A Formulation of Fast Carry Chains Suitable for Efficient Implementation with Majority Elements

Ghassem Jaberipur[1, 2], Behrooz Parhami[3], and Dariush Abedi[1]
[1]Computer Science and Engineering Dept., Shahid Beheshti Univ., Tehran 19839-63113, Iran
[2]School of Computer Sciences, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran
[3]ECE Dept., Univ. of California, Santa Barbara, CA 93106-9560, USA
e-mails: jaberipur@sbu.ac.ir, parhami@ece.ucsb.edu, d_abedi@sbu.ac.ir

***Abstract*—Carry computation is a most important notion in computer arithmetic, because it dictates the speed of addition, which is in turn vital to high-speed computation, both as a directly used primitive and as a building block for synthesizing other operations. The theory of fast addition is well-established, but from time to time, changes in technology necessitate a reassessment of strategies for carry network implementation, even though the logical functions to be realized remain the same. We study the implications of the availability of simple, fast, and power-efficient majority gates (in technologies such as quantum-dot cellular automata, single-electron tunneling, tunneling phase logic, magnetic tunnel junction, and nanoscale bar magnets) to the design of carry networks, offering a reformulation of the carry recurrence that allows for building carry networks exclusively out of fully utilized majority elements. We compare our novel implementations based on 3-input majority elements to prior proposals based on these elements, demonstrating advantages in both speed and circuit complexity.**

Keywords- Carry network; Carry recurrence; Fast adder; Lookahead; Majority logic; Parallel-prefix adder

I. INTRODUCTION

Notions of fast addition and the associated circuitry for carry computation were developed very early in the history of digital arithmetic [1]. Charles Babbage fiddled with the idea of reducing the propagation penalty of ripple-carry addition by devising mechanisms for carry anticipation [2]. More than five decades after its modern inception in electronic implementation of binary arithmetic [3], the notion of carry computation via look-ahead techniques is still being refined, both theoretically (in terms of parallel-prefix formulations) and with regard to 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 the most notable examples). Fast addition can be performed by means of 2-way or multi-way combining of carry generation and propagation information from blocks of the (not necessarily binary) operands. 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. Further expanding the design space is the fact that the carry network can be designed in many different ways.

An interesting taxonomy for parallel-prefix carry networks, which only partially covers the full design space, was proposed by Harris [4], where alternative designs for a k-bit adder entail choices of values for trade-off parameters f, t, and l, where gate fan-out is $2^f + 1$, number of wire tracks is 2^t, and number of circuit-block levels is $\lceil \log k \rceil + l$.

All aforementioned designs use AND and OR gates for implementing the carry operator. To further expand on the available options, use of multiplexers and other kinds of building blocks has been investigated [5]. Essentially, new technologies bring with them more/less optimal realizations of certain building blocks, thus necessitating a fresh look at carry networks and other circuits of interest to see if the capabilities and limitations of the new technology can be accommodated to improve the design. A blind mapping of existing designs to new technologies often leads to suboptimal designs, even if the mapped implementation does offer improvements.

Quantum-dot cellular automata (QCA) is a new technology with broad computational potential [6-7] that requires fundamental reassessment of how we perform arithmetic. There are other technologies with properties and potential similar to QCA, although we have not examined them in detail yet. Single-electron integrated circuits [8], computational use of nanomagnets [9], and molecular computing [10-11] are some examples. Full-adder blocks and simple adders have been designed with QCA [12-16], but there is no indication whether these are optimal designs or just feasible ones produced either manually or by means of design tools. On the other hand circuit design methodologies, based on majority gates, have been studied for developing QCA circuits with at most three input variables (e.g., [17-18]).

All these technologies, and a few others reviewed briefly in Section III and in the Appendix, allow efficient realization of majority gates, so a natural question, tackled in this paper, is whether one can 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 M(0, x, y) and OR as M(1, x, y), where M denotes the 3-way majority function (see also the end of Section II).

Majority elements, as well as techniques for synthesizing logic functions using such gates, have a long history. Majority with n inputs is a special case of a threshold function, with unit-weight inputs and a threshold value of $\lceil (n + 1)/2 \rceil$. During its 70-year history, threshold logic has been revisited from time to time in connection with a multitude of technologies [19]. Early interest revolved around neural networks and neuronlike computational elements [20]. Subsequent designs were realized in several technologies and entailed capacitance- and inductance-based solutions, among others [21].

1063-6889/16 $31.00 © 2016 IEEE
DOI 10.1109/ARITH.2016.14

II. PARALLEL CARRY GENERATION

Consider the n-bit addition with $c_0 = c_{in}$, viz. $a_{n-1}...a_1a_0 + b_{n-1}...b_1b_0 + c_{in} = c_{out}s_{n-1}...s_1s_0$. Eqn. set 1 defines the required sum (s_i) and carry (c_{i+1}) bit-operations, where $\oplus$ and $\vee$ denote the binary exclusive and inclusive OR, respectively.

$$s_i = a_i \oplus b_i \oplus c_i\,,\; c_{i+1} = a_ib_i \vee (a_i \vee b_i)c_i \quad (0 \le i \le n-1) \qquad (1)$$

Because c_{i+1} is either produced in position i by the generate signal $g_i = a_ib_i$, or is the same as c_i , when it propagates over position i (if the propagate signal $p_i = a_i \vee b_i$ permits), it can also be expressed as in Eqn. 2.

$$c_{i+1} = g_i \vee p_ic_i \qquad (2)$$

Likewise, c_{i+2} can be expressed in terms of c_{i+1} , and c_i as in Eqn. 3, where $G_{i+1:i}(P_{i+1:i})$ indicates the generation (propagation) of carry within (over) positions i and $i+1$.

$$\begin{aligned} c_{i+2} &= g_{i+1} \vee p_{i+1}c_{i+1} = g_{i+1} \vee g_ip_{i+1} \vee p_{i+1}p_ic_i \\ &= G_{i+1:i} \vee P_{i+1:i}c_i \end{aligned} \qquad (3)$$

Realization of a generalized version of the latter equation has led to fast parallel prefix carry generation networks (CGN), where the key operation is expressed by Eqn. 4. For example, Kogge-Stone (KS) and Ladner-Fischer (LF) parallel-prefix networks [1] (refer to Figs. 14 and 15 in Section IV), use 2-bit wide instances of Eqn. 4.

$$(G_{i:j}, P_{i:j}) = (G_{i:k} \vee P_{i:k}G_{k-1:j}, P_{i:k}P_{k-1:j}) \qquad (4)$$

A. Carry circuit as a majority gate

Because $c_{i+1} = 1$ iff at least 2 of the 3 bit-variables a_i , b_i , and c_i are 1, the carry operation within Eqn. set 1 can also be expressed as in Eqn. 5.

$$c_{i+1} = \mathrm{M}(a_i, b_i, c_i) \qquad (5)$$

Likewise, c_{i+2} can be expressed as in Eqn. 6, with two M gates in the critical delay path (CDP) c_i to c_{i+2} . Perri *et al.* [22] have shown that the number of M gates in the c_i -to- c_{i+2} CDP can be reduced to one, as is evident by the easily-proven Eqn. set 7. Note that this speed up is at the cost of three extra M gates (i.e., a total of five), two of which are partially utilized. Fig. 1 depicts the required logic, where each solid (dashed) building block represents a fully (partially) utilized M gate.

$$c_{i+2} = \mathrm{M}(a_{i+1}, b_{i+1}, \mathrm{M}(a_i, b_i, c_i)) \qquad (6)$$

$$\begin{aligned} p_i &= \mathrm{M}(a_i, b_i, 1) \\ g_i &= \mathrm{M}(a_i, b_i, 0) \end{aligned}$$

$$c_{i+2} = \mathrm{M}(\mathrm{M}(a_{i+1}, b_{i+1}, p_i), \mathrm{M}(a_{i+1}, b_{i+1}, g_i), c_i) \qquad (7)$$

Besides the majority-based carry equation, the sum bit can be also expressed via majority function as in Eqn. 8, where overlined expressions denote logical inversion, and c_{i+2} is derived via Eqn. 5.

$$s_i = \mathrm{M}(\overline{c_{i+1}}, \mathrm{M}(a_i, b_i, \overline{c_i}), c_i) \qquad (8)$$

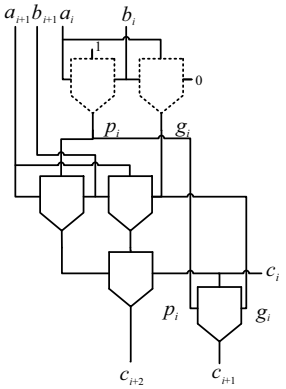

Figure 1. Two-bit M-based carry generation [22]

Other existing M-based parallel CGN are due to references [14-15]. The pertinent architecture for $n = 8$ [15], is depicted in Fig. 2 (with conventional (G, P) notation and complete carry generation), where 29 M gates are used in 5 levels. In particular, c_7 and c_8 , required (based on Eqn. 8) for $s_7 = \mathrm{M}(\overline{c_8}, \mathrm{M}(a_7, b_7, \overline{c_7}), c_7)$ are delivered in level 5. That is the CDP for the most-significant bit of sum passes through 7 M gates.

In both of the latter works, the conventional propagate and generate signals are produced via partially utilized M gates (i.e., with one constant input). In fact, a straightforward mapping of Fig. 7 to an M-based CGN, would replace each AND and OR with its equivalent M element, which leads to 7 M gates in the corresponding CDP and a total of 73 M boxes in the circuit. Therefore, the advantages of design of Fig. 2 (i.e., reducing the latter two figures of merit to 5 and 29, respectively) are considerable. In Section IV, we propose a $\lceil \log n \rceil$ -level (e.g., 3 levels for $n = 8$) M-based parallel CGN composed solely of fully utilized M building blocks.

III. EMERGING M-BASED TECHNOLOGIES

As noted in Section I, equally weighted majority function is a special case of threshold logic gates with weighted inputs. The CMOS realization of a majority gate with three Boolean inputs a , b and c yields $\mathrm{M}(a,b,c) = (a \vee b)c \vee ab$, b .

Clearly, 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, 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 $\mathrm{M}(a,b,c) = \lfloor (a+b+c+1)/3 \rfloor$, and it can be viewed as the median function. The following four properties of 3-input majority/median function, when used as axioms, define a median algebra:

$$\mathrm{M}(a,b,b) = b$$

$$\mathrm{M}(a,b,c) = \mathrm{M}(a,c,b)\,; \qquad \mathrm{M}(a,b,c) = \mathrm{M}(c,a,b)$$

$$\mathrm{M}(\mathrm{M}(a,x,b),x,c) = \mathrm{M}(a,x,\mathrm{M}(b,x,c))$$

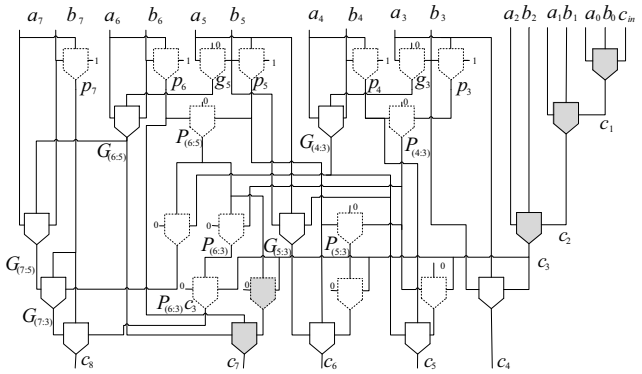

Figure 2. Five-level 8-bit parallel CGN [15]; dashed M gates are partially utilized, shaded ones define the CDP

Figure 3. Three QCA cell configurations

Figure 4. Two QCA M gates

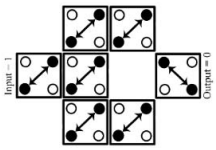

Figure 5. A robust QCA Inverter

A. Quantum-dot cellular automata (QCA)

The basic QCA cell contains four electron place-holders (often called dots), where two injected electrons can assume one of the slash and backslash configurations, as in Fig. 3, which is borrowed from [23]. QCA realization of the M gate is depicted in Fig. 4, and that of an inverter in Fig. 5. In fact, these two gates constitute a complete logic set, since AND and OR functions can be expressed in terms of majority gate. For example, direct QCA realization of a 7-gate full adder (FA) requires 7 partially utilized M gates, while based on Eqns. 5 and 8, it can be realized via 3 fully utilized M gates and 2 inverters.

B. Other majority-friendly technologies

Other emerging technologies that are either based on majority gates or lead to better circuit designs with such gates include Single-electron tunneling (SET), Tunneling phase logic (TPL), Magnetic tunnel junction (MTJ), and Nano-scale bar magnets (NBM). A brief survey is presented in the Appendix.

IV. SCALABLE CGNS WITH FULLY UTILIZED M GATES

Recall Eqn. 6, representing the cost optimized 2- M 2-bit carry expression, and the delay optimized 5- M (only one M within the CDP) Eqn. 7. A compromise solution is described below in Eqn. 9, where 1- M delay 2-bit carry expression is achieved with a cost of 3 M gates.

$$c_{i+2} = \mathrm{M}(\mathrm{M}(a_{i+1}, b_{i+1}, a_i), \mathrm{M}(a_{i+1}, b_{i+1}, b_i), c_i) \tag{9}$$

Justification of this equation follows.

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

Recalling Eqn. 9, let $A_{i+1:i} = \mathrm{M}(a_{i+1}, b_{i+1}, a_i)$ and $B_{i+1:i} = \mathrm{M}(a_{i+1}, b_{i+1}, b_i)$. Then, c_{i+2} can be expressed as in Eqn. 10, which is actually a composite extension of $c_{i+1} = \mathrm{M}(a_i, b_i, c_i)$ and $c_{i+2} = \mathrm{M}(a_{i+1}, b_{i+1}, c_{i+1})$, where $A_{i+1:i}$ and $B_{i+1:i}$ represent the radix-4 input digits $a_{i+1}a_i$ and $b_{i+1}b_i$ in carry generation. This concept can be further extended to higher-radix carry generation, as described and proven in the definition, lemma, and theorems that follow.

$$c_{i+2} = \mathrm{M}(A_{i+1:i}, B_{i+1:i}, c_i) \tag{10}$$

Definition 1 $(A_{j:i}, B_{j:i})$: $A_{j:i} = \mathrm{M}(a_j, b_j, A_{j-1:i}), B_{j:i} = \mathrm{M}(a_j, b_j, B_{j-1:i})$.

Lemma 1 $(A_{j:i}B_{j:i}, A_{j:i} + B_{j:i})$: $A_{j:i}B_{j:i} = g_i + p_j A_{j\text{-}1:i} B_{j\text{-}1:i}$, and $A_{j:i} + B_{j:i} = g_i + p_j(A_{j-1:i} + B_{j-1:i})$ for $j > i$.

Proof: $A_{j:i}B_{j:i} = \mathrm{M}(a_j, b_j, A_{j-1:i})\mathrm{M}(a_j, b_j, B_{j-1:i}) = (g_i + p_j A_{j-1:i})(g_i + p_j B_{j-1:i}) = g_i + p_j A_{j-1:i} B_{j-1:i}$.

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

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

Proof: The proof is by induction on j.

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

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

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

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

$c_{i+j+1} = A_{i+j:i} B_{i+j:i} + (A_{i+j:i} + B_{i+j:i}) c_i = \mathrm{M}(A_{i+j:i}, B_{i+j:i}, c_i)$. ■

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:

$\mathrm{M}(A_{i+3:i}, B_{i+3:i}, c_i) =$

$\mathrm{M}(\mathrm{M}(a_{i+2}, b_{i+2}, A_{i+1:i}), \mathrm{M}(a_{i+2}, b_{i+2}, B_{i+1:i}), c_i)$.

Theorem 2 (Associativity of the M operation): $A_{k+j:i} = \mathrm{M}(A_{k+j:j}, B_{k+j:j}, A_{j-1:i})$, $B_{k+j:i} = \mathrm{M}(A_{k+j:j}, B_{k+j:j}, B_{j-1:i})$.

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

Base $(k = 0)$, is obvious by Definition 1.

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

$A_{k+j:i} = \mathrm{M}(a_{k+j}, b_{k+j}, A_{k-1+j:i}) = g_{k+j} + p_{k+j} A_{k-1+j:i} =$
$g_{k+j} + p_{k+j}\mathrm{M}(A_{k-1+j:j}, B_{k-1+j:j}, A_{j-1:i}) = g_{k+j} + g_{k+j} A_{j-1:i}$
$+ p_{k+j}(A_{k-1+j:j} B_{k-1+j:j} + (A_{k-1+j:j} + B_{k-1+j:j}) A_{j-1:i})$
$= (g_{k+j} + p_{k+j}(A_{k-1+j:j} + B_{k-1+j:j})) A_{j-1:i}$.

With proper replacements based on Lemma 1, we arrive at

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

For example, consider radix-16, 4-bit operand digits and the following expression for c_{i+4}:

$\mathrm{M}(A_{i+4:i}, B_{i+4:i}, c_i) =$
$\mathrm{M}(\mathrm{M}(A_{i+3:i+2}, B_{i+3:i+2}, A_{i+1:i}), \mathrm{M}(A_{i+3:i+2}, B_{i+3:i+2}, B_{i+1:i}), c_i)$.

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

There are instances of twin majority functions with identical first parameters, and also identical second parameters. See, for example, Eqn. 9, Definition 1, and Theorem 2. Therefore, it seems useful to formally define this concept.

Definition 2 (Twin majority gate, TM): Let (A_l, B_l) and (A_r, B_r) denote arbitrary pairs per Definition 1. The twin majority function, (A, B) , is defined as $A = \mathrm{M}(A_l, B_l, A_r)$ and $B = \mathrm{M}(A_l, B_l, B_r)$. The TM function is given the symbolic representation depicted in Fig. 6. ■

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 $(A_{i+15:i}, B_{i+15i})$ pairs. The required carries c_{i+1} to c_{i+16} can then be generated, for $1 \le j \le 16$, as $c_{i+1} = \mathrm{M}(A_{i+j:i}, B_{i+j:i}, c_i)$.

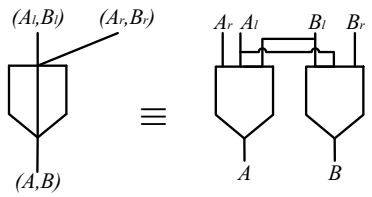

Figure 6. Notation for, and structure of, the TM gate

Parallel-prefix-like CGNs can be readily set up, as discussed in the following subsection dealing with designs based on KS and LF parallel prefix networks.

A. KS-like and LF-like parallel CGNs built of M *gates only*

Figs. 7 and 8 depict n-bit ($n = 8$) KS-like and LF-like parallel-prefix networks, respectively, for possibly nonzero carry-in c_{in}, where square boxes provide p_i and g_i signals, and the black circles (organized in $\lceil \log n \rceil$ levels) represent 2-bit wide instances of Eqn. 4. Note that the required carries for the target 8-bit sum bits (i.e., $s_7 - s_0$) are c_7 to c_{in} that are all available after 3 levels at the latest. Also c_{out} becomes available at the same time as s_7, since the former is ready at the forth level (2 gates after c_7) and the latter one XOR level (i.e., 2 gates) after c_7.

Figs. 9 and 10, corresponding to Figs. 7 and 8, respectively, represent KS-like and LF-like CGNs that are built only from majority gates. The TM (M) gates yield all the required $(A_{k:j}, B_{k:j})$ pairs (carry signals). For example, Fig. 9 contains 11 TM and 8 TM gates, leading to the overall circuit complexity of 30 M gates, all fully utilized.

The corresponding carry equations and the number of M gates on the CDP are shown in Table I, for n = 8 and n = 16. Also, the number of parallel prefix nodes (PPN) is shown for each carry of Fig. 7. Note that the total PPN counts in the KS-like AND/OR case of Table I do not include the 8 and 16 (g, p) generation nodes.

The LF parallel prefix CGN is known for reduced number of PPNs at the cost of high fan-out, which is therefore, rather impractical with current implementation technologies. However, they can be more attractive in some emerging technologies (e.g., QCA), where high fan-out can be accommodated via different cells in different clock zones of the same wire [24-25].

The only M-based LF carry-network design that we have encountered [14] uses 54 M gates for 8-bit CGN and its CDP passes through 6 M levels. However, our proposed TM -based LF-like CGN, with possibly nonzero carry-in, as depicted in Fig. 10, contains only 20 M gates with 4 M levels in the CDP of both c_7 and c_{out}.

More generally, our design scheme leads to $1 + \lceil \log n \rceil$ levels of M gates on the CDP and a total M-count of $n\lceil \log n \rceil + 4$.

Extension of the proposed KS-like design to 16-bit M-based CGN, shown in Fig. 11, helps with the understanding of the network structure and its scalability to larger sizes.

Note that in Fig. 11, the final internal carry c_{15} (i.e., the required carry for obtaining the most significant sum bit) is delivered after 4 M levels. Similarly, the c_{31} of a 32-bit CGN will be ready in 5 M levels, while the design of [15] requires at least 10 M levels for a carry network of this size. Note that the c_{out} production requiring one more M level does not delay the delivery of the most significant sum bit (see Eqn. 8).

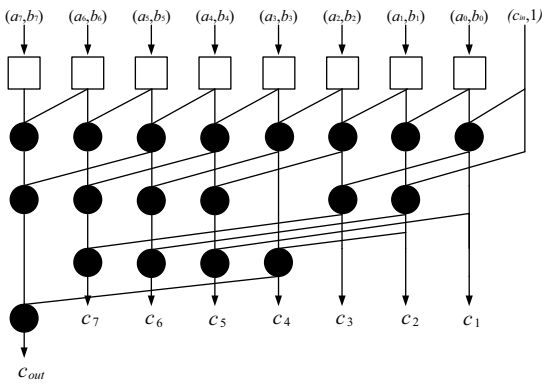

Figure 7. KS-like 8-bit parallel prefix CGN with c_{in}

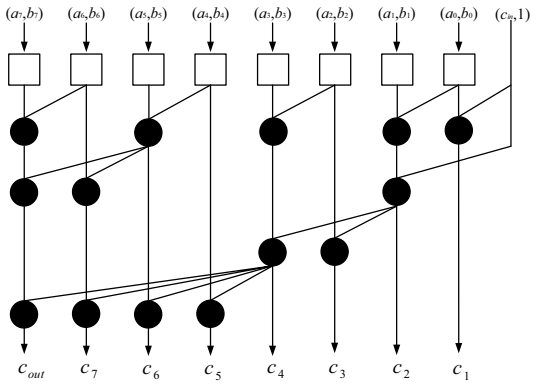

Figure 8. LF-like 8-bit parallel prefix CGN with c_{in}

TABLE I. 8- AND 16-BIT M -BASED AND PPN CARRY EXPRESSIONS

c_i	M -based carries	# of M s		KS-like carries	# of PPNs	
		Sub total	CDP		Sub total	CDP
c_1	$\mathrm{M}(a_0, b_0, c_{in})$	1	1	$(g_0, p_0) \circ (c_{in}, 1)$	1	1
c_2	$\mathrm{M}(A_{1:0}, B_{1:0}, c_{in})$	3	2	$(G_{1:0}, P_{1:0}) \circ (c_{in}, 1)$	2	2
c_3	$\mathrm{M}(A_{2:1}, B_{2:1}, c_1)$	3	2	$(G_{2:1}, P_{2:1}) \circ (c_1, 1)$	2	2
c_4	$\mathrm{M}(A_{3:2}, B_{3:2}, c_2)$	3	3	$(G_{3:2}, P_{3:2}) \circ (c_2, 1)$	2	3
c_5	$\mathrm{M}(A_{4:1}, B_{4:1}, c_1)$	5	3	$(G_{4:1}, P_{4:1}) \circ (c_1, 1)$	3	3
c_6	$\mathrm{M}(A_{5:2}, B_{5:2}, c_2)$	5	3	$(G_{5:2}, P_{5:2}) \circ (c_2, 1)$	3	3
c_7	$\mathrm{M}(A_{6:3}, B_{6:3}, c_3)$	5	3	$(G_{6:3}, P_{6:3}) \circ (c_3, 1)$	3	3
c_8	$\mathrm{M}(A_{7:4}, B_{7:4}, c_4)$	5	4	$(G_{7:4}, P_{7:4}) \circ (c_4, 1)$	3	4
Total		30	4		19	4
c_9	$\mathrm{M}(A_{8:1}, B_{8:1}, c_1)$	7	4	$(G_{8:1}, P_{8:1}) \circ (c_1, 1)$	4	4
c_{10}	$\mathrm{M}(A_{9:2}, B_{9:2}, c_2)$	7	4	$(G_{9:2}, P_{9:2}) \circ (c_2, 1)$	4	4
c_{11}	$\mathrm{M}(A_{10:3}, B_{10:3}, c_3)$	7	4	$(G_{10:3}, P_{10:3}) \circ (c_3, 1)$	4	4
c_{12}	$\mathrm{M}(A_{11:4}, B_{11:4}, c_4)$	7	4	$(G_{11:4}, P_{11:4}) \circ (c_4, 1)$	4	4
c_{13}	$\mathrm{M}(A_{12:5}, B_{12:5}, c_5)$	7	4	$(G_{12:5}, P_{12:5}) \circ (c_5, 1)$	4	4
c_{14}	$\mathrm{M}(A_{13:6}, B_{13:6}, c_6)$	7	4	$(G_{13:6}, P_{13:6}) \circ (c_6, 1)$	4	4
c_{15}	$\mathrm{M}(A_{14:7}, B_{14:7}, c_7)$	7	4	$(G_{14:7}, P_{14:7}) \circ (c_7, 1)$	4	4
c_{16}	$\mathrm{M}(A_{15:8}, B_{15:8}, c_8)$	7	5	$(G_{15:8}, P_{15:8}) \circ (c_8, 1)$	4	5
Total		56	5		32	5
Grand total		86	5		51	5

In general, for an *n*-bit adder, the CDP of KS-like M -based CGN passes through $\lceil \log n \rceil$ levels of M gates, and the carry network has an overall M -gate count of $2n\lceil \log n \rceil - 3n + 6$.

B. QCA Implementation

To demonstrate the feasibility of our proposed designs, we implemented the CGN of Fig. 10 via the QCADesigner [26], with the associated layout depicted in Fig. 12, where color is used to convey the different clock zones. For example, the $c_{in} - c_4$ path goes through the Green, Violet, Blue, and Gray clock zones, where the two CDP majority gates can be seen as a 5-cell blue cross (right after the violet wire) and the 5-cell gray cross.

The latest carry signal (i.e., c_8) is delivered after 6 clock zones, while the similar circuit design of [13] and the custom design of [15] both require 9 clock zones.

The I/O pattern related to Fig. 12, for eight different 8-bit input pairs is seen in Fig. 13. For example, the hexadecimal addition $55 + AA + c_{in}$ (i.e., the binary addition $01010101 + 10101010 + 1$), leads to $c_7 - c_1 = 11111111$, which appear highlighted in Fig. 12.

V. CONCLUSION

Our primary contribution in this paper is a formulation of the carry recurrence directly in terms of majority gates, using 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 proposed designs are applicable to several emerging technologies (including QCA, SET, TPL, MTJ, and NBM) that offer efficient realization of majority gates.

In addition to the formal derivation of the carry recurrence using only fully utilized M gates, we demonstrated fast carry-network implementations by means of LF-like and KS-like parallel-prefix networks that exhibit the same attributes as the original Ladner-Fischer and Kogge-Stone designs.

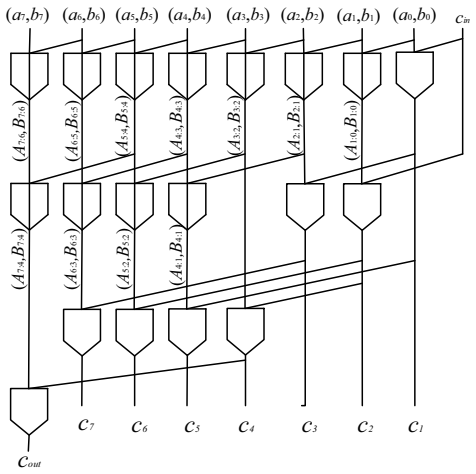

Figure 9. 8-bit KS-like CGN with fully utilized M gates

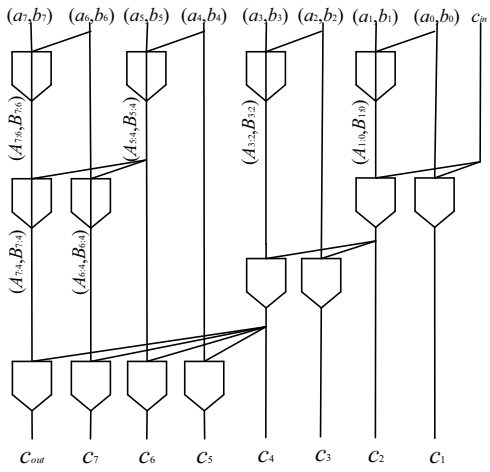

Figure 10. The proposed M-based LF-like parallel CGN

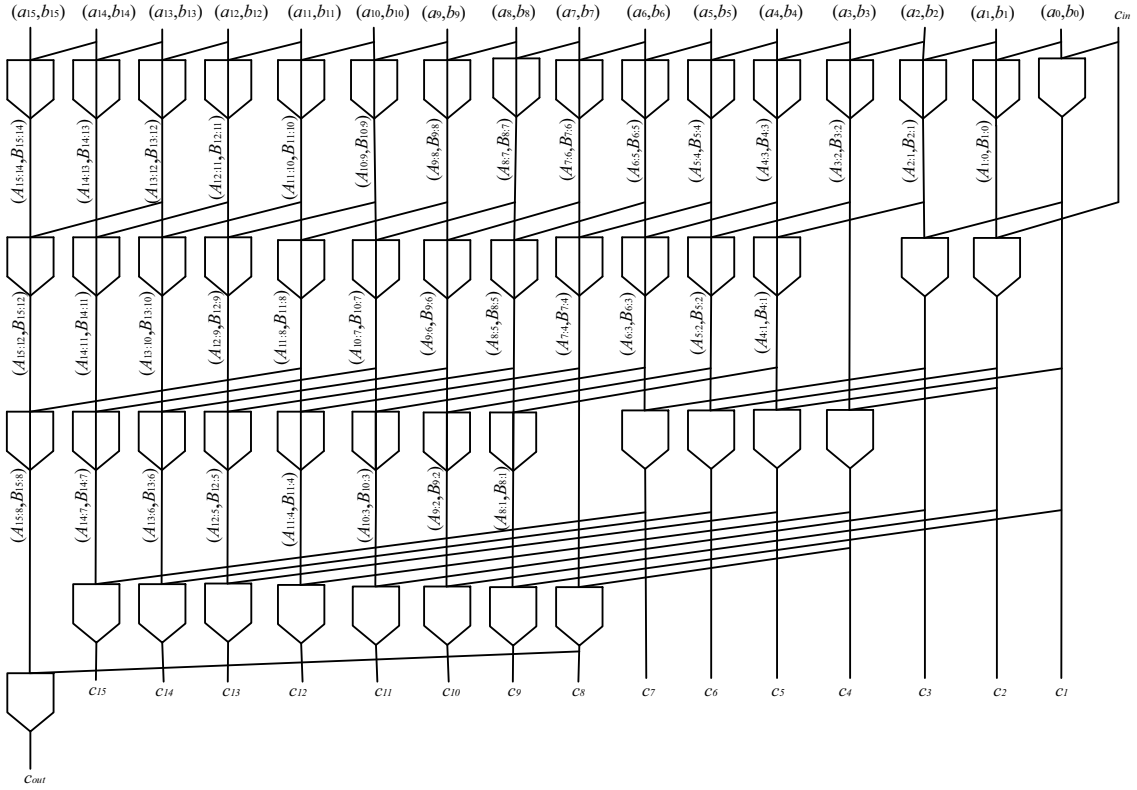

Figure 11. The 16-bit KS-like CGN with fully utilized M gates

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 M -gate inputs, in contrast to partial use that results when emulating AND and OR gates.

This work constitutes a beginning in the efficient use of majority-friendly technologies for realizing fast arithmetic circuits. Not all results derived with QCA will carry over directly to other majority-friendly technologies surveyed in the Appendix 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 designs and to investigate the extent to which the designs carry over to other majority-friendly technologies and associated implementation styles.

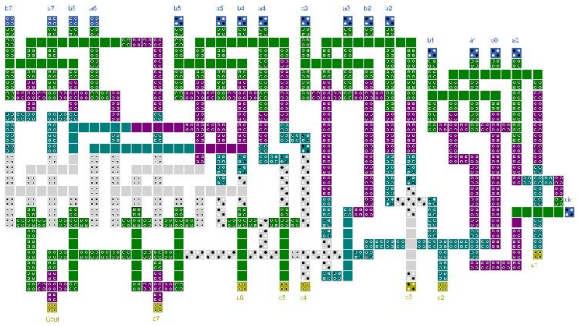

Figure 12. A QCA realization of the LF-like adder of Fig. 10

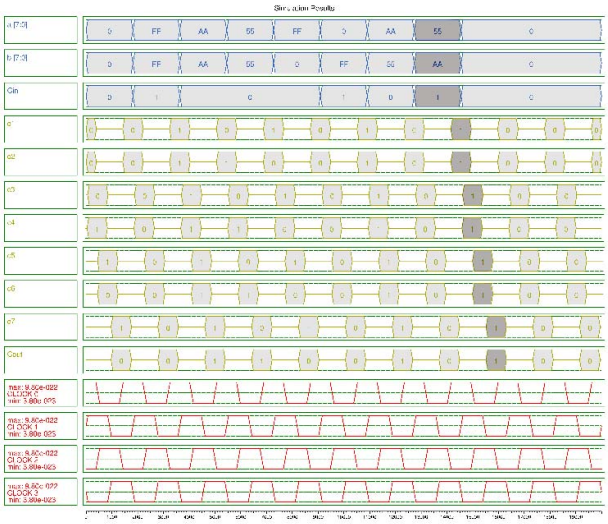

Figure 13. Sample I/O for Fig. 12

An intriguing possibility for future investigation is to consider the incorporation of reliability features [27] using triple-modular redundancy with voting, given that the required voting element is essentially a single M gate.

APPENDIX: NEW M -GATE BASED EMERGING TECHNOLOGIES

A. *Single-electron tunneling (SET)*

Single-carrier electronics offers the ultimate in compactness and energy efficiency. The technology allows for controlled transfer of individual electrons, using the single-electron tunneling phenomenon, hence the name. In order to use this technology for computation, it is necessary to demonstrate feasible logic gates, and this has been done successfully for majority elements [28].

Figure 14 shows the majority circuit along with an inverter that is needed to make the set universal.

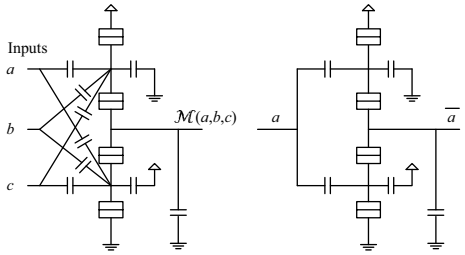

Figure 14. SET circuits for M (left) and inversion (right) [28]

B. *Tunneling phase logic (TPL)*

In this technology, several capacitively coupled inputs feed a load capacitance (Figure 15), which under the right conditions can realize the 3-input minority function [29, 30]. The output of a minority element is the inverse of one input when the other two inputs are opposites of each other, thus the element can also serve as an inverter. The same structure can form the basis for 3-input NAND and NOR gates.

C. *Magnetic tunnel junction (MTJ)*

MTJ is one of the new spintronics technologies which is based on devices with two ferromagnetic thin-film layers, free and fixed, separated by an oxide-tunneling barrier. The fixed layer's magnetization is not easily changeable, whereas the free layer can change magnetization readily to align itself with, or be opposite to, that of the fixed layer, forcing the resistance of the junction to become low or high, which in turn allows the representation of a bit [31-32]. Fig. 16 shows how two of these elements constitute an M gate.

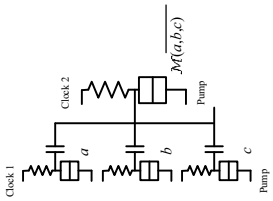

Figure 15. Basic TPL gate [30]

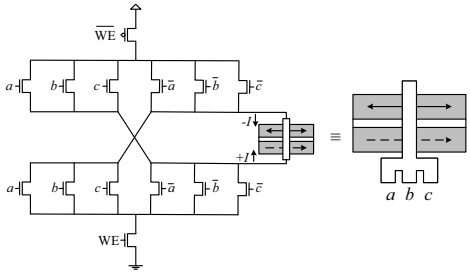

Figure 16. Majority gate in MTJ logic [32]

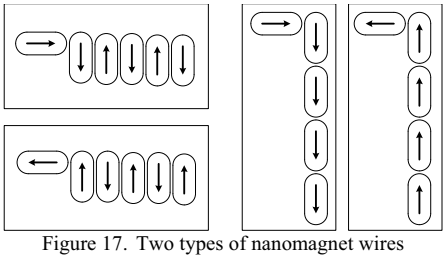

Figure 17. Two types of nanomagnet wires

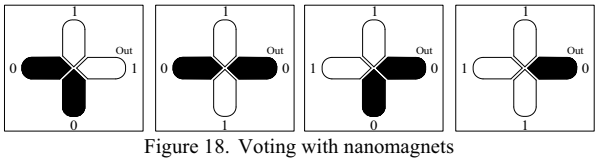

Figure 18. Voting with nanomagnets

D. Nano-scale bar magnets (NBM)

The use of nanomagnets as computational elements [9] was pursued as a way of overcoming some difficulties with QCA. Computing with magnetic elements actually dates back to the early days of digital technology, the new aspect here being the minute size of the magnets. The magnetic directions can be horizontal (on the surface) or vertical (perpendicular to the surface), with the latter option bearing some advantages. The primary benefits of computing with nanomagnets are their extreme energy efficiency and lack of need for latches in 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. Two types of nanomagnet-based wires are shown in Fig. 17 and four voting instances appear in Fig. 18, where the output on the right is actually the complement of the majority value.

E. Other technologies

We refrain from describing biological embodiments of the majority function, which form a basis for neural computation in human and animal brains [11]. 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, so we can expect additional new technologies to support its efficient realization.

ACKNOWLEDGMENT

Jaberipur's research was funded by the School of Computer Sciences, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran, under grant CS1395-2-03.

REFERENCES

[1] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, Oxford Univ. Press, 2nd ed., 2010.

[2] C. Babbage, *Passages from the Life of a Philosopher*, London, 1864 (Reissued by Cambridge Univ. Press, 2011; also available on-line).

[3] G. B. Rosenberger, "Simultaneous Carry Adder," US Patent 2 966 305, December 27, 1960.

[4] D. Harris, "A Taxonomy of Parallel Prefix Networks," *Proc. Asilomar Conf. Signals, Systems, and Computers*, 2003, Vol. 2, pp. 2213-2217.

[5] T. B. Preusser and R. G. Spallek, "Mapping basic prefix computations to fast carry-chain structures," *Int'l Conf. Field Programmable Logic and Applications*, Prague, 2009, pp. 604-608.

[6] C. S. Lent, P. Tougaw, W. Porod, and G. Bernstein, "Quantum Cellular Automata" *Nanotechnology*, Vol. 4, 1993, pp. 49─57.

[7] P. Tougaw and C. Lent, "Logical Devices Implemented Using Quantum
Cellular Automata," *J. Applied Physics*, Vol. 75, No. 3, pp. 1818–1825, February 1994.

[8] H. Iwamura, M. Akazawa, and Y. Amemiya, "Single-Electron Majority Logic Circuits," *IEICE Trans. Electronics*, Vol. E81-C, No. 1, pp. 42-48, January 1998.

[9] 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, September 2015.

[10] L. M. Adleman, "Molecular Computation of Solutions to Combinational Problems," *Science*, Vol. 266, No. 5187, pp. 1021-1024, 1994.

[11] 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, pp. 2980-2988, 2013.

[12] H. Cho and E. Swartzlander, "Adder Designs and Analyses for Quantum-Dot Cellular Automata," *IEEE Trans. Nanotechnology*, Vol. 6, No. 3, pp. 374–383, May 2007.

[13] V. Pudi and K. Sridharan, "Low Complexity Design of Ripple Carry and Brent–Kung Adders in QCA," *IEEE Trans. Nanotechnology*, Vol. 11, No. 1, pp. 105–119, January 2012.

[14] V. Pudi and K. Sridharan, "Efficient Design of a Hybrid Adder in Quantum Dot Cellular Automata," *IEEE Trans. VLSI Systems*, Vol. 19, No. 9, pp. 1535–1548, September 2011.

[15] V. Pudi and K. Sridharan, "New Decomposition Theorems on Majority Logic for Low-Delay Adder Designs in Quantum Dot Cellular Automata," *IEEE Trans. Circuits and Systems II*, Vol. 59, No. 10, pp. 678–682, October 2012.

[16] W. Liu, E. E. Swartzlander Jr., and M. O'Neill, *Design of Semiconductor QCA Systems*, Artech House, 2013.

[17] K. Walus, et al., "Circuit Design Based on Majority Gates for Applications with Quantum-Dot Cellular Automata," *Proc. 38th Asilomar Conf. Signals, Systems and Computers*, pp. 1354-1357, 2004.

[18] R. Zhang, et al., "A Method of Majority Logic Reduction for Quantum Cellular Automata," *IEEE Trans. Nanotechnology*, Vol. 3, pp. 443-450, 2004.

[19] S. Muroga, *Threshold Logic and Its Applications*, Wiley, 1971.

[20] W. S. McCulloch and W. H. Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity," *Bull. Mathematical Biophysiology*, Vol. 5, pp. 115-133, 1943.

[21] V. Beiu, J. M. Quintana, and M. J. Avedillo, "VLSI Implementation of Threshold Logic: A Comprehensive Survey," *IEEE Trans. Neural Networks*, Vol. 14, No. 5, September 2003.

[22] S. Perri, P. Corsonello, and G. Cocorullo, "Area-Delay Efficient Binary Adders in QCA," *IEEE Trans. VLSI Systems*, Vol.22, No. 5, pp. 1174-1179, 2013.

[23] D. Abedi, G. Jaberipur, and M. Sangsefidi, "Coplanar Full Adder in Quantum-Dot Cellular Automata via Clock-Zone Based Crossover," *IEEE Trans. Nanotechnology*, Vol. 14, No. 3, pp. 497-504, May 2015.

[24] K. K. Yadavalli, A. O. Orlov, J. P. Timler, C. S. Lent, and G. L. Snider, "Fanout Gate in Quantum-Dot Cellular Automata" *Nanotechnology*, Vol. 18, no. 37, p. 375401, 2007.

[25] E. P. Blair, M. Liu, and C. S. Lent. "Signal Energy in Quantum-Dot Cellular Automata Bit Packets," J. *Computational and Theoretical Nanoscience*, Vol. 8, No.6, pp. 972-982, June 2011.

[26] K. Walus, T. J. Dysart, G. A. Jullien, and R. A. Budiamn, "QCADesigner: A Rapid Design and Simulation Tool for Quantum-Dot Cellular Automata," *IEEE Trans*. *Nanotechnology*, Vol. 3, No. 1, pp. 26–31, March 2004.

[27] J. Han, E. Taylor, J. Gao, and J. Fortes, "Reliability Modeling of Nanoelectronic Circuits," *Proc. 5th IEEE Conf. Nanotechnology*, Vol. 1, pp. 104-107, July 2005.

[28] H. Iwamura, M. Akazawa, and Y. Amemiya, "Single-Electron Majority Logic Circuits," *IEICE Trans. Electronics*, Vol. E81-C, No. 1, pp. 42-48, January 1998.

[29] T. Ohshima and R. A. Kiehl, "Operation of Bistable Phase-Locked Single-Electron Tunneling Logic Elements," *J. Applied Physics*, Vol. 80, pp. 912-923, July 1996.

[30] H. A. H. Fahmy and R. A. Kiehl, "Complete Logic Family Using Tunneling-Phase-Logic Devices," *Proc. 11th Int'l Conf. Microelectronics*, pp. 153-156, 1999.

[31] 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. 3215-18, November 2012.

[32] 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, 2007.