Efficient Majority Logic Parallel-Prefix Adder Design

Constantinos Efstathiou, Ioannis Kouretas, Paris Kitsos

***Abstract*—Beyond CMOS nanotechnology has been attracted interest by many researchers. The logical fundamental elements of many nanotechnologies are the majority, minority gates and inverters. The design of efficient adder systems and especially the parallel prefix adders is of very importance. In this paper efficient majority logic implementation of parallel prefix is introduced. The proposed methodology can be generalized to any parallel prefix structure design. Moreover, the proposed majority logic parallel prefix adder designs demonstrate decreased circuit complexity when compared to the literature.**

***Index Terms*—parallel-prefix adders, majority logic, emerging nanotechnologies, arithmetic circuits.**

I. Introduction

SCaling-down of CMOS technology has been predicted to reach to a limit in the forthcoming years [1]. In order to mitigate that a lot of beyond-CMOS devices have been studied [2]–[14]. The fundamental logic components of several developing nanotechnologies include majority, minority gate and inverter [15], [16]. Forming minority logic networks from their corresponding majority logic ones is straightforward. Thus, the efficient majority logic synthesis of digital circuits emerges for the beyond-CMOS era. In particular designing arithmetic circuits is of high importance. Since addition is a fundamental function in practically all digital processors, the design of efficient adders [17] is crucial. Numerous designs for high-performance adders have been studied. Parallel-prefix adders are a significant class among them due to their ability to produce high performance low-complexity hardware [17]. Majority logic parallel-prefix adders are proposed In [18]–[25]. Among them the most competitive designs are demonstrated in [24].

This manuscript extends the majority logic carry bit computation methodology from [26] to majority logic parallel-prefix adder design. By following this process any parallel-prefix adder architecture can be constructed based to majority logic. Additionally, new parallel-prefix adder architectures with efficient majority logic are introduced. The proposed architectures are designed using a novel method for computing sum bits. The novel adder designs that are proposed are implemented with significantly decreased number of majority gates (up to 38.5%), a smaller maximum fan out, or equal complexity and fanout but with diminished by one majority gate level than the initial ones. By utilizing cutting-edge nanotechnologies, the proposed adder design can be used to efficiently create efficient arithmetic units.

Tthe remainder of the paper is organized as follows. In section II, majority logic and emerging nanotechnologies are briefly discussed. Section III introduces the proposed majority logic parallel-prefix adders' design. Section IV quantitates complexity and provides comparisons with the most efficient majority logic parallel-prefix adder designs in the literature. Section V finalizes the paper with conclusions.

II. Majority logic and emerging nanotechnologies

The fundamental logic components of various nanotechnologies are majority, minority gates and inverters. The (3-input) majority gate (MG) carries out the following logic function:

$$M\left(a,b,c\right)=ab\vee\left(a\vee b\right)c \tag{1}$$

By using MG and the inverter a complete logic set can be formulated. The two input OR and AND fundamental logic operators are implemented in majority logic in accordance with the $a\vee b=M(a,b,1)$ and $ab=M\left(a,b,0\right)$ equations, respectively. Moreover, the two-input logic operators, NOR and NAND are implemented in minority logic by using the equations $\overline{a\vee b}=\overline{M}\left(a,\,b,\,1\right)$ and $\overline{ab}=\overline{M}\left(a,b,0\right)$, respectively. Therefore MG forms also a complete logic gate set.

It is simple to convert majority logic networks into their matching minority logic networks. Minority logic circuits produced by this process depict equal circuit delay complexity as their majority logic counterparts. As a result, the only thing we address in this work is the effective design of parallel-prefix adders. The following text provides a brief introduction to a few beyond-CMOS technologies that use majority logic as the design primitive.

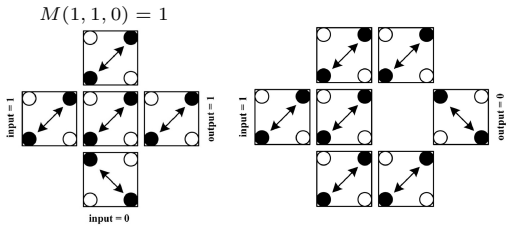
MGs and inverters are the fundamental building blocks of logic circuit implementation in quantum-dot cellular automata (QCA) technology [2], [3]. The cells used in QCA devices are made up of four quantum dots, or electron locations, positioned at a square's corners. Only two stable states exist each one of which can be achieved by charging a cell with two electrons that can tunnel between dots. Fig. 1a depicts the implementation of a MG in QCA technology, whereas in Fig. 1b the implementation of the QCA inverter is depicted.

Memristors are non-volatile memory components with extremely low power consumption that can alter their resistance state in response to the amount of charge applied to them. The fundamental building blocks of logic operations in memristive

Constantinos Efstathiou is with Dept. of Computer Engineering and Informatics University of West Attica Athens, Greece

Ioannis Kouretas is with Dept. of Electrical and Computer Engineering University of Patras, Greece

Paris Kitsos is with Dept. of Electrical and Computer Engineering University of Peloponnese, Greece



(a) (b)

Fig. 1. QCA majority gates

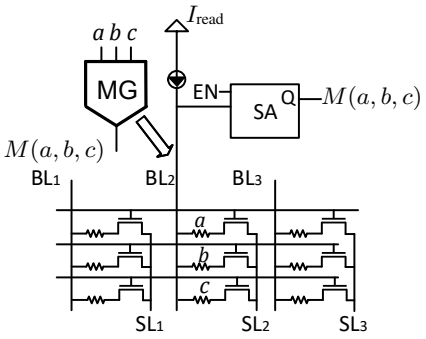


Fig. 2. A ReRAM based majority gate. (SA: sense amplifier)

logic are the resistance states of memristors [4]. By enabling logic gates to be executed in an array configuration, memristive logic has recently been utilized for in-memory computing [5]. Fig. 2 illustrates the MG implementation by adopting a Resistive Random Access Memory (ReRAM) array.

Spin-wave technology, all spin logic, and spin-transfer torque technology are all included in the spin-based MG implementation. Magnetic tunnel junctions (MTJs) can be used to realize a magnetic graphene graph (MG), using the spin-transfer torque technology as an example [6]. An MTJ is a type of emerging spintronic technology consisting of two ferromagnetic layers divided by a thin tunnel barrier that acts as an insulator. There is a fixed magnetization in one free layer and a changeable magnetization in the other. A small amount of information is encoded in the junction's low or high resistance as a result. MGs are also the basic building blocks in nanomagnetic logic [8] and other technologies [9], [10].

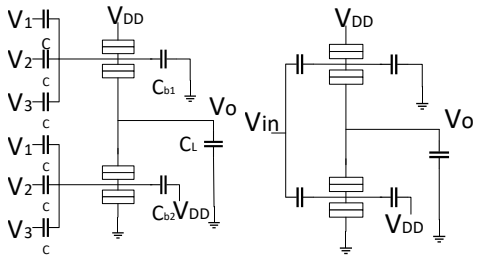
The single electron tunneling (SET) technology can also be used to create energy-efficient and compact minority gate digital circuits [11], [12]. The SET implementation of a MG is shown in Fig. 3a, while a SET inverter is depicted in Fig. 3b. In addition, the fundamental component of tunneling phase logic (TPL) circuits is the minority gate [13], [14].

III. Majority logic design of parallel adders

Let A and B be n-bit binary numbers that are to be added, expressed as $A = a_{n-1}a_{n-2}\dots a_1a_0$ and $B = b_{n-1}b_{n-2}\dots b_1b_0$, respectively. For the binary addition of A and B, the carries c_is are generated based on the following recursive formula:

$$c_i = g_i \vee p_i c_{i-1}, \tag{2}$$

where the bit generate signal is $g_i = a_i b_i$, while the propagate signal is $p_i = a_i \vee b_i$, for the inclusive-OR addition and $p_i = a_i \oplus b_i$, for the exclusive-OR one.



(a) (b)

Fig. 3. SET minority gate (a) and inverter (b)

The inclusive-OR addition is adopted for the proposed adder designs, which is of higher efficiency for the implementation of carry generation using majority logic [26]. Expanding equation (2) for the computation of the carry c_i with $c_{-1} = 0$ it is derived that

$$c_i = g_i \vee p_i g_{i-1} \vee p_i p_{i-1} g_{i-1} \vee \dots \vee p_i p_{i-1} \dots p_1 g_0 \tag{3}$$

The terms $G_{m:k}$ and $P_{m:k}$ stand for the group generate and group propagate signals, while the term $Q_{m:k}$ denotes a new modified group generate signal derived from bits $m, m-1, ..., k$. These signals are defined as follows:

$$G_{m:k} = g_m \vee p_m g_{m-1} \vee \dots \vee p_m p_{m-1} \dots p_{k+1} g_k \tag{4}$$

$$Q_{m:k} = g_m \vee p_m g_{m-1} \vee \dots \vee p_m p_{m-1} \dots p_{k+1} p_k \tag{5}$$

$$P_{m:k} = p_m p_{m-1} \dots p_{k+1} p_k \tag{6}$$

The proposed design also adopts partial group generate signals the definition of which is as follows:

$$G_{m:ka} = g_m \vee p_m g_{m-1} \vee \dots \vee p_m p_{m-1} \dots p_{k+1} a_k \tag{7}$$

$$G_{m:kb} = g_m \vee p_m g_{m-1} \vee \dots \vee p_m p_{m-1} \dots p_{k+1} b_k \tag{8}$$

The computation of the carries of the proposed adder design is based on the following Lemmas proven in [26]:

Lemma 1: If $g_i = a_i b_i$,$p_i = a_i \vee b_i$, then

$$g_i \vee p_i f = M\left(a_i, b_i, f\right) \tag{9}$$

Lemma 2: It holds the following:

$$G_{i:j} \vee P_{i:j} f = G_{i:j} \vee Q_{i:j} f \tag{10}$$

Lemma 3: The following equations hold for the group generate signals:

$$G_{i:j} = G_{i:ja} G_{i:jb} \tag{11}$$

$$Q_{i:j} = Q_{i:ja} Q_{i:jb} \tag{12}$$

The subsequent Lemma [18] is also used in the proposed design.

Lemma 4: It holds the following:

$$M\left(a, b, \overline{M\left(a, b, c\right)}\right) = M\left(a, b, \ c\right) \tag{13}$$

A. Carry bit computation

Using equations (3), (6) we get that the i-th carry of the binary adder is $c_i = G_{i:0}$. We have that

$$G_{i:0} = G_{i:k} \vee P_{i:k} G_{k-1:0} \tag{14}$$

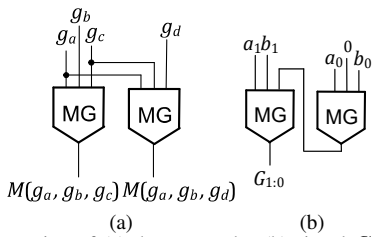


Fig. 4. Implementation of (a) the tree nodes (b) signal $G_{1:0}$

According to Lemmas 2, 3, we get that $G_{i:0} = G_{i:k} \vee Q_{i:k}G_{k-1:0}$, or $G_{i:0} = G_{i:ka}G_{i:kb} \vee (G_{i:ka}G_{i:kb})G_{k-1:0}$. That is,

$$G_{i:0} = M\left(G_{i:\mathrm{ka}}, G_{i:\mathrm{kb}}, G_{k-1:0}\right) \tag{15}$$

Similarly, we get

$$G_{i:ka} = G_{i:j} \vee Q_{i:j}G_{j-1:ka},$$
$$G_{i:kb} = G_{i:j} \vee Q_{i:j}G_{j-1:kb},$$
$$G_{i:ka} = G_{i:ja}G_{i:jb} \vee (G_{i:ja}G_{i:jb})G_{j-1:ka},$$
$$G_{i:kb} = G_{i:ja}G_{i:jb} \vee (G_{i:ja}G_{i:jb})\, G_{j-1:kb},$$

or

$$G_{i:ka} = M\left(G_{i:ja}G_{i:jb}, G_{j-1:ka}\right) \tag{16}$$
$$G_{i:kb} = M\left(G_{i:ja}G_{i:jb}, G_{j-1:kb}\right) \tag{17}$$

Also,

$$G_{k-1:0} = M\left(G_{k:ja}, G_{k:jb}, G_{j-1:0}\right) \tag{18}$$

Concluding, the root node of the carry computation tree is implemented according to relation (15), while the group generate signals at level L, are derived by the group generate signals at level $L-1$ using equations (16), (17), and (18).

The group generate signals are generated recursively until the terms $G_{i:(i-1)a}$,$G_{i:(i-1)b}$, $G_{1:0}$, are formed. These signals are generated by using the following equations:

$$G_{i:(i-1)a} = g_i \vee p_i a_{i-1}$$
$$G_{i:(i-1)b} = g_i \vee p_i\ b_{i-1}$$
$$G_{1:0} = g_1 \vee p_1\ g_0$$

Using Lemma 1 we get that:

$$G_{i:(i-1)a} = M\left(a_i, b_i, a_{i-1}\right) \tag{19}$$
$$G_{i:(i-1)b} = M\left(a_i, b_i, b_{i-1}\right) \tag{20}$$
$$G_{1:0} = M\left(a_1, b_1, g_0\right) = \ M\left(a_1, b_1, M\left(a_0, b_0, 0\right)\right) \tag{21}$$

Application of the proposed technique derives to a full tree implementation of the carry bits, using only MGs.

The root node of each carry bit is implemented using one majority gate, while the partial group generate signal pairs and the group generate signal pairs are computed based to equations (16), (17) and (20), (21), respectively. Furthermore, the partial group generate signal pairs and the group generate signal pairs are implemented using the nodes shown in Fig. 4a, which includes two majority gates. The design of $G_{1:0}$, is shown in Fig. 4b.

Conventional parallel-prefix adder architectures are reviewed and compared in [24]. Among them the Kogge-Stone (KS) adder architecture has minimal depth, high node count (implies more hardware complexity) and fan out of each node equal to 1. The Ladner-Fisher (LF) architectures has also minimal depth, low node count, but they have high fan out. The Brent-Kung (BK) adders shows maximum logic depth (implies longer calculation time) and minimum number of nodes (implies minimum hardware complexity). The Han-Carlson (HC) adders are modified Kogge-Stone architectures with decreased node count and increased by one node logical depth. This modification can also be applied to Ladner-Fisher adders, resulting to modified Ladner-Fisher (MLF) adder architectures with smaller number of nodes, decreased fan out, and increased by one logical depth. The 8-bit carry computation units of these architectures are given in Fig. 5.

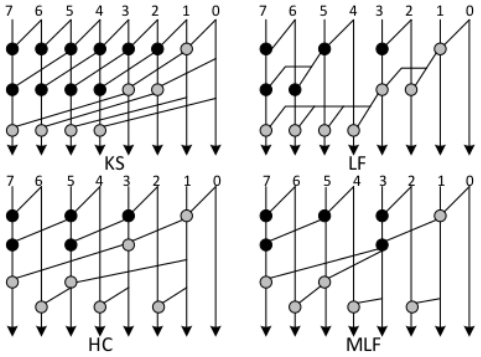


Fig. 5. 8-bit parallel-prefix carry computation unit architectures

According to the proposed majority logic carry computation methodology, all the existing conventional parallel-prefix carry computation architectures can efficiently be implemented using majority logic. As a first step, the carry computation unit of the considered adder is designed according to the targeting conventional carry architecture. Then this architecture is implemented using majority gates according to the proposed carry computation methodology. A significant difference is that the derived majority logic architectures does not require the computation of the g_i and p_i signals [26].

B. Sum bit computation

The i-th sum bit of an n-bit adder is computed as

$$s_i = a_i \oplus b_i \oplus c_{i-1} \tag{22}$$

According to the analysis provided in [27], the following equation holds

$$s_i = M\left(c_i,\ c_{i-1},\ M\left(a_i, b_i, c_{i-1}\right)\right) \tag{23}$$

Using Lemma 4 it derives

$$s_i = M\left(c_i,\ c_{i-1},\ M\left(a_i, b_i, M\left(a_i, b_i, c_{i-1}\right)\right)\right) \tag{24}$$

According to Lemma 1 it derives $c_i = M\left(a_i,\ b_i,\ c_{i-1}\right)$. It holds that

$$s_i = M\left(c_i,\ c_{i-1},\ M\left(a_i, b_i, c_i\right)\right) \tag{25}$$

Equation (25) results to the efficient majority logic implementation of the s_i bit using two majority gates and one inverter shown in Fig. 6a.

An 8-bit majority logic KS parallel-prefix adder is given in Fig. 7.

C. Efficient majority logic parallel-prefix adder architectures

Our design is based on a new sum bit computation, which results to simplified parallel-prefix carry computation unit architectures and therefore to efficient majority logic adder architectures.

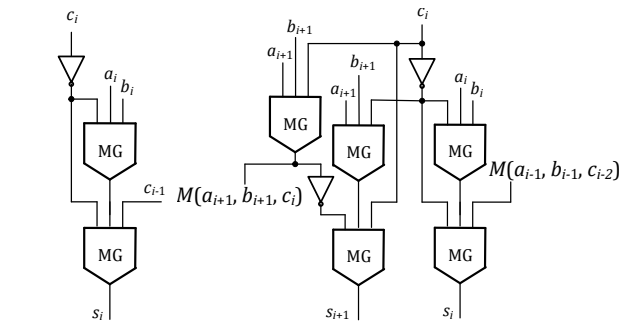


Fig. 6. Majority logic implementation of (a) bit s_i (b) $s_{i+1},\ s_i$

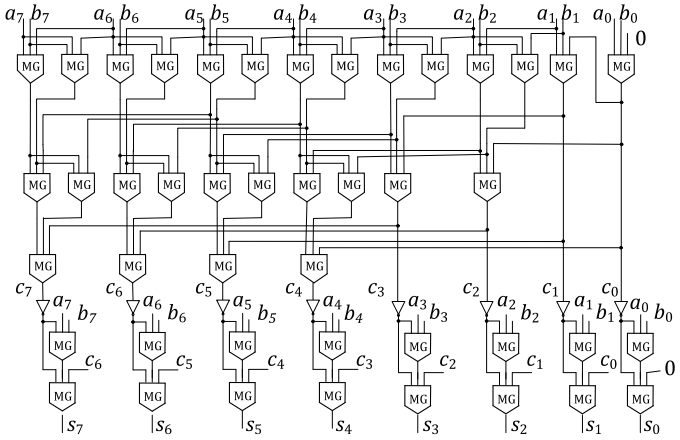


Fig. 7. Majority logic Kogge-Stone 8-bit carry computation unit

D. New sum bit computation

We have that $c_{i-1} = M\left(a_{i-1},\ b_{i-1},\ c_{i-2}\right)$. Then from equation (23) we get

$$s_i = M\left(c_i,\ M(a_{i-1},\ b_{i-1},\ c_{i-2}),\ M\left(a_i, b_i, c_i\right)\right) \quad (26)$$

From equation (23) we get that bit s_{i+1} is computed according to the following:

$$s_{i+1} = M\left(c_{i+1},\ c_i,\ M\left(a_{i+1}, b_{i+1}, c_i\right)\right) \quad (27)$$

Since $c_{i+1} = M\left(a_{i+1},\ b_{i+1},\ c_i\right)$ the following holds

$$s_{i+1} = M\left(M\left(a_{i+1},\ b_{i+1},\ c_i\right), c_i,\ M\left(a_{i+1}, b_{i+1}, c_i\right)\right) \quad (28)$$

According to equations (26) and (28) bits $s_{i+1},\ s_i$ are computed only from carries c_i, c_{i-2}. Therefore, carry c_{i-1} does not need to be computed. The majority logic implementation of s_{i+1}, s_i bits according to equations (26) and (28) is shown in Fig. 6b.

Concluding, according to the proposed methodology only the carries $c_{n-1}\left(= c_{\text{out}}\right), c_{n-3}, c_{n-5}, c_3, c_1, c_0$ of an n-bit adder is computed resulting to simplified parallel-prefix adder architectures.

We exemplify our methodology by designing an 8-bit majority logic simplified Kogge Stone parallel-prefix adder.

Example. Let the 8-bit Kogge-Stone (KS) adder architecture shown in Fig. 5. According to the proposed methodology only carries c_7, c_5, c_3, c_1, c_0 need to be computed. The simplified Kogge-Stone (SKS) parallel-prefix architecture of the carry computation unit is shown in Fig. 8.

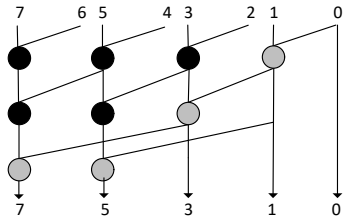


Fig. 8. 8-bit SKS carry computation unit

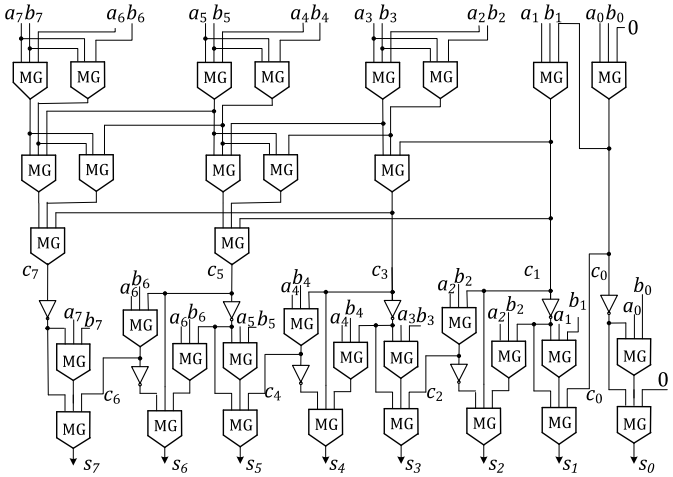


Fig. 9. Proposed 8-bit majority logic SKS parallel-prefix adder

The carry computation unit of Fig. 8 is implemented using majority logic according to the proposed methodology. We have that $c_0 = g_0 = M\left(a_0, b_0, 0\right)$. Using relation (21) we get that $c_1 = G_{1:0} = g_1 \vee p_1 g_0 = M\left(a_1, b_1, g_0\right)$. According to relation (15) we get that $c_3 = G_{3:0} = M\left(G_{3:2a}, G_{3:2b}, G_{1:0}\right)$ where $G_{3:2a} = M\left(a_3, b_3, a_2\right)$, $G_{3:2b} = M(a_3, b_3, b_2)$. In the same way

$$c_5 = G_{5:0} = M\left(G_{5:2a}, G_{5:2b}, G_{1:0}\right),$$

where

$$G_{5:2a} = M\left(G_{5:4a}, G_{5:4b}, G_{3:2a}\right)$$
$$G_{5:2b} = M\left(G_{5:4a}, G_{5:4b}, G_{3:2b}\right),$$

Similarly,

$$c_7 = G_{7:0} = M\left(G_{7:4a}, G_{7:4b}, G_{3:0}\right),$$

where

$$G_{7:4a} = M\left(G_{7:6a}, G_{7:6b}, G_{5:4a}\right)$$
$$G_{7:4b} = M\left(G_{7:6a}, G_{7:6b}, G_{5:4b}\right).$$

Bits s_0 and s_7 of the proposed adder are implemented according to equation (25), while the bit pairs are implemented with 5 majority gates and 2 inverters according to equation (28). The proposed majority logic design of the 8-bit SKS parallel-prefix adder is depicted in Fig. 9.

It is noted that the proposed methodology can be efficiently applied to all the existing conventional parallel-prefix carry computation architectures.

IV. Complexity analysis and comparisons

Majority logic parallel-prefix adder designs are published previously in [18]–[25]. Among them, the adders in [24] are the most efficient. Therefore, the proposed new majority logic parallel-prefix adders are compared only against the designs

in [24]. The proposed designs have been described in Verilog and verified by using extensive simulation. The number of the majority gates required for the implementation of all the considered design is the sum of the number of the majority gates of the parallel-prefix carry computation unit and that of the sum unit. Besides, the delay complexity required is the sum of the delays required for generation of all carries and sums.

The complexity of the carry computation unit depends on the selected parallel-prefix adder architecture. In particular, the Kogge-Stone (KS) parallel-prefix unit is implemented using $n\log_2 n - n + 1$ nodes [28]. Each node is implemented with two majority gates, except the $n-1$ root nodes, which are implemented using only one. One more majority gate is required for the computation of signal g_0. Then the total complexity of the KS carry computation unit is $2\left(n\log_2 n - n + 1\right) - (n-1) + 1 = 2n\log_2 n - 3n + 4$.

The LF parallel-prefix units are implemented using $\frac{1}{2}n\log_2 n$ nodes [28]. In the same way we get that they are implemented with $n\log_2\ n - (n-1) + 1 = n\log_2 n - n + 2$ majority gates. Each sum bit of all the majority logic designs in [24] is implemented using two majority gates and one inverter. That is, the complexity of the sum unit of the adders in [24] is $2n$ majority gates and n inverters.

Summarizing, the Kogge-Stone adders are implemented with $A_{\text{KS}} = 2n\log_2 n - 3n + 4$ majority gates and n inverters, while the Ladner-Fisher adders are implemented with $A_{\text{LF}} = n\log_2 n + n + 2$ majority gates and n inverters.

The proposed SKS parallel-prefix units are implemented using $\frac{1}{2}n\log_2\ n-(\frac{n}{2}-1)$ nodes. Therefore, the total complexity of the SKS carry computation unit is $2\left(\frac{1}{2}n\log_2\ n - \frac{n}{2} + 1\right) - \frac{n}{2} + 1 \ = n\log_2\ n - \frac{3n}{2} + 3$ majority gates. Also, the proposed SLF parallel-prefix units are implemented using $\frac{1}{4}n\log_2 n + \frac{n}{4}$ nodes. Therefore, they are implemented with $\frac{1}{2}n\log_2 n + \frac{n}{2} - \frac{n}{2} + 1 = \frac{1}{2}n\log_2 n + 1$ majority gates.

The s_{n-1} and the s_0 bits of both the proposed SKS and SLF designs are implemented with 2 majority gates and one inverter. The rest of the sum bits are implemented in pairs, where each pair is implemented with 5 majority gates and 2 inverters. Concluding the total complexity of the proposed sum unit is $4 + 5\frac{n-2}{2} = \frac{5n}{2} - 1$ majority gates and n inverters.

That is, the total complexity of the proposed SKS adders is $A_{\text{SKS}} = \frac{1}{2}n\log_2 n + 1$ majority gates and n inverters, while that of the proposed SLF adders is $A_{\text{SLF}} = \frac{1}{2}n\log_2\ n+1$ and n inverters. Since all the considered designs have the same number of inverters, they are not included in our comparisons.

Subsequently the delay complexity is computed. The delay for the carry computation unit in parallel-prefix adders is the sum of the total prefix adder stages and one majority gate delay for computing carry c_0 as shown in Fig. 5. The delay of the sum unit is the delay of two majority gates and one inverter.

More specifically the number of stages for the Kogge-Stone is $\log_2 n$, therefore the corresponding delay complexity is $\log_2 n + 3$. The same holds for the Ladner-Fischer adders $\log_2 n + 3$. Similarly, the delay for the proposed SKS adders and the proposed SLF adders $\log_2 n + 3$. Summarizing, the delay complexity for all the designs is the same and hence it is omitted from the comparisons.

TABLE I
NUMBER OF MG FOR VARIOUS SIZES OF KS AND THE PROPOSED SKS ADDERS.

n	KS [24] A_{KS}	**Proposed (SKS)** A_{SKS}	Saving (%)
8	44	34	22.7
16	116	82	29.3
32	292	194	33.6
64	708	450	36.4
128	1668	1026	38.5

TABLE II
NUMBER OF MG FOR VARIOUS SIZES OF LF AND THE PROPOSED SLF ADDERS.

n	LF [24] A_{LF}	**Proposed (SLF)** A_{SLF}	Saving (%)
8	34	32	5.9
16	82	72	12.2
32	194	160	17.5
64	450	352	21.8
128	1026	768	25.1

Table I compares the complexity of the conventional majority logic KS designs [24] and the complexity of the proposed simplified Kogge-Stone adder (SKS) adder design. In particular, the proposed design exhibits from 22.7% up to 38.5% savings in terms of the number of majority gates.

Table II compares the number of majority gates used in the conventional majority logic LF designs, and the number of majority gates of the proposed simplified Ladner-Fisher adder (SLF) adder design. It is shown that the proposed circuit design depicts up to 25.1% reduction. Furthermore, the proposed design depicts smaller fanout and hence significant gains can be achieved for the implementations, where fanout is important.

Compared against the majority logic Han-Carlson adders the proposed simplified Kogge-Stone adders have the same hardware complexity, but they have the advantage that they are implemented with one majority gate level less. The same holds for the comparison of the proposed simplified Ladner-Fisher adders with the modified Ladner-Fisher adders.

V. CONCLUSIONS

In this work a new majority logic carry computation methodology is presented. Using this methodology all the existing parallel-prefix adder architectures can be implemented using majority logic. New efficient majority logic parallel-prefix adder architectures are also presented. The new majority logic parallel-prefix adders compared against the existing ones, are implemented using smaller number of majority gates, and smaller fanout or with decreased by one level of majority gates. The proposed designs can be applied to the efficient implementation of fast arithmetic units using emerging beyond CMOS nanotechnologies.

REFERENCES

[1] R. H. Dennard, J. Cai, and A. Kumar, "A perspective on today's scaling challenges and possible future directions," *Solid-State Electronics*, vol. 51, no. 4, pp. 518–525, Apr. 2007. [Online]. Available: http://dx.doi.org/10.1016/J.SSE.2007.02.004

[2] C. S. Lent, P. D. Tougaw, W. Porod, and G. H. Bernstein, "Quantum cellular automata," *Nanotechnology*, vol. 4, no. 1, pp. 49–57, Jan. 1993. [Online]. Available: http://dx.doi.org/10.1088/0957-4484/4/1/004

[3] P. D. Tougaw and C. S. Lent, "Logical devices implemented using quantum cellular automata," *Journal of Applied Physics*, vol. 75, no. 3, pp. 1818–1825, Feb. 1994. [Online]. Available: http://dx.doi.org/10.1063/1.356375

[4] G. S. Rose, J. Rajendran, H. Manem, R. Karri, and R. E. Pino, "Leveraging Memristive Systems in the Construction of Digital Logic Circuits," vol. 100, no. 6. Institute of Electrical and Electronics Engineers (IEEE), Jun. 2012, pp. 2033–2049. [Online]. Available: http://dx.doi.org/10.1109/JPROC.2011.2167489

[5] J. Reuben, "Rediscovering majority logic in the post-cmos era: A perspective from in-memory computing," *Journal of Low Power Electronics and Applications*, vol. 10, no. 3, 2020. [Online]. Available: https://www.mdpi.com/2079-9268/10/3/28

[6] V. Jamshidi, "A VLSI Majority-Logic Device Based on Spin Transfer Torque Mechanism for Brain-Inspired Computing Architecture," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 8, pp. 1858–1866, Aug. 2020. [Online]. Available: http://dx.doi.org/10.1109/TVLSI.2020.2997369

[7] 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. 44–60, Sep. 2015. [Online]. Available: http://dx.doi.org/10.1109/MSPEC.2015.7226612

[8] 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 Letters*, vol. 13, no. 6, pp. 2980–2988, May 2013. [Online]. Available: http://dx.doi.org/10.1021/nl4016107

[9] A. K. George and H. Singh, "Three-input majority gate using spatially localised DNA hairpins," *Micro & Nano Letters*, vol. 12, no. 3, pp. 143–146, 2017. [Online]. Available: http://dx.doi.org/10.1049/mnl.2016.0535

[10] S. Dutta, O. Zografos, S. Gurunarayanan, I. Radu, B. Soree, F. Catthoor, and A. Naeemi, "Proposal for nanoscale cascaded plasmonic majority gates for non-Boolean computation," *Scientific Reports*, vol. 7, no. 1, Dec. 2017. [Online]. Available: http://dx.doi.org/10.1038/s41598-017-17954-2

[11] H. Iwamura, M. Akazawa, and Y. Amemiya, "Single-Electron Majority Logic Circuits," *IEICE technical report. Electron devices*, vol. 97, pp. 39–44, 1997. [Online]. Available: https://api.semanticscholar.org/CorpusID:86630494

[12] T. Oya, T. Asai, T. Fukui, and Y. Amemiya, "A majority-logic device using an irreversible single-electron box," *IEEE Transactions On Nanotechnology*, vol. 2, no. 1, pp. 15–22, Mar. 2003. [Online]. Available: http://dx.doi.org/10.1109/TNANO.2003.808507

[13] H. Fahmy and R. Kiehl, "Complete logic family using tunneling-phase-logic devices," in *ICM'99. Proceedings. Eleventh International Conference on Microelectronics (IEEE Cat. No.99EX388)*. IEEE, 2000. [Online]. Available: http://dx.doi.org/10.1109/ICM.2000.884828

[14] S. Lee, S. Choa, S. Lee, and H. Shin, "Magneto-Logic Device Based on a Single-Layer Magnetic Tunnel Junction," *IEEE Transactions on Electron Devices*, vol. 54, no. 8, pp. 2040–2044, Aug. 2007. [Online]. Available: http://dx.doi.org/10.1109/TED.2007.900683

[15] B. Parhami, D. Abedi, and G. Jaberipur, "Majority-Logic, its applications, and atomic-scale embodiments," *Computers & Electrical Engineering*, vol. 83, p. 106562, May 2020. [Online]. Available: http://dx.doi.org/10.1016/j.compeleceng.2020.106562

[16] T. Zhang, H. Jiang, W. Liu, F. Lombardi, L. Liu, S.-B. Ko, and J. Han, "A Survey of Majority Logic Designs in Emerging Nanotechnologies for Computing," *IEEE Transactions on Nanotechnology*, vol. 22, pp. 732–739, 2023. [Online]. Available: http://dx.doi.org/10.1109/TNANO.2023.3326199

[17] I. Koren, *Computer Arithmetic Algorithms*. A K Peters/CRC Press; 2nd edition, 2001.

[18] V. Pudi and K. Sridharan, "Low Complexity Design of Ripple Carry and Brent-Kung Adders in QCA," *IEEE Transactions on Nanotechnology*, vol. 11, no. 1, pp. 105–119, Jan. 2012. [Online]. Available: http://dx.doi.org/10.1109/TNANO.2011.2158006

[19] ——, "New Decomposition Theorems on Majority Logic for Low-Delay Adder Designs in Quantum Dot Cellular Automata," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 59, no. 10, pp. 678–682, Oct. 2012. [Online]. Available: http://dx.doi.org/10.1109/TCSII.2012.2213356

[20] S. Perri, P. Corsonello, and G. Cocorullo, "Area-Delay Efficient Binary Adders in QCA," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 5, pp. 1174–1179, May 2014. [Online]. Available: http://dx.doi.org/10.1109/TVLSI.2013.2261831

[21] A. Roohi, R. Zand, and R. F. DeMara, "A Tunable Majority Gate-Based Full Adder Using Current-Induced Domain Wall Nanomagnets," *IEEE Transactions on Magnetics*, vol. 52, no. 8, pp. 1–7, Aug. 2016. [Online]. Available: http://dx.doi.org/10.1109/TMAG.2016.2540600

[22] G. Jaberipur, B. Parhami, and D. Abedi, "A Formulation of Fast Carry Chains Suitable for Efficient Implementation with Majority Elements," in *2016 IEEE 23nd Symposium on Computer Arithmetic (ARITH)*. IEEE, Jul. 2016. [Online]. Available: http://dx.doi.org/10.1109/ARITH.2016.14

[23] M. SangSefidi, D. Abedi, and G. Jaberipur, "Radix-8 full adder in QCA with single clock-zone carry propagation delay," *Microprocessors and Microsystems*, vol. 51, pp. 176–184, Jun. 2017. [Online]. Available: http://dx.doi.org/10.1016/j.micpro.2017.04.005

[24] V. Pudi, K. Sridharan, and F. Lombardi, "Majority Logic Formulations for Parallel Adder Designs at Reduced Delay and Circuit Complexity," *IEEE Transactions on Computers*, vol. 66, no. 10, pp. 1824–1830, Oct. 2017. [Online]. Available: http://dx.doi.org/10.1109/TC.2017.2696524

[25] G. Jaberipur, B. Parhami, and D. Abedi, "Adapting Computer Arithmetic Structures to Sustainable Supercomputing in Low-Power, Majority-Logic Nanotechnologies," *IEEE Transactions on Sustainable Computing*, vol. 3, no. 4, pp. 262–273, Oct. 2018. [Online]. Available: http://dx.doi.org/10.1109/TSUSC.2018.2811181

[26] C. Efstathiou and P. Kitsos, "Efficient majority logic magnitude comparator design," *Microprocessors and Microsystems*, vol. 82, p. 103832, Apr. 2021. [Online]. Available: http://dx.doi.org/10.1016/j.micpro.2021.103832

[27] W. Wang, K. Walus, and G. Jullien, "Quantum-dot cellular automata adders," in *2003 Third IEEE Conference on Nanotechnology, 2003. IEEE-NANO 2003.*, ser. NANO-03. IEEE. [Online]. Available: http://dx.doi.org/10.1109/NANO.2003.1231818

[28] R. Zimmermann, *Binary adder architectures for cell-based VLSI and their synthesis*. ETH Zürich, 1997.