Decimal Full Adders Specially Designed for Quantum-Dot Cellular Automata

Dariush Abedi and Ghassem Jaberipur

***Abstract*—New emerging technologies, with low area/power/latency properties, are gaining momentum as replacements for CMOS. In particular, for quantum-dot cellular automata (QCA) realization, many arithmetic circuits have been redesigned. The basic QCA elements are a three-way majority gate and an inverter. The trivial mapping of logical circuits to their QCA equivalents via direct replacement of AND and OR gates with partially utilized majority (PUM) gates (i.e., with a "0" and "1" input, respectively) leads to exploitation of QCA basic components. Regarding the revitalized decimal arithmetic units in digital processors, researchers have begun to design decimal arithmetic circuits on QCA. For example, several QCA decimal full adders are trivially designed via the aforementioned direct mapping. However, only one proposition in the literature tries to make better use of majority gates, but yet replaces many AND and OR gates by PUMs. In this brief, we propose a QCA decimal full adder that is mostly composed of fully utilized majority gates (i.e., with no constant inputs), and rarely includes PUMs. The proposed circuit has been designed and tested by QCADesigner, and compared with relevant previous works, where the cell count, area, and delay, show 39%, 78%, 12% improvement, respectively.**

***Index Terms*—Quantum-dot cellular automata, decimal full adder.**

I. Introduction

SEVERAL digital processors that have been commercialized at the dawn of 21st century include decimal arithmetic hardware units (e.g., IBM eServer z900 [1], IBM POWER6 [2], IBM z10 [3], Fujitsu SparcX [4]). This is in response to growing demands for decimal computations that are required for banking and monetary applications [5]. As the processing demands approach astronomical dimensions, computer industry is considering a possible migration towards new emerging technologies to overcome CMOS limitations on speed and problems of extensive power dissipation on miniature devices.

Quantum dot cellular automata (QCA), is known as high speed low area/power nanoscale technology [6], [7]. Simple QCA circuits (e.g., a full adder) have already been fabricated [8] and more complex designs (e.g., 32-bit adders [9]) have been simulated via QCA design tools (e.g., QCADesigner [10]).

The basic QCA gates are the majority gate (QM) and inverter (QI), which together comprise a complete logic set. The reason is that $\mathcal{M}(a, b, 0) = a \wedge b$, and $\mathcal{M}(a, b, 1) = a \vee b$, where $\mathcal{M}$ denotes the majority function, as is described by Eqn. 1.

$$\mathcal{M}(a, b, \mathrm{c}) = (a \wedge b) \vee (a \vee b) \wedge \mathrm{c} \tag{1}$$

Any logical circuit that is realized via AND, OR, and NOT gates can be mapped to an equivalent QCA circuit via direct replacement of the latter three gates with their QCA counterparts. However, it is often possible to define the function of a given logical circuit in terms of majority gates and inverters. Such design approaches have led to more efficient QCA realization of logical functions. For example the function of a full adder (FA) that is normally defined by Eqn. set 2, can be reformulated as in Eqn. set 3 [11], and realized via 3 fully utilized (i.e., with no constant inputs) QMs and 2 QIs. Otherwise, trivial mapping of the conventional gate level full adder implementation with 7 AND/OR gates [12], would require 7 partially utilized (i.e., with one constant input) QMs.

$$s = a \oplus b \oplus c_{in}, c_{out} = ab \vee c_{in}(a \vee b) \tag{2}$$

$$s = \mathcal{M}(\mathcal{M}(a, b, \overline{c_{in}}), c_{in}, \overline{c_{out}}), c_{out} = \mathcal{M}(a, b, c_{in}) \tag{3}$$

The basic radix-10 arithmetic cell is the decimal full adder (DFA), whose QCA realization is proposed by some researchers. Few of such works have directly mapped an existing relevant CMOS design to the equivalent QCA realization [13]–[16]. However, in order to fully utilize the $\mathcal{M}$ gates, a DFA design is proposed in [17], instead of partially utilized ones that trivialy replace AND and OR gates. This is similar to the effort made by Pudi and Sridharan [20] in the context of parallel prefix QCA adders. Nevertheless, this design contains only 9 fully utilized and 35 partially utilized $\mathcal{M}$ gates.

In this brief, we aim to propose a QCA DFA, based on vast use of fully utilized $\mathcal{M}$ gates and as little partially utilized $\mathcal{M}$ gates as possible. This approach is expected to lead to less total $\mathcal{M}$ gates and more efficient QCA DFA.

The rest of this brief is organized as follows. A background on DFAs and two of the best QCA DFA designs are presented in Section II. The proposed DFA function, in terms of mostly fully utilized majority elements, is provided in Section III. Section IV regards the evaluation and comparison of the proposed and previous relevant works. Finally, Section V draws our concluding remarks.

Manuscript received December 20, 2015; revised February 21, 2017 and April 6, 2017; accepted May 9, 2017. Date of publication May 12, 2017; date of current version December 22, 2017. The work of G. Jaberipur was supported in part by the IPM under Grant CS1396-2-03, and in part by the Shahid Beheshti University. This brief was recommended by Associate Editor H.-T. Zhang. *(Corresponding author: Ghassem Jaberipur.)*

D. Abedi is with the Department of Computer Science and Engineering, Shahid Beheshti University, Tehran 19839-63113, Iran (e-mail: d_abedi@sbu.ac.ir).

G. Jaberipur is with the Department of Computer Science and Engineering, Shahid Beheshti University, Tehran 19839-63113, Iran, and also with the School of Computer Science, Institute for Research in Fundamental Sciences, Tehran 19538-33511, Iran (e-mail: jaberipur@sbu.ac.ir).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSII.2017.2703942

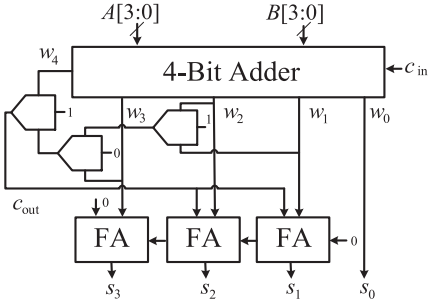

Fig. 1. Implementation of Eqn. 5.

II. Decimal Full Adder

Eqn. set 4 describes the function of a DFA, where A, B, c_{in}, S and c_{out} denote the main input digits, carry-in, digit-sum, and the carry-out, respectively.

$$S = |A + B + c_{in}|_{10}, c_{out} = \left\lfloor \frac{A + B + c_{in}}{10} \right\rfloor \tag{4}$$

The conventional implementation of Eqn. set 4, uses a 4-bit adder and a correction circuit to transform the generated hexadecimal carry and sum to the equivalent decimal values. This is done via Eqn. set 5, where $W = 16w_4 + \widehat{W} = A + B + c_{in}$, and $\widehat{W} = w_3w_2w_1w_0$. The required circuitry is depicted by Fig. 1, where the pentagon shape elements denote majority gates.

$$c_{out} = w_4 \vee w_3(w_2 \vee w_1), S = \widehat{W} + 6c_{out} \tag{5}$$

The previous QCA implementations of DFA include that of [17], which unlike some others has not directly mapped a conventional AND/OR/NOT circuit to the equivalent majority/NOT circuit. However, it turns out that in their QCA implementation the number of partially utilized $\mathcal{M}$ gates is almost 4 times as much as the fully utilized $\mathcal{M}$ gates (i.e., 35 versus 9). The implemented equations of this design is reproduced as in Eqn. 6, where $W' = w_3'w_2'w_1'w_0' = \widehat{W} + 6$, and $S = s_3s_2s_1s_0 = W'$, if $c_{out} = 1$, and $S = W$, otherwise, and $0 \le i \le 3$.

$$c_{out} = w_4 \vee w_3(w_2 \vee w_1), s_i = w_i'c_{out} \vee w_i\overline{c_{out}} \tag{6}$$

In order to increase the utilization of fully utilized $\mathcal{M}$ gates, Liu *et al.* [17] have further elaborated on Eqn. set 6, where the result is reproduced here as in Eqn. set 7. The AND and OR functions are realized by 18 partially utilized $\mathcal{M}$ gates. There are also 8 gates for generate ($g_i = a_ib_i$) and propagate ($p_i = a_i \vee b_i$) signals and 9 gates for the three multiplexers; hence total of 35 partially utilized $\mathcal{M}$ gates vs. the 9 fully utilized ones that are also represented within Eqn. set 7.

$$\begin{aligned}
&c_1 = \mathcal{M}(g_0, p_0, c_{in}), c_2 = \mathcal{M}(g_1, p_1, c_1),\\
&c_3 = \mathcal{M}(g_2, p_2, c_2)\\
&w_3' = g_3c_1,\\
&w_2' = g_3\overline{c_1} \vee p_3p_2(p_1 \vee c_1) \vee g_2g_1c_1,\\
&w_1' = \overline{\mathcal{M}(\overline{c_2}, \mathcal{M}(g_1, p_1, \overline{c_2}), c_1)},\\
&w_0' = \mathcal{M}(\overline{c_1}, \mathcal{M}(g_0, p_0, \overline{c_1}), c_{in}),\\
&w_3 = p_3 \vee c_3, w_2 = \overline{c_3}(p_2 \vee c_2), w_1 = \overline{w_1'}, w_0 = w_0',\\
&c_{out} = \mathcal{M}(g_3, p_3, p_2) \vee (p_3 \vee g_2)p_1 \times \vee(p_3 \vee \mathcal{M}(g_2, p_2g_1))c_1.
\end{aligned} \tag{7}$$

III. The Proposed New DFA

In this brief, we use Eqn. set 8, as a modified version of Eqn. set 5 in order to employ more fully utilized $\mathcal{M}$ gates (i.e., 16 vs. 12 partially utilized), with the due justification to follow. Note that $w_4w_3 = w_4w_2 = 0$, since otherwise $A + B + c_{in} > 19$, which cannot obviously occur. Also, $w_4w_3 = 0 \Longrightarrow (w_4\overline{w_3} = w_4, w_3\overline{w_4} = w_3)$, where similar results for $w_4w_2 = 0$ apply.

$$\begin{aligned}
s_0 &= w_0, s_1 = \mathcal{M}(\overline{w_1}, w_4\overline{w_1}, w_3w_2) \vee \overline{w_4 \vee w_3}w_1,\\
s_2 &= \mathcal{M}(w_4 \vee w_1, \overline{w_3}w_2, w_2 \vee \overline{w_1}),\\
s_3 &= \mathcal{M}(\overline{w_2}, w_4w_1, w_3\overline{w_1}),\\
c_{out} &= \mathcal{M}(w_4, w_3 \vee w_4, \mathcal{M}(w_1, w_2, w_3))
\end{aligned} \tag{8}$$

Recalling that $S = \widehat{W} + 6c_{out}$, we can express it as $s_3s_2s_1s_0 = w_3w_2w_1w_0 + 0c_{out}c_{out}0$, such that the expressions for bits of S and c_{out} (Eqn. set 8) can be derived as in Eqn. set 9, and further improved in Eqns. 10-13.

$$\begin{aligned}
s_0 &= w_0,\\
s_1 &= w_1 \oplus c_{out} = w_1 \oplus (w_4 \vee w_3(w_2 \vee w_1))\\
&= (w_1 \vee w_4 \vee w_3w_2)\overline{w_4w_1 \vee w_3w_1}\\
&= (w_1 \vee w_4 \vee w_3w_2)(\overline{w_1} \vee \overline{w_4}\ \overline{w_3})\\
&= w_4\overline{w_1} \vee w_3w_2\overline{w_1} \vee \overline{w_4}\ \overline{w_3}w_1,\\
s_2 &= w_2 \oplus c_{out} \oplus w_1c_{out}\\
&= w_2 \oplus c_{out}\overline{w_1} = w_2 \oplus (\overline{w_1}(w_4 \vee w_3(w_2 \vee w_1)))\\
&= w_2 \oplus (w_4\overline{w_1} \vee w_3w_2\overline{w_1})\\
&= (w_2 \vee w_4\overline{w_1})\overline{(w_3w_2\overline{w_1} \vee w_4w_2\overline{w_1})}\\
&= (w_2 \vee w_4\overline{w_1})(\overline{w_3} \vee \overline{w_2} \vee w_1)\\
&= \overline{w_3}w_2 \vee w_2w_1 \vee w_4\overline{w_1},\\
s_3 &= w_3 \oplus (c_{out}(w_2 \vee w_1))\\
&= w_3 \oplus (w_4(w_2 \vee w_1) \vee w_3(w_2 \vee w_1))\\
&= w_3 \oplus (w_4w_1 \vee w_3(w_2 \vee w_1))\\
&= (w_3 \vee w_4w_1)\overline{(w_4w_3w_1 \vee w_3w_2 \vee w_3w_1)}\\
&= (w_3 \vee w_4w_1)(\overline{w_3} \vee \overline{w_2}\ \overline{w_1})\\
&= w_3\overline{w_2}\ \overline{w_1} \vee w_4w_1,\\
c_{out} &= w_4 \vee w_3(w_2 \vee w_1)
\end{aligned} \tag{9}$$

A direct $\mathcal{M}$ gate implementation of the s_1 expression requires 5 partially utilized $\mathcal{M}$ gates and 1 fully utilized in 4 levels. However, further elaboration, as in Eqn. set 10, leads to a 3-level realization with no extra $\mathcal{M}$ gates.

$$\begin{aligned}
s_1 &= \overline{w_1}w_4\overline{w_1} \vee w_3w_2\overline{w_1} \vee w_4w_3w_2\overline{w_1} \vee \overline{w_4}\ \overline{w_3}w_1\\
&= \mathcal{M}(\overline{w_1}, w_4\overline{w_1}, w_3w_2) \vee \overline{w_4}\ \overline{w_3}w_1\\
&= \mathcal{M}(\overline{w_1}, w_4\overline{w_1}, w_3w_2) \vee \overline{w_4 \vee w_3}w_1
\end{aligned} \tag{10}$$

Complexity of a direct implementation of s_2 expression amounts to $5\mathcal{M}$ gates in 3 levels, with one shared with s_1 for $w_4\overline{w_1}$. However, the following leads to 4 independent $\mathcal{M}$ gates in 2 levels.

$$\begin{aligned}
s_2 &= (w_2 \vee w_4\overline{w_1})\overline{(w_3w_2\overline{w_1} \vee w_4w_2\overline{w_1})}\\
&= (w_2 \vee w_4\overline{w_1})(\overline{w_4}\ \overline{w_3} \vee \overline{w_2} \vee w_1)\\
&= \overline{w_4}\ \overline{w_3}w_2 \vee w_2w_1 \vee w_4\overline{w_2}\ \overline{w_1}\\
&= \overline{w_3}w_2 \vee w_2w_1 \vee w_4\ \overline{w_1}\\
&= (w_4\overline{w_3}w_2 \vee \overline{w_3}w_2w_1) \vee (w_4w_2 \vee w_4\overline{w_1} \vee w_2w_1)
\end{aligned}$$

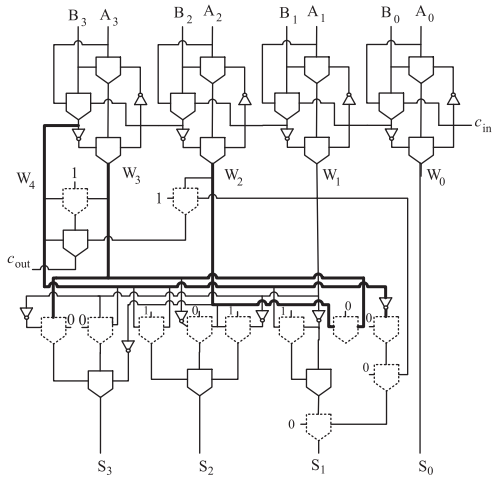

Fig. 2. Majority implementation of Eqn. set 8.

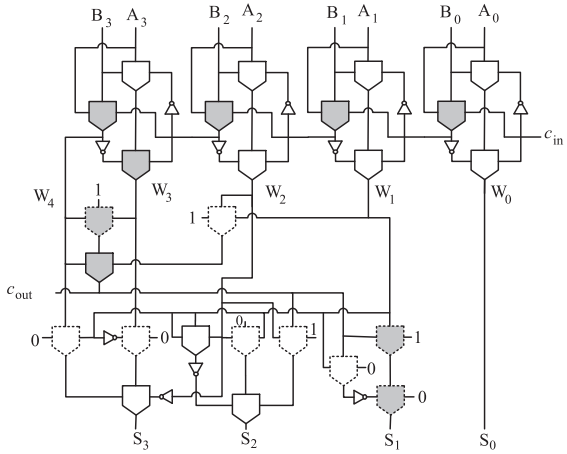

Fig. 3. DFA with 16 fully- and 9 partially-utilized $\mathcal{M}$ gates.

$$\begin{aligned} &\times \vee (\overline{w_3} w_2 \vee \overline{w_3} w_2 \overline{w_1}) \\ &= \mathcal{M}(w_4 \vee w_1, \overline{w_3} w_2, w_2 \vee \overline{w_1}) \end{aligned} \tag{11}$$

Direct implementation of the s_3 expression requires 4 $\mathcal{M}$ gates in 3 levels. However, the following leads to a 3-$\mathcal{M}$ realization in 2 levels.

$$\begin{aligned} s_3 &= w_4 \overline{w_2} w_1 \vee w_3 \overline{w_2}\ \overline{w_1} \vee w_4 w_3 w_1 \overline{w_1} \\ &= \mathcal{M}(\overline{w_2}, w_4 w_1, w_3 \overline{w_1}) \end{aligned} \tag{12}$$

The c_{out} expression normally requires 3 $\mathcal{M}$ gates in 3 levels. However, the following realization (Eqn. 13) reduces the number of levels to 2.

$$\begin{aligned} c_{out} &= w_4 \vee w_3(w_2 \vee w_1) \\ &= w_4 \vee w_4 w_3 \vee w_4(w_3 w_2 \vee w_3 w_1 \vee w_2 w_1) \\ &\quad \times \vee w_3(w_2 \vee w_1 \vee w_1 w_2) \\ &= w_4 w_4 \vee w_4 w_3 \vee w_4(w_3 w_2 \vee w_3 w_1 \vee w_2 w_1) \\ &\quad \times \vee w_3(w_3 w_2 \vee w_3 w_1 \vee w_1 w_2) \\ &= w_4(w_4 \vee w_3) \vee (w_4 \vee w_3 \vee w_4)\mathcal{M}(w_1, w_2, w_3) \\ &= w_4(w_3 \vee w_4) \vee w_4 \mathcal{M}(w_1, w_2, w_3) \\ &\quad \times \vee (w_3 \vee w_4)\mathcal{M}(w_1, w_2, w_3) \\ &= \mathcal{M}(w_4, w_3 \vee w_4, \mathcal{M}(w_1, w_2, w_3)) \end{aligned} \tag{13}$$

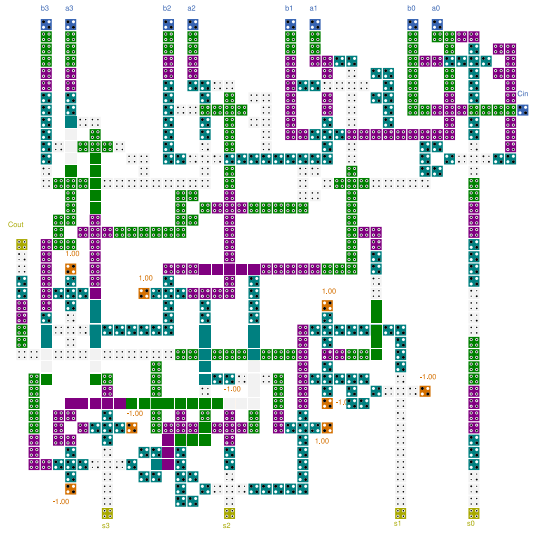

Fig. 4. QCA realization of DFA of Fig. 3.

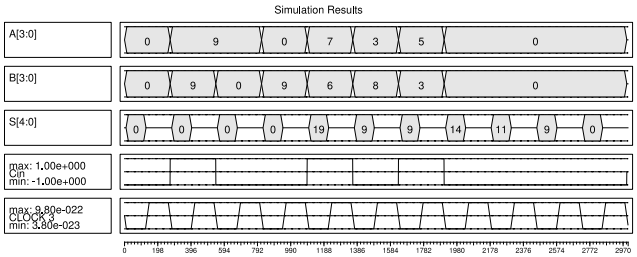

Fig. 5. I/O pattern for the layout of Fig. 4.

Fig. 2 presents a 28 $\mathcal{M}$ gate implementation of the latter described DFA, with 16 fully utilized versus 12 partially utilized (distinguished by dashed boxes) $\mathcal{M}$ gates.

One drawback of this circuit is the long backward wires that are required in the paths for s_1 and s_2 (the thick lines). This leads to many crossovers (QX), which are hard to implement and degrade the circuit efficiency.

To remedy this problem, we propose the new equations 14 and 15, for s_1 and s_2, respectively (with due justification to follow), where only the c_{out} wire needs to be back warded. These new equations can be realized with 16 fully utilized and 9 partially utilized $\mathcal{M}$ gates as in Fig. 3.

$$\begin{aligned} s_1 &= c_{out} \oplus w_1 = (c_{out} \vee w_1)\overline{c_{out} w_1} \\ &= \mathcal{M}(w_1 \vee c_{out}, \overline{c_{out} w_1}, 0) \end{aligned} \tag{14}$$

$$\begin{aligned} s_2 &= w_2 \oplus c_{out} \oplus (c_{out} w_1) = w_2 \oplus (c_{out} \overline{w_1}) \\ &= \overline{w_2} c_{out} \overline{w_1} \vee w_2 \overline{c_{out} \overline{w_1}} = \overline{w_1 w_2} c_{out} \vee w_1 w_2 \vee w_2 \overline{c_{out}} \\ &= \overline{w_1 w_2} c_{out} \vee w_1 w_2 (1 \vee c_{out}) \vee w_2 \overline{c_{out}} (1 \vee \overline{w_1}) \\ &= \overline{w_1\ w_2} c_{out} \vee w_1 w_2 \vee w_1 w_2 c_{out} \vee w_2 \overline{c_{out}} \vee \overline{w_1} w_2 \overline{c_{out}} \\ &= \overline{w_1 w_2} c_{out} \vee \overline{w_1 w_2} s_2 \vee \overline{w_1} w_2 \overline{c_{out}} \vee \overline{w_1} c_{out} \overline{c_{out}} \\ &\quad \times \vee w_2 \overline{w_2 c_{out}} \vee w_2 c_{out} \overline{c_{out}} \vee w_1 w_2\ \vee w_1 w_2 c_{out} \\ &= \overline{w_1 w_2}(c_{out} \vee w_2) \vee \overline{w_1 c_{out}}(c_{out} \vee w_2) \\ &\quad + \overline{w_2 c_{out}}(c_{out} \vee w_2) \vee w_1 w_2 (w_2 \vee c_{out}) \\ &= (\overline{w_1 w_2} \vee \overline{w_1 c_{out}} \vee \overline{w_2 c_{out}})(c_{out} \vee w_2) \vee w_1 w_2 (w_2 \vee c_{out}) \\ &\quad \times \vee w_1 w_2 (\overline{w_1 w_2} \vee \overline{w_1 c_{out}} \vee \overline{w_2 c_{out}}) \\ &= \mathcal{M}((\overline{w_1 w_2} \vee \overline{w_1 c_{out}} \vee \overline{w_2 c_{out}}), (c_{out} \vee w_2), w_1 w_2) \\ &= \mathcal{M}\big(\overline{\mathcal{M}(w_1, w_2, c_{out})}, c_{out} \vee w_2, w_1 w_2\big) \end{aligned} \tag{15}$$

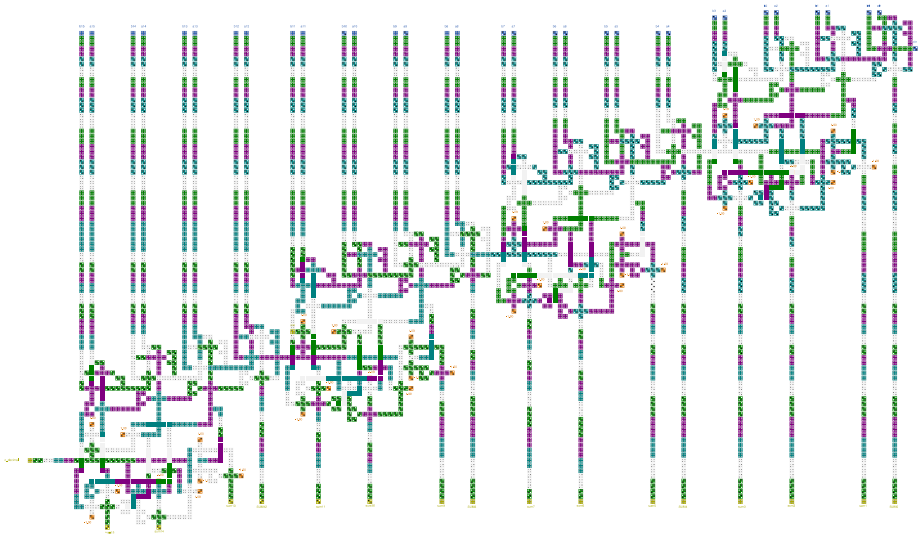

Fig. 6. 4-digit decimal adder.

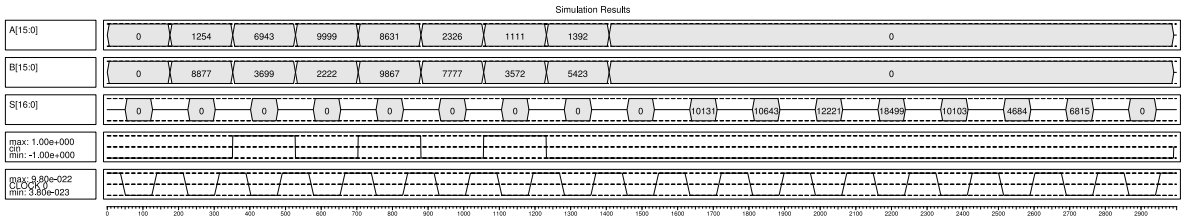

Fig. 7. I/O pattern for the layout of Fig. 6.

The required QCA layout is produced by the QCADesigner [10], as in Fig. 4, where the FA layouts are borrowed from [18]. Note that this layout is more efficient than what would be achieved for Fig. 2, since the number of crossovers is reduced from 30 to 20, and the number of $\mathcal{M}$ gates is reduced from 31 to 25. The new DFA is simulated by coherence vector and bistable simulation engines with default QCADesigner parameters (except for Number of Samples = 30000). The resulted I/O pattern for the layout of Fig. 4 is presented in Fig. 5 for 7 different pairs of decimal digits, where the output are shown after 12 clock zones (i.e., 3 clock cycles).

In order to evaluate the proposed DFA in an actual QCA decimal adder, we have realized a 4-digit decimal adder, as in the layout of Fig. 6. The corresponding I/O pattern is offered in Fig. 7, where seven pairs of 4-digit decimal numbers are used as the input, with alternative c_{in} values.

IV. Evaluation and Comparison

Table I compiles figures of merit (i.e., number of QCA cells, QM, QI, QX, CZ, and the consumed area) for the proposed DFA and other previous relevant works, where CZ denotes the number of clock zones, as a measure of delay.

The figures regarding [13]–[16] are less competitive, which is mainly due to their design strategy of direct replacement of AND/OR gates with QM components. Several measures for [13] are not available. Note that although the QM count of [14] (i.e., 27) is competitive with ours (i.e., 25 QMs), its cell count, area and delay is twice, triple and over twice than ours, respectively. Contrary to the previous works, we are reporting both sum and c_{out} delays. The latter is crucial in delay evaluation of multi-digit decimal adders (as in Fig. 6), where, that of [16] is not available. Note that the c_{out} delay figures regard those of DFAs within a multi-digit adder. Otherwise that of a single isolated DFA would include one more zone.

According to the contents of Table I, the first circuit of [17] merits best, among the previous works. However, although its ripple carry implementation of DFA uses one less $\mathcal{M}$ gate than the proposed design, its QCA realization uses 39% more cells with 57% more delay in the carry path. The main reason is that we use FA of [18], which is composed of less QCA cells than that of [17]. This reason also partially applies in justification of more cell utilization in [14], where also more QMs do exist. Moreover, our careful placement of QMs reduces the wiring cell count. Latency of the faster version of the same work (designated as [17]-2) is yet one clock zone (12%) more than that of the proposed design, while uses 174% more cells.

Another recently practiced evaluation scheme [19] uses a tunable cost function that is described by Eqn. 16, where M, I, C, and T refer to QM, QI, and QX, and number of clock cycles. Also, k, l, and p are the cost tuning parameters. We have evaluated all the designs of Table I (except for the serial realization of [16]), for n-digit ripple carry decimal adders ($1 \leq n \leq 128$). The tuning parameters are set to $k, l, p = 2$, as is recommended in [19]. The values for M, I, C, and T variables are n times those of 1-digit adder (see Table I). The results are depicted as the curves of Fig. 8, where the bottom curve clearly illustrates the advantage of the proposed design.

$$Cost_{QCA} = \left(M^k + I + C^l\right) \times T^p, \quad k, l, p \geq 1. \tag{16}$$

TABLE I
COMPARISON OF FIGURES OF MERITS

DFA design	# of $\mathcal{M}$		Cell		Area		Delay (CZ)		# of QI	# of QX
	Fully	Partial	Count	Ratio	μm^2	Ratio	sum	c_{out}		
Proposed	16	9	669	1	0.76	1	12	7	12	20
[17]-1	16	8	932	1.39	1.36	1.78	19	11	14	17
[17]-2	9	35	1838	2.74	1.86	2.44	10	8	9	70
[13]	-	-	-	-	5.8	7.63	20	-	16	-
[14]	19	8	1348	2.01	2.28	3	32	20	16	36
[15]	0	79	3560	5.32	4.0	5.26	28	32	34	121
[16]	1	43	1130	1.68	1.77	2.32	40	-	18	16

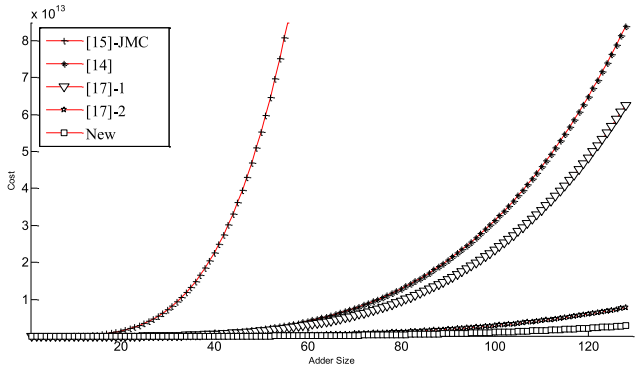

Fig. 8. Cost comparison based on the cost function of [19].

V. CONCLUSION

Many logical circuits that have been realized in CMOS, including those of computer arithmetic circuits are also realized in QCA platform. Instead of trivial mapping of AND/OR gates to partially utilized majority gates, one can reformulate the logical expressions of the corresponding CMOS designs in terms of maximizing the use of fully utilized majority gates. This is generally expected to lead to less number of total majority gates, as is the case in the design of binary full adders with only 3 majority gates.

Several previous attempts for QCA implementation of decimal full adders (DFA) have led to rather inefficient designs, where the number of partially utilized majority gates is considerably more than that of fully utilized ones.

With careful examination of the DFA logical equations, we reformulated them towards increased use of fully utilized majority gates, such that the proposed DFA realization is composed of 16 fully and only 9 partially utilized majority gates, and 12 inverters. The QCA realization of this DFA uses 669 cells that are due to 25 QMs, 12 QIs, 14 multilayer QXs, and 6 coplanar QXs. The total consumed area is 0.76 μm^2, and the c_{in}-c_{out} delay is equal to 7 clock zones. The delay and area improvements with respect to fastest (least costly) previous DFA is 1 (4) less clock zones, and 59% (44%) less area consumption; that is 0.76 versus 1.86 (1.36) μm^2.

REFERENCES

[1] F. Y. Busaba, C. A. Krygowski, W. H. Li, E. M. Schwarz, and S. R. Carlough, "The IBM z900 decimal arithmetic unit," in *Proc. Conf. Rec. 35th Asilomar Conf. Signals Syst. Comput.*, vol. 2. Pacific Grove, CA, USA, 2001, pp. 1335–1339.

[2] S. Shankland, "IBM's POWER6 gets help with math, multimedia," *ZDNet News*, Oct. 2006.

[3] C. F. Webb, "IBM z10: The next-generation mainframe microprocessor," *IEEE Micro*, vol. 28, no. 2, pp. 19–29, Mar./Apr. 2008.

[4] T. Yoshida *et al.*, "Sparc64 X: Fujitsu's new-generation 16-core processor for unix servers," *IEEE Micro*, vol. 33, no. 6, pp. 16–24, Nov./Dec. 2013.

[5] L.-K. Wang, M. J. Schulte, J. D. Thompson, and N. Jairam, "Hardware designs for decimal floating-point addition and related operations," *IEEE Trans. Comput.*, vol. 58, no. 3, pp. 322–335, Mar. 2009.

[6] C. S. Lent, P. D. Tougaw, W. Porod, and G. H. Bernestein, "Quantum cellular automata," *Nanotechnology*, vol. 4, no. 1, pp. 49–57, 1993.

[7] R. A. Wolkow *et al.*, "Silicon atomic quantum dots enable beyond-CMOS electronics," in *Field-Coupled Nanocomputing* (LNCS 8280). Berlin, Germany: Springer, 2014, pp. 33–58.

[8] G. H. Bernstein, "Quantum-dot cellular automata: Computing by field polarization," in *Proc. Design Autom. Conf.*, Anaheim, CA, USA, 2003, pp. 268–273.

[9] H. Cho and E. E. Swartzlander, Jr., "Adder and multiplier design in quantum-dot cellular automata," *IEEE Trans. Comput.*, vol. 58, no. 6, pp. 721–727, Jun. 2009.

[10] K. Walus, T. J. Dysart, G. A. Jullien, and R. A. Budiman, "QCADesigner: A rapid design and simulation tool for quantum-dot cellular automata," *IEEE Trans. Nanotechnol.*, vol. 3, no. 1, pp. 26–31, Mar. 2004.

[11] W. Wang, K. Walus, and G. A. Jullien, "Quantum-dot cellular automata adders," in *Proc. 3rd IEEE Conf. Nanotechnol.*, vol. 2. San Francisco, CA, USA, 2003, pp. 461–464.

[12] A. Pishvaie, G. Jaberipur, and A. Jahanian, "Redesigned CMOS (4; 2) compressor for fast binary multipliers," *Can. J. Elect. Comput. Eng.*, vol. 36, no. 3, pp. 111–115, Summer 2013.

[13] M. Taghizadeh, M. Askari, and K. Fardad, "BCD computing structures in quantum-dot cellular automata," in *Proc. Int. Conf. Comput. Commun. Eng.*, Kuala Lumpur, Malaysia, 2008, pp. 1042–1045.

[14] F. Kharbash and G. M. Chaudhry, "The design of quantum-dot cellula-rautomata decimal adder," in *Proc. IEEE Int. Multitopic Conf.*, Karachi, Pakistan, 2008, pp. 71–75.

[15] M. A. Gladshtein, "The signal propagation delay reduction of the combinational adder of decimal digits encoded by the Johnson-Mobius code," *Autom. Control Comput. Sci.*, vol. 44, no. 2, pp. 103–109, May 2010.

[16] M. Gladshtein, "Quantum-dot cellular automata serial decimal adder," *IEEE Trans. Nanotechnol.*, vol. 10, no. 6, pp. 1377–1382, Nov. 2011.

[17] W. Liu, L. Lu, M. O'Neill, and E. E. Swartzlander, "Cost-efficient decimal adder design in quantum-dot cellular automata," in *Proc. IEEE Int. Symp. Circuits Syst.*, Seoul, South Korea, 2012, pp. 1347–1350.

[18] 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.

[19] 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.

[20] 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, Exp. Briefs*, vol. 59, no. 10, pp. 678–682, Oct. 2012.