are expressed by Definition 1, which is validated by Theorem 1.

$$c_{i+2} = \mathcal{M}(A_{i+1:i}, B_{i+1:i},\ c_i) \tag{3}$$

Definition 1 (Radix-4-like carry generation operands). $A_{i+1:i} = \mathcal{M}(a_{i+1}, b_{i+1}, a_i)$ *and* $B_{i+1:i} = \mathcal{M}(a_{i+1}, b_{i+1}, b_i)$.

Theorem 1 (Radix-4-like carry generation). $c_{i+2} = \mathcal{M}(A_{i+1:i}, B_{i+1:i},\ c_i) = \mathcal{M}(\mathcal{M}(a_{i+1}, b_{i+1}, a_i), \mathcal{M}(a_{i+1}, b_{i+1}, b_i),\ c_i)$.

Proof. The conventional expression of c_{i+2} in terms of generate and propagate signals of the corresponding main inputs and c_i leads to the desired result, after some manipulation, as follows.

$$\begin{aligned} c_{i+2} &= g_{i+1}\ + g_i p_{i+1} + c_i p_{i+1}\ p_i \\ &= g_{i+1}\ + g_i p_{i+1} + g_{i+1} p_{i+1} p_i + c_i p_{i+1} p_i + g_{i+1}\ c_i \\ &= g_{i+1}\ + p_{i+1} a_i b_i + g_{i+1} p_{i+1} a_i + g_{i+1} p_{i+1} b_i \\ &\quad + (g_{i+1} + p_{i+1} p_i)\ c_i \\ &= (g_{i+1} + p_{i+1} a_i)\ (g_{i+1} + p_{i+1} b_i) \\ &\quad + (g_{i+1} + p_{i+1} a_i + g_{i+1} + p_{i+1} b_i)\ c_i \\ &= \mathcal{M}\ (g_{i+1} + p_{i+1} a_i, g_{i+1} + p_{i+1} b_i, c_i) \\ &= \ \mathcal{M}\ (\mathcal{M}(a_{i+1}, b_{i+1}, a_i), \mathcal{M}(a_{i+1}, b_{i+1}, b_i),\ c_i) \\ &= \mathcal{M}(A_{i+1:i}, B_{i+1:i},\ c_i) \end{aligned}$$

□

The majority-based radix-4-like carry generation scheme above can be further extended to higher radices, as described and proven in the definitions, lemma, and theorems that follow. The $A_{j-1:i}$ and $B_{j-1:i}$ of Definition 2 represent the carry generation operands for expressing $c_j = \mathcal{M}(A_{j-1:i}, B_{j-1:i}, c_i)$ in terms of c_i, and the corresponding majority group generate and propagate signals are expressed as in Definition 3.

Definition 2 (Higher-radix-like carry generation operands). $A_{j:i} = \mathcal{M}(a_j, b_j, A_{j-1:i})$ *and* $B_{j:i} = \mathcal{M}(a_j, b_j, B_{j-1:i})$, *where we postulate that* $A_{i:i} = a_i$ *and* $B_{i:i} = b_i$ *. Definition 1 represents a special case for* $j = i + 1$.

Similar to the group generate ($G_{j:i}$) and propagate ($P_{j:i}$) signals of the conventional parallel prefix CGN, we can define the corresponding majority-group generate ($\Gamma_{j:i}$) and propagate ($\Pi_{j:i}$) signals. Such designation is supported by the recurrences in Lemma 1 that are exactly the same as those of conventional group generate and propagate signals.

Definition 3 ($\mathcal{M}$-group generate and propagate signals). $\Gamma_{j:i} = A_{j:i}\ B_{j:i}$, $\Pi_{j:i} = A_{j:i} + B_{j:i}$.

Lemma 1. $\Gamma_{j:i} = g_j + p_j \Gamma_{j-1:i}$, $\Pi_{j:i} = g_j + p_j \Pi_{j-1:i}$ *for* $j > i$.

Proof .

$$\begin{aligned} \Gamma_{j:i} = A_{j:i}\, B_{j:i} &= \mathcal{M}\big(a_j, b_j, A_{j-1:i}\big)\mathcal{M}\ \big(a_j, b_j, B_{j-1:i}\big) \\ &= \big(g_j + p_j A_{j-1:i}\big)\ \big(g_j + p_j B_{j-1:i}\big) = g_j\ + p_j A_{j-1:i}\ B_{j-1:i} \\ &= g_j + p_j \Gamma_{j-1:i}. \end{aligned}$$

$$\begin{aligned} \Pi_{j:i} = A_{j:i} + \ B_{j:i} &= \big(g_j + p_j A_{j-1:i}\big)\ +\ \big(g_j + p_j B_{j-1:i}\big) \\ &= g_j + p_j\big(A_{j-1:i} + B_{j-1:i}\big) = g_j\ + p_j \Pi_{j-1:i}. \end{aligned}$$

□

Theorem 2 (Radix-2^j-like carries). $c_{i+j+1} = \mathcal{M}(A_{i+j:i}, B_{i+j:i}, c_i)$

Proof . The proof is by induction on j.

Base ($j = 0$): $c_{i+1} = \mathcal{M}\ (a_i, b_i, c_i) = \mathcal{M}(A_{i:i}, B_{i:i}, c_i)$.

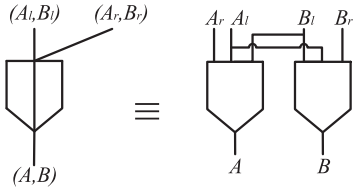


Fig. 9. Notation for, and structure of, the TM gate.

Induction step: $c_{i+j} = \mathcal{M}(A_{i+j-1:i}, B_{i+j-1:i}, c_i)$.

$$\begin{aligned} c_{i+j+1} &= g_{i+j} + p_{i+j}\, c_{i+j} \\ &= g_{i+j} + p_{i+j}\mathcal{M}\left(A_{i+j-1:i}, B_{i+j-1:i}, c_i\right) \\ &= g_{i+j} + p_{i+j}\left(A_{i+j-1:i}B_{i+j-1:i} + \left(A_{i+j-1:i} + B_{i+j-1:i}\right)c_i\right) \\ &= \left(g_{i+j} + p_{i+j}A_{i+j-1:i}B_{i+j-1:i}\right) \\ &\quad + \left(g_{i+j} + p_{i+j}\left(A_{i+j-1:i} + B_{i+j-1:i}\right)c_i\right). \end{aligned}$$

The proof can be completed from the latter by appropriate substitution, per Lemma 1:

$$c_{i+j+1} = \Gamma_{i+j:i} + \Pi_{i+j:i}\, c_i = \mathcal{M}\left(A_{i+j:i}, B_{i+j:i}, c_i\right).$$

□

For example, in radix 8, we have the digits $a_{i+2}a_{i+1}a_i$ and $b_{i+2}b_{i+1}b_i$, with c_{i+3} expressed as in:

$$\begin{aligned} c_{i+3} &= \mathcal{M}\ (A_{i+2:i}, B_{i+2:i}, c_i) \\ &= \mathcal{M}(\mathcal{M}(a_{i+2}, b_{i+2}, A_{i+1:i}), \mathcal{M}(a_{i+2}, b_{i+2}, B_{i+1:i}),\ c_i). \end{aligned} \tag{4}$$

Theorem 3 (Associativity of the $\mathcal{M}$ operation).

$A_{k+j:i} = \mathcal{M}(A_{k+j:j}, B_{k+j:j}, A_{j-1:i})$ and $B_{k+j:i} = \mathcal{M}(A_{k+j:j}, B_{k+j:j}, B_{j-1:i})$.

Proof. We provide the proof only for $A_{k+j:i}$A$_{k+j:i}$, using induction on k. The proof for B$_{k+j:i}$ B$_{k+j:i}$ is similar. □

Base ($k = 0$), is obvious by Definition 2.

Induction step: $A_{k-1+j:i} = \mathcal{M}(A_{k-1+j:j}, B_{k-1+j:j}, A_{j-1:i})$.

Next, we derive $A_{k+j:i}$ per Definition 2 and the induction step, as follows.

$$\begin{aligned} A_{k+j:i} &= \mathcal{M}\left(a_{k+j}, b_{k+j}, A_{k-1+j:i}\right) = g_{k+j} + p_{k+j}A_{k-1+j:i} \\ &= g_{k+j} + p_{k+j}\mathcal{M}\left(A_{k-1+j:j}, B_{k-1+j:j}, A_{j-1:i}\right) \\ &= g_{k+j} + g_{k+j}A_{j-1:i} \\ &\quad + p_{k+j}\left(\Gamma_{k-1+j:j} + \Pi_{k-1+j:j}A_{j-1:i}\right) \\ &= \left(g_{k+j} + p_{k+j}\Gamma_{k-j+j:j}\right) + \left(g_{k+j} + p_{k+j}\Pi_{k-1+j:j}A_{j-1:i}\right). \end{aligned}$$

With proper replacements based on Lemma 1, we arrive at

$$\begin{aligned} A_{k+j:i} &= A_{k+j:j}B_{k+j:j} + \left(A_{k+j:j} + B_{k+j:j}\right)A_{j-1:i} \\ &= \mathcal{M}\left(A_{k+j:j}, B_{k+j:j}, A_{j-1:i}\right). \end{aligned}$$

For example, Theorem 3 can be used for expressing c_{i+4} in term of c_i via 2-bit A and B variables, as in:

$$\begin{aligned} c_{i+4} &= \mathcal{M}(A_{i+3:i}, B_{i+3:i}, c_i) \\ &= \mathcal{M}(\mathcal{M}(A_{i+3:i+2}, B_{i+3:i+2}, A_{i+1:i}), \mathcal{M}(A_{i+3:i+2}, B_{i+3:i+2}, B_{i+1:i}), c_i). \end{aligned} \tag{5}$$

Note the c_i-c_{i+j} ($j > 0$) path goes through only one $\mathcal{M}$ gate. Therefore, the 4 equations derived for c_{i+1}, c_{i+2}, c_{i+3}, and c_{i+4}, can serve as basic equations for a carry-lookahead (CLA) logic block [2] with blocking factor of 4, where the CDP travels through 3 $\mathcal{M}$ levels, while the total cost is 12 $\mathcal{M}$ gates.

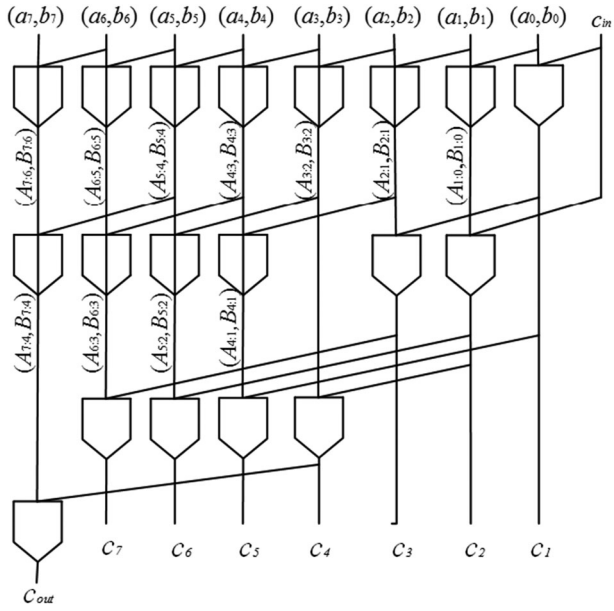


Fig. 10. An 8-bit KS-like CGN with fully utilized $\mathcal{M}$ gates [43].

There are instances of twin majority functions with identical first parameters, and also identical second parameters. See, for example, Definition 2, and Theorems 1 and 3. Therefore, it seems useful to formally define this concept.

Definition 4 (Twin majority gate, TM). *Let (A_l, B_l) and (A_r, B_r) denote arbitrary pairs per Definition 2. The twin majority function output, (A, B) is defined as $A = \mathcal{M}(A_l, B_l, A_r)$ and $B = \mathcal{M}(A_l, B_l, B_r)$. The TM function is given the symbolic representation depicted in Fig. 9.*

To set up a 16-bit CGN, we can use four of the latter CLA blocks, in parallel, to generate the required (A, B) pairs $(A_{i+3:i}, B_{i+3:i})$, $(A_{i+7:i+4}, B_{i+7:i+4})$, $(A_{i+11:i+8}, B_{i+11:i+8})$, and $(A_{i+15:i+11}, B_{i+15:i+11})$, that can serve as inputs to another block which generates, among others, $(A_{i+7:i}, B_{i+7:i})$, $(A_{i+11:i}, B_{i+11:i})$, and $(A_{i+15:i}, B_{i+15:i})$ pairs. The required carries c_{i+1} to c_{i+16} can then be generated as $c_{i+j+1} = \mathcal{M}(A_{i+j:i}, B_{i+j:i}, c_i)$, for $1 \le j \le 16$.

6 Actual Majority-Based CGNs

Parallel-prefix-like $\mathcal{M}$-based CGNs can be readily set up. For example, majority realization of Kogge-Stone (KS)-like and LF-like CGNs with c_{in} are illustrated by Figs. 10 and 11, respectively, which are borrowed from [43]. To show the scalability of such designs, we provide a 16-bit KS-like CGN in Fig. 12 (also borrowed from [43]), where it is easy to verify that structure of the rightmost 8 positions (i.e., the gray FUMs) is same as in the 8-bit architecture of Fig. 10. Note that the output c_{out} in the latter is available in the 4th level, that is, one level later than the conventional 8-bit KS parallel prefix adders with $c_{in} = 0$. This is not important however, since c_{out} is actually delivered at the same time as the most-significant sum bit. Furthermore, it does not delay the carry bits of the most significant 8-bit part of the 16-bit design of Fig. 12.

On the other hand, inclusion of c_{in} in the LF-like architecture of Fig. 11 (for fair comparison with the previous

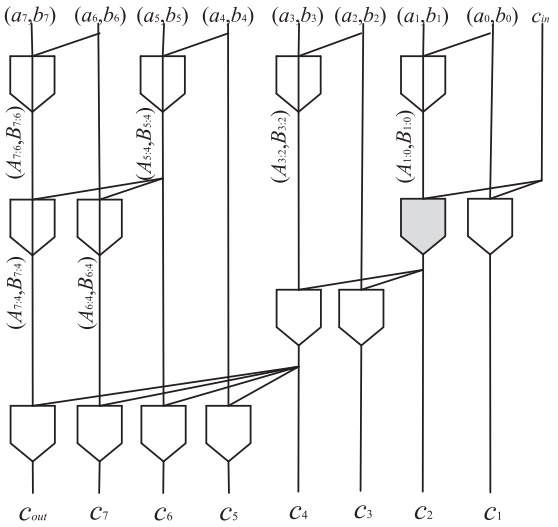


Fig. 11. The 8-bit LF-like parallel CGN with c_{in} [43].

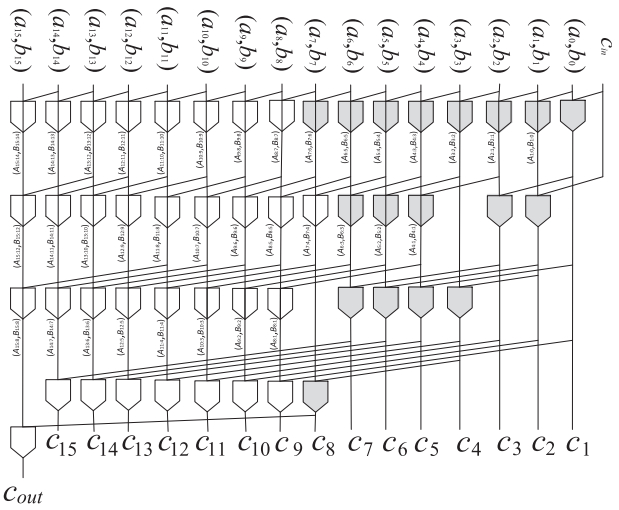


Fig. 12. The 16-bit KS-like CGN with fully utilized $\mathcal{M}$ gates.

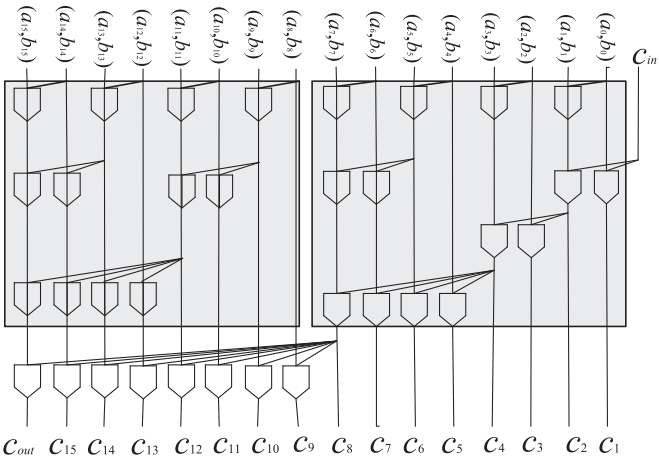


Fig. 13. The 16-bit LF-like parallel CGN with c_{in}.

relevant works [16], [17]) has led to 1-level later delivery of c_5-c_7 (i.e., at the same time as the c_{out}). However, this causes no problem for c_{out}, as was explained above in the KS-like case, but it does matter regarding the carry signals c_5-c_7, which is not problematic for CGNs with more than 8 bits. Nevertheless, no extra parallel prefix level exists in the

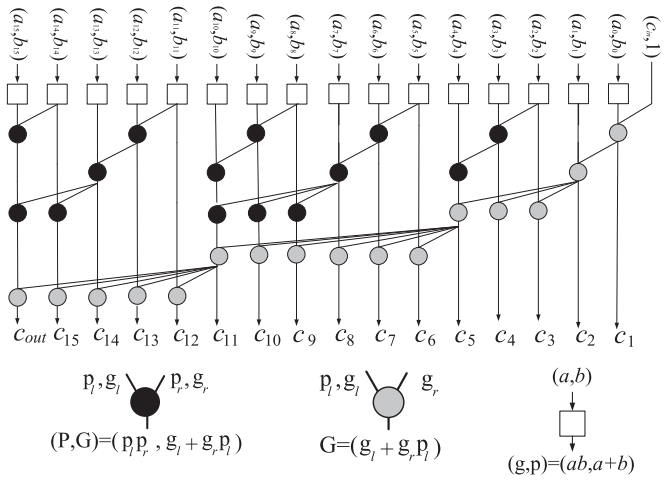


Fig. 14. New 16-bit parallel prefix CGN with c_{in}.

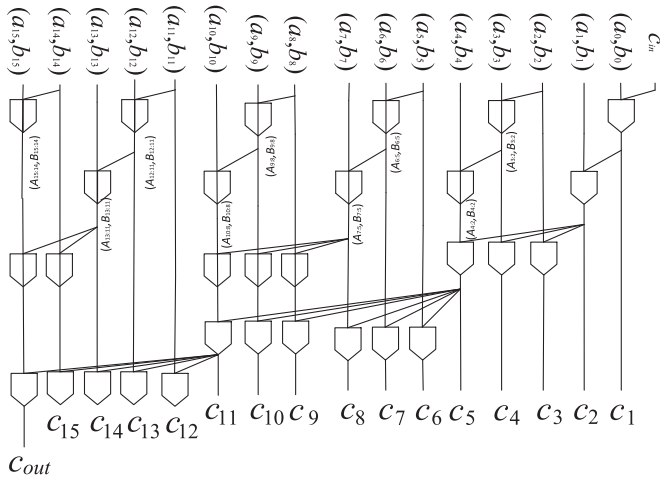


Fig. 15. The new $\mathcal{M}$-based parallel CGN with c_{in}.

architecture of 3-level 8-bit LF-like design with $c_{in} = 0$, where the gray $\mathcal{M}$ gate of Fig. 11 will be omitted. The 16-bit version of Fig. 11 is given in Fig. 13, which contains two slightly modified copies of the architecture of Fig. 11, with the required extra $\mathcal{M}$ gates in the last level.

In the remainder of this section, we discuss briefly the design of further-optimized $\mathcal{M}$-based CGNs. Recalling the LF-like design of Fig. 11, we note that a custom-designed approach can further reduce the costs. This new parallel prefix CGN is depicted by Fig. 14 for $n = 16$, whose corresponding FUM network is illustrated in Fig. 15, bearing a total of 44 FUMs, with no additional level with respect to 16-bit version (with 52 FUMs) of our previous design of Fig. 13. Furthermore, maximum fan-out is reduced from 8 to 6.

7 QCA Realization

QCA layout of the LF-like adder of Fig. 11 (borrowed from [43]) is illustrated in Fig. 16(a). Also, the layouts for 8-bit and 16-bit versions of Fig. 15 are shown in Figs. 16(b) and 16(c), respectively. Dashed rectangles in the layouts represent twin majority gates, with TM counts of 6, 4, and 14 in our previous and new 8-bit and 16-bit designs, respectively. Furthermore, the critical delay paths are also highlighted as heavy black lines towards the bottom of layouts, where the

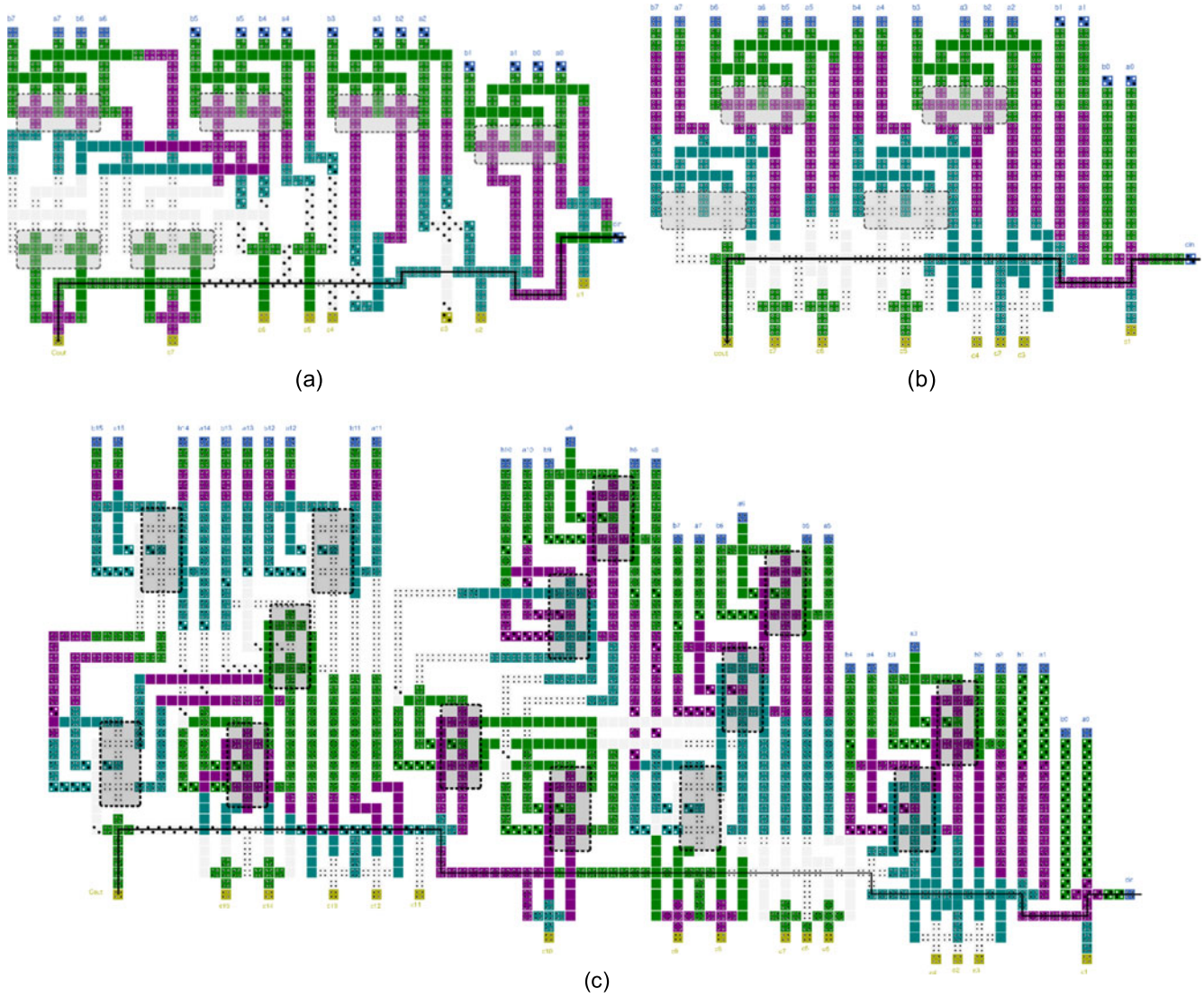


Fig. 16. QCA layouts of the adders of Fig. 11a, 8- and 16-bit versions of Figs. 15b and 15c.

number of zones can be counted easily based on different colors within the critical delay paths; i.e., 6, 5, and 9 for Figs. 16(a), 16(b), and 16(c), respectively. Recalling our discussion on the working frequency (see the 2nd paragraph in Section 3), and the maximum number of cells in one zone being 16, the working frequency of the provided circuits is at least 1 THz. Therefore, the aforementioned number of zones correspond to 1.5, 1.25, and 2.25 ps, since each 4 zones fit in one clock cycle.

The QCADesigner that has provided the latter layouts has reported the corresponding area consumption, as $0.60\ \mu m^2$, $0.51\ \mu m^2$ and $2.02\ \mu m^2$, respectively.

The actual functioning of the new adder is captured by the sample 16-bit I/O pattern of Fig. 17 (also as another output of the QCADesigner simulation), where 33 triple inputs (i.e., a_{15}-a_0, b_{15}-b_0, and c_{in}) are provided. The corresponding outputs (i.e., c_{15}-c_0 and c_{out}) are shown in subsequent rows. Moreover, the clocking patterns for the four zones are illustrated in the bottom 4 rows. For example, the delivery of c_1 signal at Zone 2, per the bottom right portion of Fig. 16(b), can also be captured by the correspondence line between the valid c_1 for the first 16-bit input and the first hold phase of Clock 2. Finally, the corresponding outputs for the rightmost complementary main inputs and $c_{in} = 1$ (i.e., 1010101010101010 + 0101010101010101 + 1) are distinguished via gray shading that read as 1111111111111111.

All the above measures have been obtained via the default QCADesigner parameters. That is we used Coherence vector simulation engines under $\text{Temperature} = 1\text{K}$, $\text{Relaxation time} = 1 \times 10^{-15}\,\text{s}$, $\text{Time step} = 1 \times 10^{-16}\,\text{s}$, $\text{Total simulation time} = 7 \times 10^{-11}\,\text{s}$, $\text{Radius of effect} = 80\,\text{nm}$, $\text{Relative permittivity} = 12.9$, $\text{Layer separation} = 11.5\,\text{nm}$.

8 Evaluations and Comparisons

We evaluate and compare the figures of merit for the best previous majority-based parallel prefix CGN of [17] (see Fig. 7), our previous relevant work in [43] (see Fig. 11), and the new design just presented (see Fig. 15).

Table 2 contains the number of PUMs and FUMs that compose all the new and selected previous 8-bit and 16-bit adders. Regarding the 16-bit version of [17], no measures were found there, nor a detailed figure was available that could help in our evaluation. However, a later work by the same authors [20] provides the required measures for the corresponding 16-bit adder that is included in Table 2.

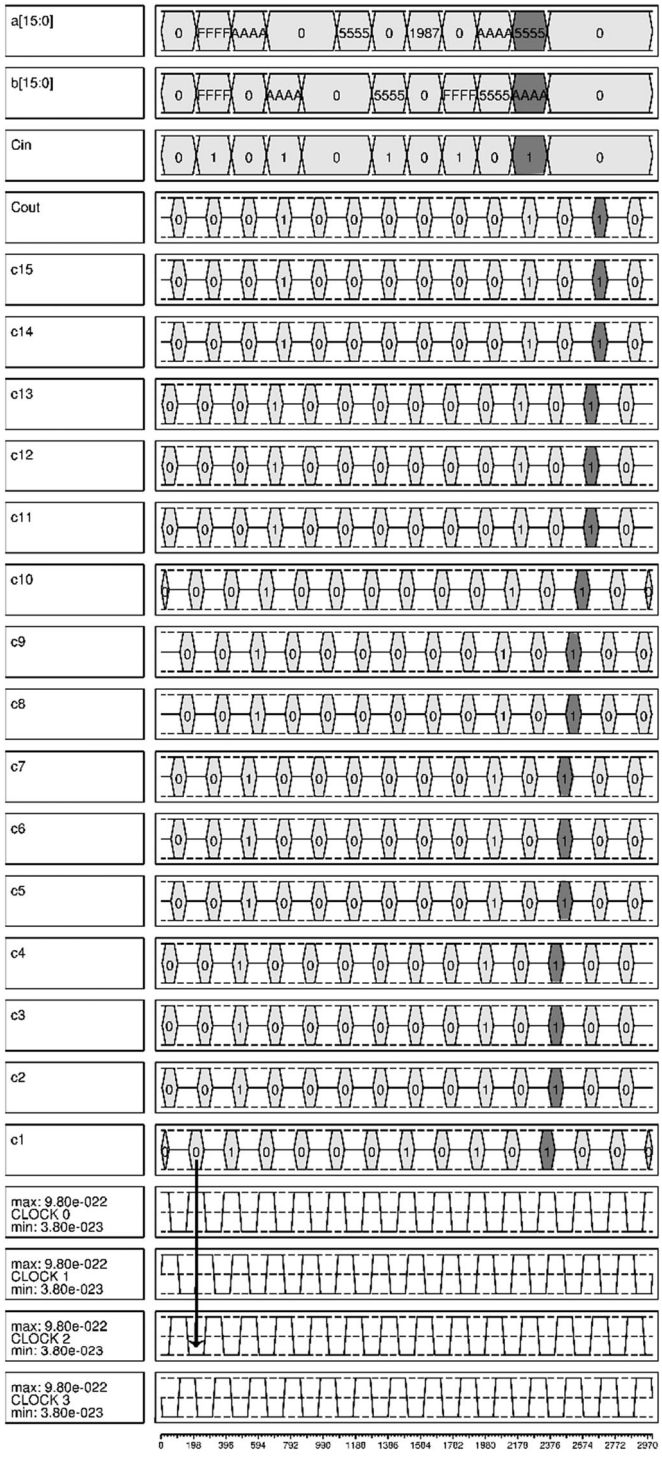


Fig. 17. Sample I/O for Fig. 16c.

Delay figures are measured by the number of clock zones (CZ) within the critical delay path of each design.

Further comparison, based on the QCADesigner outputs, with the work of [17] is not possible, since the provided layout therein is not complete. However, we could figure out the number of $\mathcal{M}$ gates in the critical delay path of its 8-bit version, which can be considered as an estimation of the delay. The results are compiled in Table 3 for selected bit widths.

Another evaluation method for QCA circuits [44] calculates a composite figure of merit based on time delay (T), number of majority gates (M), number of inverters (I), and

TABLE 3
Number of $\mathcal{M}$ Gates in the CDP

n	8	16
New	4	5
Reference [43]	4	5
Reference [15]	5	6
Reference [17]	5	-
Reference [20]	6	8

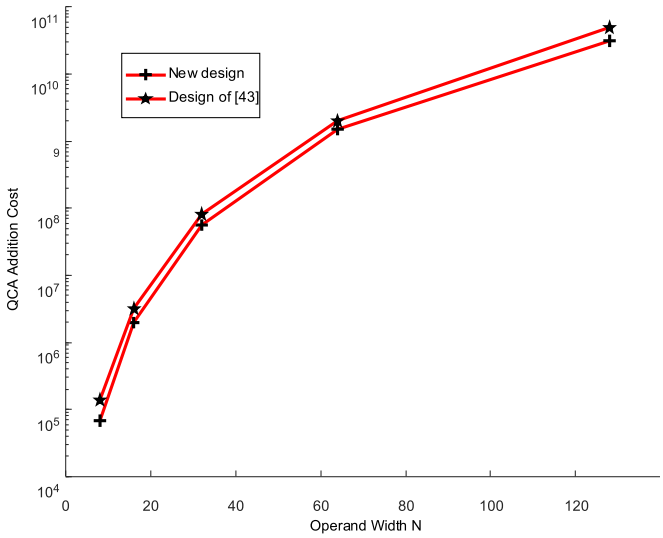


Fig. 18. QCA cost function for the new design and that of [43].

TABLE 4
QCA-Specific Cost Function Results

size	new	ratio	[43]	ratio	[20]	ratio
8	$6.7e+04$	1.0	$1.3e+05$	1.9	$2.7e+05$	4.0
16	$1.9e+06$	1.0	$3.1e+06$	1.6	$5.6e+06$	2.9
32	$5.6e+07$	1.0	$8.0e+07$	1.4	-	-
64	$1.5e+09$	1.0	$1.9e+09$	1.2	-	-
128	$3.1e+10$	1.0	$4.9e+10$	1.5	-	-

number of crossovers (C), according to Eqn. (6), where the parameters p, k, and l signify the impact of T, M, and C measures, respectively. Values of these parameters can be adjusted depending on the overall design optimization goal. Nevertheless, We use the default values $p = k = l = 2$, for the aforementioned impact parameters.

The $k = 2$ choice is justified by the fact that M effects both on the irreversible power and complexity. Likewise, since C effects on fabrication issue and complexity, $l = 2$ is enforced. Finally considering the same weight for delay as the other two parameters leads to $p = 2$.

$$Cost_{QCA} = \left(M^k + I + C^l\right) * T^p. \tag{6}$$

The corresponding cost functions of our previous work [43] and the one presented here are plotted in Fig. 18 for 8-, 16-, 32-, 64-, and 128-bit operands. These plots demonstrate the superiority of the present work in terms of the composite cost function. The actual measures are compiled in Table 4, where the ratios clearly show the superiority of the proposed designs such that 20-90 percent cost reduction is evident. We derived the pertinent cost parameters for the

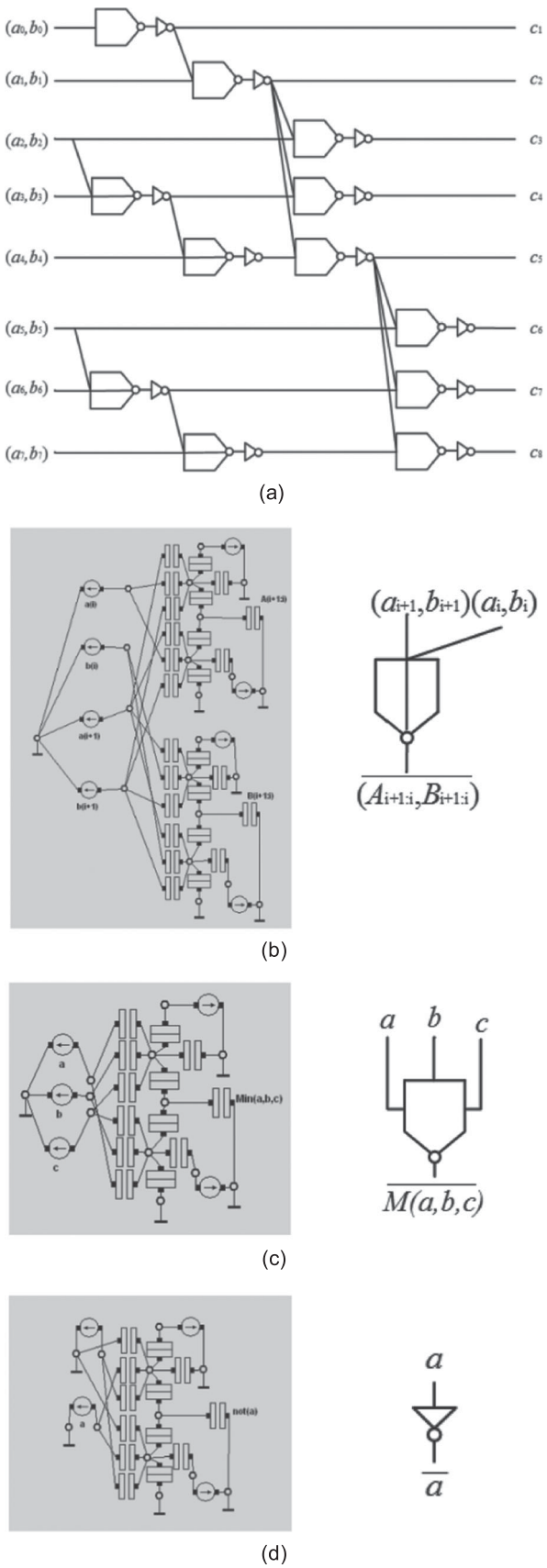


Fig. 19. Twin Miniority gate illustration of the SET CGN.

new and old [43] designs via inspecting the counts of utilized $\mathcal{M}$ gates and crossovers, where we considered two crossovers per one $\mathcal{M}$ gate (see Fig. 16).

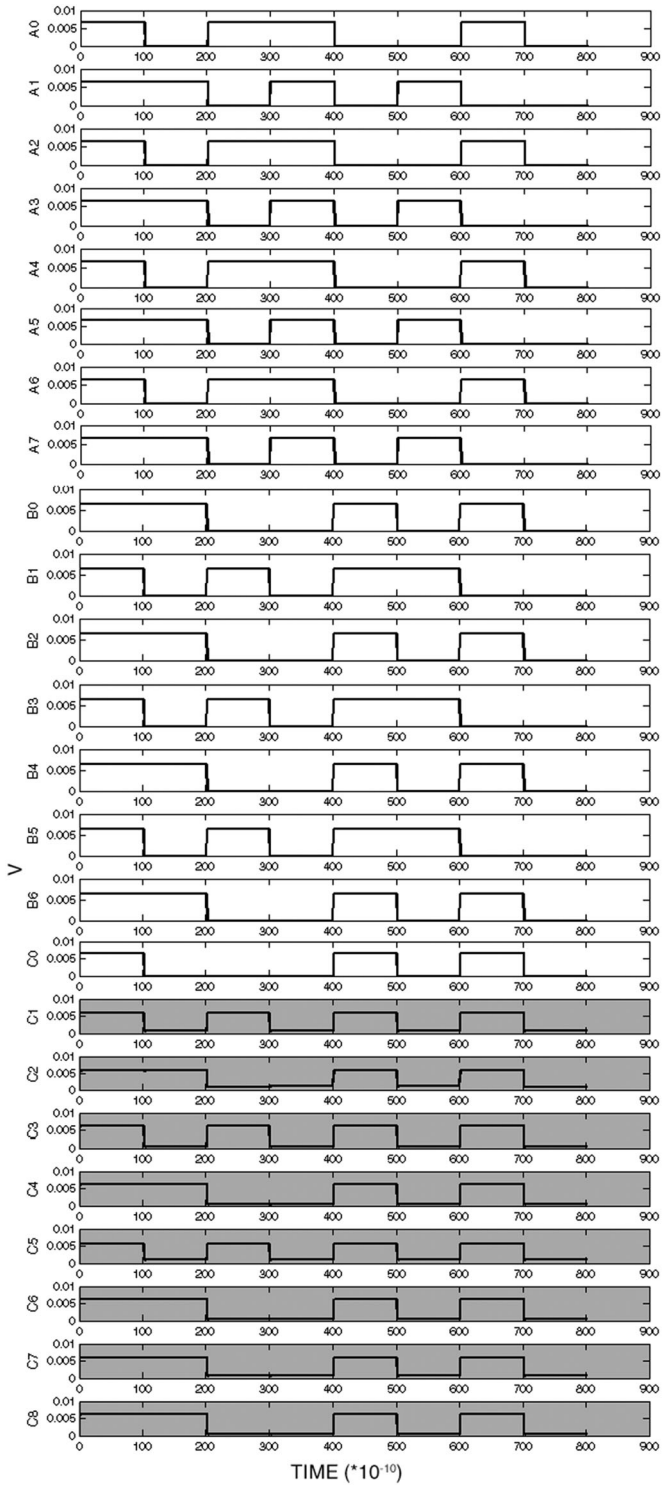


Fig. 20. I/O Sample for the SET realization.

As for the delay figures, our estimates are based on the number of cascaded *TM*s. For example, in the layouts of Fig. 16, length of the solid black critical delay path, extending between the two sides toward the bottom of the diagrams, is directly proportional to the number of *TM*s along the path, that is, 5, at most 4, and 13 *TM*s for Figs. 16(a), 16(b), and 16(c), respectively. Deriving similar estimates is impossible for [15], [17], and 32-, 64- and 128-bit hypothetical layouts of [20], due to lack of adequate information.

9 Extending Our Design Methods Beyond QCA

Another new emerging technology of interest, for which the required simulation tool (i.e., SIMON [45]) is readily available, is the single electron transistor (SET) paradigm. As was mentioned in Section 2.1, the basic cell in this technology is the minority and inverter functions. Therefore, we modified the 8-bit CGN of Fig. 15 accordingly as in Fig. 19(a), where the SET realizations of twin minority (b), minority (c), and inverter (d) gates are also illustrated.

Furthermore, a complete SIMON version of the corresponding SET CGN is provided in the Appendix, where the two minority gates of each twin minority are shown separately. We used the corresponding SIMON output file to produce Fig. 20 via MATLAB, which illustrates an I/O sample with similar input as that of Fig. 17. Thus far, we have not found any prior relevant work on the design and evaluation of adders with the SET technology for comparison.

10 Conclusion

Our primary contribution in this paper is a formulation of the carry recurrence directly in terms of majority gates in a way that allows us to employ fully-utilized $\mathcal{M}$-gates, as opposed to partially-utilized ones. One construct that has allowed us to achieve our speed and circuit efficiency goals is the *TM*-gate parallel-prefix operator that possesses the important associativity attribute, and thus lends itself to the synthesis of parallel-prefix networks in a manner similar to those used with today's more conventional circuit technologies.

Our designs, practically demonstrated for QCA and SET technologies, are applicable to several other emerging technologies (including TPL, MTJ, and NBM) that offer efficient realization of majority gates.

Besides formally deriving the carry recurrence using only fully utilized $\mathcal{M}$ gates, we demonstrated fast carry-network designs in the form of LF-like and KS-like parallel-prefix networks that exhibit the same attributes as the original Ladner-Fischer and Kogge-Stone designs.

Given that prior fast-adder designs exist for QCA, we focused on implementing our ideas in QCA technology to facilitate comparisons. A key to greater efficiency in our approach is the full use of $\mathcal{M}$-gate inputs, in contrast to partial use that results when emulating AND and OR gates. We also showed that our ideas are applicable to SET technology, providing additional evidence that it is the strength of majority logic, rather than other particulars of QCA, that leads to desired attributes.

This work constitutes a beginning in the efficient use of new majority-friendly technologies for realizing fast arithmetic circuits. Not all results derived with QCA and SET will carry over directly to other technologies surveyed in Section 2 and others that may emerge in future.

Layouts and some other circuit implementation details will no doubt vary, creating a need for optimizations in each case. However, unless serious unanticipated overheads arise in the course of implementation and optimization, we expect that similar advantages will accrue in these other cases as well. We plan to pursue improvements and fine-tuning of our QCA and SET designs and to investigate the extent to which the designs carry over to other technologies and implementation styles.

An intriguing possibility for future investigation is to consider the incorporation of reliability features [46] using triple-modular redundancy with voting, given that the required voting element is essentially a single $\mathcal{M}$ gate.

Acknowledgments

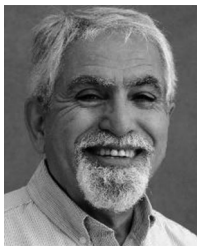
Dr. Jaberiur's research was supported in part by two sources: IPM under Grant CS1397-2-03 and Shahid Beheshti University.

References

[1] Q. Xu, N. S. Kim, and T. Mytkowicz, "Approximate computing: A survey," *IEEE Design Test*, vol. 33, no. 1, pp. 8–22, Feb. 2016.
[2] B. Parhami, Computer arithmetic: Algorithms and hardware designs, London, U.K.: Oxford Univ. Press, 2nd ed., 2010.
[3] C. Babbage, Passages from the life of a philosopher, London, 1864 (Reissued by London, U.K.: Cambridge Univ. Press, 2011; also available online).
[4] G. B. Rosenberger, "Simultaneous carry adder," US Patent 2 966 305, Dec. 27, 1960.
[5] D. Harris, "A taxonomy of parallel prefix networks," in *Proc. Asilomar Conf. Signals Syst. Comput.*, 2003, vol. 2, pp. 2213–2217.
[6] T. B. Preusser and R. G. Spallek, "Mapping basic prefix computations to fast carry-chain structures," in *Proc. Int. Conf. Field Programmable Logic Appl.*, 2009, pp. 604–608.
[7] C. S. Lent, P. Tougaw, W. Porod, and G. Bernstein, "Quantum cellular automata" *Nanotechnol.*, vol. 4, pp. 49–57, 1993.
[8] P. Tougaw and C. Lent, "Logical devices implemented using quantum cellular automata," *J. Appl. Physics*, vol. 75, no. 3, pp. 1818–1825, Feb. 1994.
[9] H. Iwamura, M. Akazawa, and Y. Amemiya, "Single-electron majority logic circuits," *IEICE Trans. Electron.*, vol. E81-C, no. 1, pp. 42–48, Jan. 1998.
[10] W. Porod and M. Niemier, "Better computing with magnets: The simple bar magnet, shrunk down to the nanoscale, could be a powerful logic device," *IEEE Spectrum*, vol. 52, no 9, pp. 44–48 & 59–60, Sep. 2015.
[11] L. M. Adleman, "Molecular computation of solutions to combinational problems," *Sci.*, vol. 266, no. 5187, pp. 1021–1024, 1994.
[12] W. Li, Y. Yang, H. Yan, and Y. Liu, "Three-input majority logic gate and multiple input logic circuit based on DNA strand displacement," *Nano Lett.*, vol. 13, pp. 2980–2988, 2013.
[13] L. O. Chua, "Memristor—The missing circuit element," *IEEE Trans. Circuit Theory*, vol. 18, no. 5, pp. 507–519, Sep. 1971.
[14] H. Cho and E. Swartzlander, "Adder designs and analyses for quantum-dot cellular automata," *IEEE Trans. Nanotechnol.*, vol. 6, no. 3, pp. 374–383, May 2007.
[15] V. Pudi and K. Sridharan, "Low complexity design of ripple Carry and Brent–Kung Adders in QCA," *IEEE Trans. Nanotechnol.*, vol. 11, no. 1, pp. 105–119, Jan. 2012.
[16] V. Pudi and K. Sridharan, "Efficient design of a hybrid adder in quantum dot cellular automata," *IEEE Trans. VLSI Syst.*, vol. 19, no. 9, pp. 1535–1548, Sep. 2011.
[17] V. Pudi and K. Sridharan, "New decomposition theorems on majority logic for low-delay adder designs in quantum dot cellular automata," *IEEE Trans. Circuits Syst. II*, vol. 59, no. 10, pp. 678–682, Oct. 2012.
[18] W. Liu, E. E. Swartzlander Jr., and M. O'Neill, *Design of Semiconductor QCA Systems*, Norwood, MA, USA: Artech House, 2013.
[19] D. Abedi, G. Jaberipur, and M. Sangsefidi, "Coplanar full adder in quantum-dot cellular automata via clock-zone based crossover," *IEEE Trans. Nanotechnol.*, vol. 14, no. 3, pp. 497–504, May 2015.
[20] K. Sridharan and V. Pudi. "Design of Ripple Carry and prefix adders in QCA," *Design of Arithmetic Circuits in Quantum Dot Cellular Automata Nanotechnology*, pp. 27–55. Berlin, Germany: Springer, 2015.
[21] K. Walus, G. Schulhof, G. A. Jullien, R. Zhang, and W. Wang, "Circuit design based on majority gates for applications with quantum-dot cellular automata," in *Proc. 38th Asilomar Conf. Signals Syst. Comput.*, 2004, pp. 1354–1357.
[22] R. Zhang, R. Zhang, K. Walus, W. Wang, and G. A. Jullien, "A method of majority logic reduction for quantum cellular automata," *IEEE Trans. Nanotechnol.*, vol. 3, no. 4, pp. 443–450, Dec. 2004.

[23] S. Muroga, *Threshold Logic and Its Applications*, Hoboken, NJ, USA: Wiley, 1971.
[24] W. S. McCulloch and W. H. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophysiology*, vol. 5, pp. 115–133, 1943.
[25] V. Beiu, J. M. Quintana, and M. J. Avedillo, "VLSI implementation of threshold logic: A comprehensive survey," *IEEE Trans. Neural Netw.*, vol. 14, no. 5, pp. 1217–1243, Sep. 2003.
[26] D. Volger, "The roadmap to 5 nm: Covergence of many solutions needed," *SEMI*, http://www.semi.org/en/node/55926, May 2015.
[27] M. Razavy, *Mohsen, Quantum Theory of Tunneling*, Singapore: World Scientific, 2003.
[28] J. R. Isbell, "Median Algebra," *Trans. Amer. Mathematical Society*, vol. 260, no. 2, pp. 319–362, Aug. 1980.
[29] W. Wang, K. Walus, and G. A. Jullien, "Quantum-dot cellular automata adders," in *Proc. 3rd IEEE Conf. Nanotechnol., IEEE-NANO*, Aug. 2003, vol. 1, pp. 461–464.
[30] M. T. Niemier, *Designing Digital Systems in Quantum Cellular Automata Master's Thesis, University of Notre Dame*, Indiana, USA: Notre Dame, 2004.
[31] J. Bhattacharya, S. Srimani, S. Panda, and B. Maji, "Comparative analysis of noise, power, delay and area of different full adders in 45 nm technology," *J. Electron. Commun. Eng.*, vol. 9, no. 3, pp. 12–23, 2014.
[32] S. Srivastava, A. Asthana, S. Bhanja, and S. Sarkar, "QCAPro-an error-power estimation tool for QCA circuit design." in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2011, pp. 2377–2380.
[33] T. Ohshima and R. A. Kiehl, "Operation of bistable phase-locked single-electron tunneling logic elements," *J. Appl. Physics*, vol. 80, pp. 912–923, Jul. 1996.
[34] H. A. H. Fahmy and R. A. Kiehl, "Complete logic family using tunneling-phase-logic devices," in *Proc. 11th Int'l Conf. Microelectronics*, 1999, pp. 153–156.
[35] D. M. Bromberg, D. H. Morris, L. Pileggi, and J.-G. Zhu, "Novel STT-MTJ device enabling all-metallic logic circuits," *IEEE Trans. Magnetics*, vol. 48, no. 11, pp. 3215–3218, Nov. 2012.
[36] S. Lee, S. Choa, S. Lee, and H. Shin, "Magneto-logic device based on a single-layer magnetic tunnel junction," *IEEE Trans. Electron. Devices*, vol. 54, no. 8, pp. 2040–2044, Aug. 2007.
[37] G. S. Rose, J. Rajendran, H. Manem, R. Karri, and R. E. Pino, "Leveraging memristive systems in the construction of digital logic circuits," *Proc. IEEE*, vol. 100, no. 6, pp. 2033–2049, Jun. 2012.
[38] S. Kvatinsky, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "Memristor-based material implication (IMPLY) logic: Design principles and methodologies," *IEEE Trans. VLSI Syst.*, vol. 22, no. 10, pp. 2054–2066, Oct. 2013.
[39] S. Perri, P. Corsonello, and G. Cocorullo, "Area-delay efficient binary adders in QCA," *IEEE Trans. VLSI Syst.*, vol. 22, no. 5, pp. 1174–1179, Jun. 2013.
[40] E. E. Swartzlander, H. Cho, I. Kong and S. W. Kim, "Computer arithmetic implemented with QCA: A progress report," in *Proc. Conf. Record 44th Asilomar Conf. Signals Syst. Comput.*, 2010, pp. 1392–1398.
[41] W. Liu, L. Lu, M. Máire, E. E. Swartzlander, and R. Woods, "Design of quantum-dot cellular automata circuits using cut-set retiming," *IEEE Trans. Nanotechnol.*, vol. 10, no. 5, pp. 1150–1160, Sep. 2011.
[42] D. Abedi and G. Jaberipur, "Coplanar QCA serial adder and multiplier via clock-zone based crossover," in *Proc. 18th CSI Int. Symp. Comput. Architecture Digital Syst.*, 2015, pp. 1–4.
[43] G. Jaberipur, B. Parhami, and D. Abedi, "A formulation of fast carry chains suitable for efficient implementation with majority elements," in *Proc. IEEE 23nd Symp. Comput. Arithmetic*, Jul. 2016, pp. 8–15.
[44] W. Liu, L. Lu, M. O'Neill, and E. E. Swartzlander, "A first step toward cost functions for quantum-dot cellular automata designs," *IEEE Trans. Nanotechnol.*, vol. 13, no. 3, pp. 476–487, May 2014.
[45] C. Wasshuber and H. Kosina, "SIMON-A simulator for single-electrontunnel devices and circuits," *IEEE Trans. Comput.-Aided Design*, vol. 16, no. 9, pp. 937–944, Jul. 1997.
[46] J. Han, E. Taylor, J. Gao, and J. Fortes, "Reliability modeling of nanoelectronic circuits," in *Proc. Proc. 5th IEEE Conf. Nanotechnol.*, Jul. 2005, vol. 1, pp. 104–107.

Ghassem Jaberipur received the BS degrees in electrical engineering, in 1974, the MS degree in engineering from the University of California, Los Angeles, in 1976, the MS degree in computer science from the University of Wisconsin, Madison, in 1979, and the PhD degree in computer engineering from the Sharif University of Technology, in 2004, (where he's been recognized as one of the 50 distinguished graduates for years 1966-2016). He is a professor of computer engineering in the Department of Computer Science and Engineering of Shahid Beheshti University, Tehran, Iran. His main research interests include computer arithmetic. He is also affiliated with the School of Computer Science, Institute for Research in Fundamental Sciences (IPM), in Tehran, Iran.



Behrooz Parhami received the BSEE degree from Tehran University, in 1968, the MS degree from Oregon State University, in 1970, and the PhD degree from the University of California, Los Angeles, in 1973. He is a professor of electrical and computer engineering, and former associate dean for academic personnel, College of Engineering, University of California, Santa Barbara, where he teaches and does research in computer arithmetic, parallel processing, and dependable computing. He is a life fellow of the IEEE, a fellow of IET and British Computer Society, and recipient of several other awards (including a most-cited paper award from *J. Parallel & Distributed Computing*). He has written six textbooks and more than 300 peer-reviewed technical papers. Professionally, he has served on journal editorial boards (including multiple IEEE Transactions) and conference program committees and is also active in technical consulting.

Dariush Abedi received the BS degree in hardware computer engineering from the Department of Computer Engineering and Information Technology, Sadjad University of Technology, Mashhad, in 2011, and the MS degree in computer engineering from the Department of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran, in 2015. He is currently working toward the PhD degree in computer architecture from Shahid Beheshti University. His research interests include computer arithmetic, nano-technology, and embedded system design.