# ESOP Transformation to Majority Gates for Quantum-dot Cellular Automata Logic Synthesis

David Y. Feinstein and Mitchell A. Thornton
Department of Computer Science and Engineering
Southern Methodist University
Dallas, TX, USA
**{dfeinste, mitch}@engr.smu.edu**

## Abstract

*We investigate the benefit of using ESOP minimization in the synthesis of quantum-dot cellular automata (QCA). We determine the size and delay of multi-input XOR and AND/OR gates when implemented in using 3-input majority gates that are easily realized in QCA, and use our results in our QCAexor estimator tool. We found that most benchmarks have at least one output that can be better minimized using EXOR techniques.*

## 1. Introduction

It has been projected that by around 2015, mainstream CMOS technology will not be able to continue along Moore's curve of speed and density that currently doubles every 18 months. Emerging nanotechnologies have been developed in the hope of achieving performance that go beyond the end-of-curve limits of CMOS. *Quantum-dot Cellular Automata* (QCA) is such a promising technology.

QCA was first suggested by Lent, et al. in 1993 [7,14]. Promising size density of several orders of magnitude smaller than CMOS, fast switching time in the range of $10^{-12}$ to $10^{-15}$ seconds, and extremely low power, has caused QCA to become a topic of intense research. Experimental circuits utilizing metal dots were demonstrated in the late 90s [1], followed by the recent development of molecular QCA circuits [10], that put QCA at the front line of emerging nanotechnologies.

QCA is radically different than today's CMOS technology. Information in QCA propagates along a line of cells via Coulombic charge interaction as opposed to conventional electronic current in CMOS devices. Unlike the gate versatility of CMOS, QCA technology natively implements the 3-input majority gate as the basic building block. QCA also requires a new multi-zone clock mechanism for proper data propagation. Recently, the QCADesigner synthesis and simulation tool was developed by Walus et al. [15,16], allowing convenient experimentation with virtual QCA circuits.

In this paper we are interested in how to use the QCA native 3-input majority gate in an efficient way to synthesize logic circuits. More particularly, we investigate techniques for direct transformation of *Exclusive-OR Sum-of-Products* (ESOP) forms of logic description into an implementation with 3-input majority gates. It is well known that ESOP minimization of the logic description can often (but not always) produce better results (less and smaller cubes) than the more common inclusive-OR Sum-of-Products (SOP) forms [3,4,8,11,12, 13].

The paper is organized as follows. Quantum-dot Cellular Automata circuits and related work on majority logic reductions are described in more detail in Section 2. In Section 3, we describe our synthesis approach using ESOP minimization of the target circuit. Section 4 contains experimental results. Conclusions and areas of future improvement are presented in Section 5.

## 2. Background and Related Work

The basic QCA cell can be schematically depicted as a square substrate with four quantum dots located at each of cell's four corners [7]. Each cell contains only two free electrons so that they tend to be in the two bistable arrangements shown in Fig. 1.a. The electrons cannot occupy two adjacent dots due to the *Coulombic* repulsion between them. We arbitrarily select the first cell as logic 1 encoding, leaving the second cell to encode logic 0.

The same *Coulombic* repulsion aligns the data within the cells when they are arranged along a "wire" as shown in Fig. 1.b. Here we assume that information flows from left to right and the cells in the dotted box are next to respond. Various clocking mechanisms, as described in [16] are used to enforce proper data propagation. It should be noted that since no current is flowing between the cells, QCA is essentially a very

low power technology (even though current is used in some implementations for the clocking zones [18]).

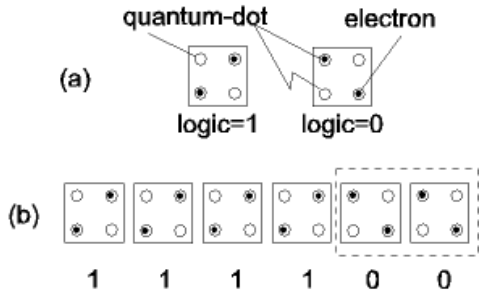



**Figure 1: Basic QCA cell (a) and a wire (b)**

## 2.1. QCA Majority Gate and Inverters

The native gate for QCA is created when five cells are placed adjacently as shown in Fig. 2. The result is a majority gate with three inputs and one output. This three input majority gate performs the binary function $M(a,b,c) = ab + bc + ac$. Basic two variable AND and OR functions can be readily implemented by $M(a,b,0)$ and $M(a,b,1)$ respectively. However, the majority gate is not a universal gate since it is unable to implement the NOT function. Therefore, majority logic gates must be combined with inverter circuits in order to implement arbitrary logic.

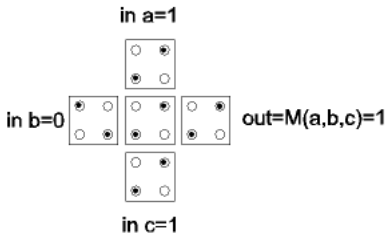



**Figure 2: QCA 3-input majority gate**

Fig. 3 shows a fork inverter. The output cell is affected by both end cells of the fork to insure reliable operations. Other inverters that use $45^0$ and $90^0$ cells orientations have been developed [6].

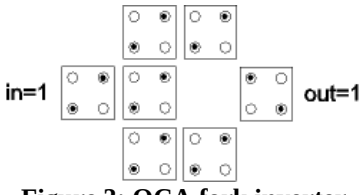



**Figure 3: QCA fork inverter**

## 2.2. Early work on 3-input Majority Gate Synthesis

Early interest in logic synthesis based on majority voters was briefly experienced in Japan during the late 50s and early 60s due to the invention of the Parametron by E. Goto [5] and its inclusion in early Japanese computers [9]. The Parametron natively realized 3-input majority function but quickly became obsolete due to the emergence of high speed transistors. Miyata [9] distinguished between majority elements of the *first kind*, which implement the simple 2-input AND or OR gates, and majority elements of the *second kind* where the third input is advantageously used to minimize the design. Since majority voter circuits require relatively complex implementation within mainstream CMOS circuits, they posed little interest as a building block for logic synthesis [3].

## 2.3. Recent Work on Majority Logic Reduction

QCA prompted new research interest with 3-input majority gate synthesis. Zhang et al. [19] described an algebraic method for reducing the number of majority gates required to implement all arbitrary 3-variable Boolean functions. Observing the three-cube representation of such functions, there are 8 minterms that may be included, providing a total of 256 possible Boolean functions. They identify 13 standard functions which can be used with rotation and translation to represent any of these arbitrary 256 Boolean functions. They proposed minimal implementations of these functions using 3-input majority gates.

Walus et al. [17] further refined their previous work in [19] to improve three of the standard function implementations using Karnaugh maps minimizations. Improvement is made when the total number of majority gates is reduced or when the number of levels (delay) is reduced. Boolean functions of more than 3 inputs must be factored (or use algebraic division) in order to decompose the function into 3-input functions that use this and the previous work. Zhang and Jha [20] developed the MALS tool that produces optimized majority logic circuits. They use factorization methods in a pre-synthesis step that can reduce the size of the circuit. Their factorization technique is applied to the SOP representation of the desired logic.

Another approach was presented by Xiaobo et al. [18] proposing a QCA based Programmable Logic Array (PLA) implemented by a combination of AND and OR planes. Each PLA cell can be used as logic or as "wire" thus achieving a flexible architecture.

Yet another approach was suggested by Huang et al. [6] for tile-based QCA logic synthesis, where the cells are arranged in regular $K \times K$ tiles. Such tiles might be fully or partially populated, producing majority-like and inverter functions. They

demonstrated that the 3x3 cell tiles show much promise for efficient designs.

### 2.4. ESOP vs. SOP Logic Synthesis

In standard Sum-of-Products (SOP) logic synthesis, the required circuit is converted into a list of cubes which are *inclusively* OR-ed together to form the required output. It is crucial to minimize the number of cubes and their sizes (number of literals that are AND-ed in the cube) in order to achieve the most efficient synthesis results. Heuristic tools like MINI and Espresso find optimal or near-optimal minimization of SOPs [2,11].

Exclusive-OR-Sum-of-Products (ESOP) synthesis is an alternative to SOP synthesis in which the cubes are exclusively OR-ed together. Important sub-groups of ESOP are the Reed-Muller expressions which were gradually developed over the past seventy years [13]. Interest in ESOP synthesis increased in the early 90s when Sasao and Besslich [12] noted that on average, ESOPs require fewer products than corresponding SOP forms. However, it is well known that a given technique for subsets of the ESOP family (e.g. PPRM, FPRM, KRO, PSDRM, PSDKRO, or GRM) may score much more worth than SOP based synthesis in particular applications [11]. Fleisher et al. [4] developed a computer algorithm that minimizes Reed-Muller canonical forms which examine an EXOR cube in a parallel way to the work described in Section 2.3..

We are motivated by the potential of obtaining better minimization of arbitrary binary functions using ESOPs as opposed to standard SOP synthesis for application to QCA circuit synthesis. This ESOP investigation is further motivated in view of the better testability that is usually obtained with ESOP designs [11,13].

## 3. Our Approach

In this research we compare the size and delay of QCA majority gate based virtual circuits that are formed by either SOP or ESOP synthesis of standard benchmarks (in two level *.PLA format). In the first step of our research we perform minimizations in the classical logic sense. We then use our estimation program to determine and compare the size and cost of our SOP and ESOP minimizations when they are mapped to QCA majority gate technology. We use ESPRESSO to minimize our benchmarks for SOP minimizations, and EXORCISM4, a tool by Mishchenko and Perkowski [8] to minimize our benchmarks in ESOP form. Since EXORCISM4 uses a PLA input format, we provide this tool with the ESPRESSO output files. We have verified that the size of ESOP implementations by the EXORCISM4 tool when input with the disjoint version of the PLA file (ESPRESSO –Ddisjoint) is equal to the size achieved with the SOP minimized version of the PLA file. Before we describe our test flow, we investigate multi-input XOR implementations with QCA majority gates and compare them with the corresponding AND/OR implementation.

### 3.1. Implementing Multi-input XOR Gates by 3-input Majority Gates

A 2-input XOR circuit can be realized with three 3-input majority gates as shown in Fig. 4. The circuit performs the well known function

$$a \oplus b = \bar{a}\, b + a\, \bar{b} \qquad (1)$$

and it uses two majority gates to implement the AND functions and one majority gate to implement the OR. All these majority gates are of the *first kind*, in accordance with [9]. The schematic diagram of this gate is also shown in Fig. 4.

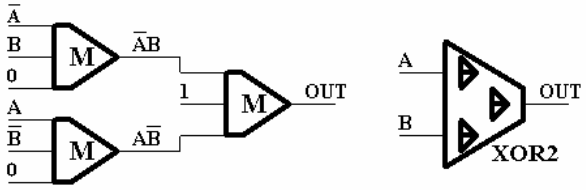



**Figure 4: 2-input XOR gate use 3 majority gates. Schematic symbol shown on the right**

A naïve implementation of a 3-input XOR gate can be achieved by connecting two XOR2 gates as shown in Fig. 5.a.

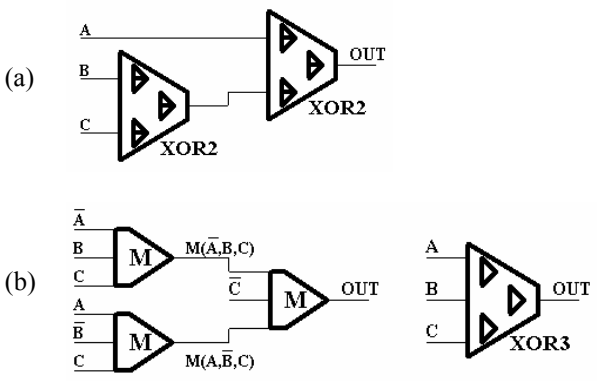


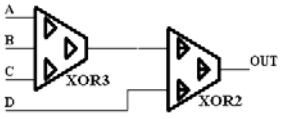

**Figure 5: 3-input XOR gate. (a) Naïve implementation with 2-input XOR gates. (b) minimized implementation and its symbol**

This implementation is fairly inefficient as it requires a total of 6 majority gates of the *first kind*. A better implementation of the **XOR3** gate is illustrated in Fig. 5.b. Here we efficiently use only 3 majority gates of the *second kind*. We can see that the **XOR2** gate is equivalent to the minimized **XOR3** gate of Fig. 5.b. with one input set to 0, performing

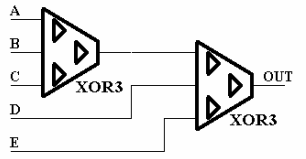
$$a \oplus b = a \oplus b + 0 \qquad (2)$$

Fig. 6 explores the architecture of XOR gates with 4,5,6,7,8, and 9 inputs. These results indicate that only **XOR3**, **XOR5**, **XOR7** and **XOR9** are relatively efficient in the sense that they exclusively use majority gates of the *second kind*. Although we assume, following [9], that once all inputs of the 3-input majority gate are used we have a relatively efficient implementation, it is still possible that the techniques shown in [19] combined with factoring or decomposition may still improve our results. However, our construction provides a reasonable baseline estimation of the complexity of the general *k*-input XOR QCA implementation.

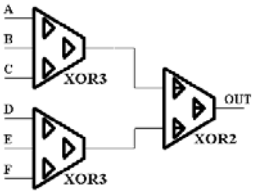
(a) 4-in XOR



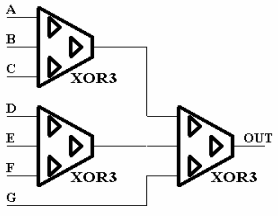

(b) 5-in XOR



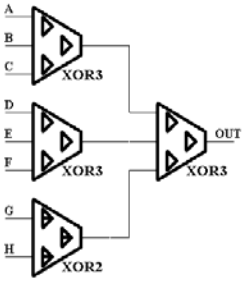

(c) 6-in XOR




(d) 7-in XOR




(e) 8-in XOR




(f) 9-in XOR

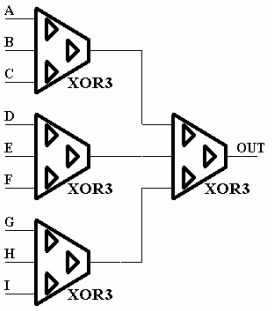



**Figure 6: Implementation of 4, 5, 6, 7, 8, and 9 input XOR gates**

We summarize our analysis for the number of majority gates (circuit size) and number of levels (delay) in Table 1. We note that each XOR3 circuit in Fig. 6 also requires 3 inverters, which incur higher cost in QCA versus CMOS technology. The more efficient implementations are marked in bold in Table 1.

TABLE 1
SIZE AND DELAY OF MULTI-INPUT XOR GATES IN QCA

| Inputs | 2 | **3** | 4 | **5** | 6 | **7** | 8 | **9** |
|---|---|---|---|---|---|---|---|---|
| Majority gates | 3 | **3** | 6 | **6** | 9 | **9** | 12 | **12** |
| Inverters | 2 | **3** | 5 | **6** | 8 | **9** | 11 | **12** |
| Levels | 2 | **2** | 4 | **4** | 4 | **4** | 4 | **4** |

Based on this construction, the number of levels $d_{xor}$ is:

$$d_{xor} = 2 \times \lceil \log_3 n \rceil \qquad (3)$$

The number of 3-input majority gates $m_{xor}$ is

$$m_{xor} = 3 \times \left\lfloor \frac{n}{2} \right\rfloor \qquad (4)$$

The number of inverters $i_{xor}$ is

$$i = \begin{cases} m & if \quad n \quad is \quad odd \\ m-1 & if \quad n \quad is \quad even \end{cases} \qquad (5)$$

Clearly, the complexity of XOR implementation is $O(\log_3 n)$ delay and $O(n)$ in size.

For example, an **XOR10** requires 14 majority gates, 13 inverters, and has a delay of 4.

### 3.3. Implementing Multi-input AND/OR Gates by 3-input Majority Gates

We intuitively construct a multi-input AND or OR gate using majority gates of the first kind arranged in a binary tree-like architecture. Fig. 7 illustrates the construction of a 9-input gate. Note that setting S to 0 selects the AND functions, while S=1 selects the OR function.

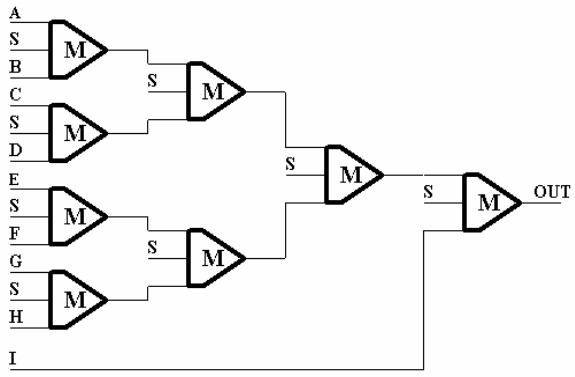



**Figure 7: Implementation of a 9-input AND or OR gate. The AND function is selected when the select line S=0, the OR functions selected when S=1**

Based on this construction, the number of levels $d_{and}$ is:

$$d_{and} = \lceil \log_2 n \rceil \quad (6)$$

The number of 3-input majority gates $m$ is

$$m_{and} = n - 1 \quad (7)$$

The OR and AND functions have the same size and delay since the only difference is the value of S. Notice that the OR/AND construction follows the same asymptotic complexity ($O(\log_2 n)$ delay and $O(n)$ size) as the XOR construction. However, it is clear that even before considering the extra cost of inverters for the XOR construction, the AND/OR is significantly smaller.

### 3.3. Comparison between SOP and ESOP Virtual Implementations

An overview of our approach is shown in Fig. 8. In our pre-processing step, the benchmark files in the *.pla format are minimized by ESPRESSO to produce the SOP cube list. The resulting files are input to EXORCISM4 to produce corresponding ESOP cube lists in a *.esop file. For size comparisons between benchmark files in the *.pla and *.esop formats, the reader is asked to refer to [8].

Our QCAexor tool accepts both the *.pla and *.esop forms of the investigated benchmark file. The tool matches the individual outputs and works on one output at a time. Using the size and delay functions of sections 3.1 and 3.2, QCAexor estimates the size of the SOP cube lists when implemented in majority gates. Similarly, it estimates the size and delay of the ESOP cube list. The tool also logs the number of inverters required by the XOR gate implementation since inverters are expensive resources for QCA. After its' analysis is complete, QCAexor saves the comparison results in a comma delimited file *.cvs. The data in the *.cvs is shown in the tables 2-5 of Section 4.

The virtual implementations based on the constructions of section 3.1 and 3.2 require the ESOP cube list to be implemented by one multi-input XOR function and several multi-input AND functions. Similarly, the SOP cube list uses the simpler OR implementation. Therefore, it is expected that ESOP synthesis may show a gain over SOP synthesis only when the number of cubes (that is, the size of the XOR) and their sizes are smaller. For this reason, we found it useful to make the comparison per each output in the ON-list rather than the entire circuit (which is the common practice when considering standard CMOS logic [8]).

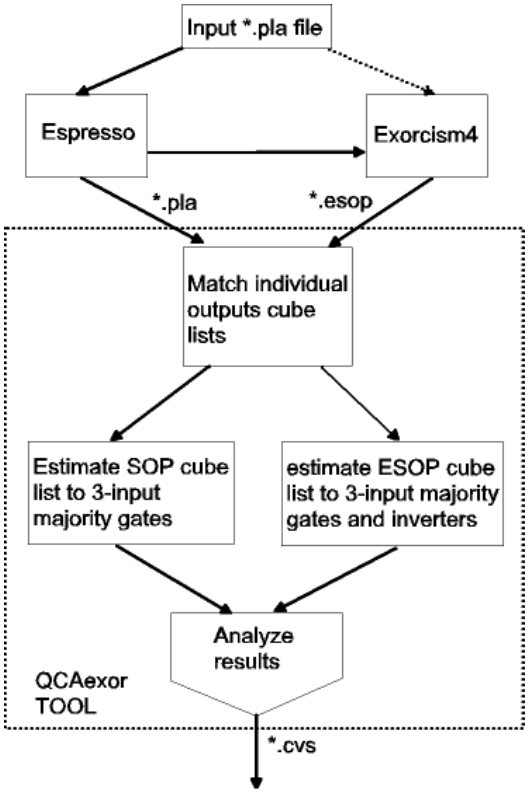



**Figure 8: An overview of our QCAexor Tool**

We emphasize that for this research, we use general EXOR minimization (as provided by the EXORCISM4 tool). We note that Falkowski and Kannurao [21] suggested a method to identify disjunctive decomposition using majority gates for balanced Boolean functions based on the coefficients of their Walsh spectrum. We plan to consider using such spectral approaches in our future research.

## 4. Experimental Results

We first found that it is not possible for an ESOP minimization to produce better results for 3-input general Boolean functions than the reduction obtained by the algebraic and K-map methods of Section 2.2. This result was expected since a simple XOR of 2 variables consumes three majority gates while an inclusive OR of 2 variables is implemented by a single majority gate. In fact, after minimizing the 13 standard functions of [19] in ESOP form, we were able to obtain several instances of smaller cubes but the complexity of the 2-input XOR implementation caused the circuits to use more majority gates. The only exceptions were the trivial single cube function $f$=1 and function 13, which is identical to the XOR form $a \oplus b \oplus c$ as is shown in Fig. 6.

We have analyzed a large number of the standard PLA benchmarks and show representative results in Tables 2-5. In the tables, the first column depicts the current output investigated in the table rows. Columns 2-4 show the results for that output using the SOP specification data and column 5-8 shows the results using the ESOP data. The second and fifth columns show the number of products associated with the output. The third and sixth columns show the total size in majority gates. For ESOP implementations, column 7 shows the extra number of inverters required by the top XOR gate. We show the maximum delay, as measured by majority gate levels, in columns 4 and 8.

The GAIN ESOP column shows the *size* benefit (if any) when an output is implemented in ESOP form instead of SOP, and it is computed as follows:

*GAIN = 100% (SOPsize–ESOsize–ESOPinverters)/SOP size.*

A negative result indicates that the SOP form produces implementations of smaller size than ESOP forms. Table 2 shows the results the benchmark *ALU4.PLA*. The Espresso minimization of this 14 input and 8 output files had 575 cubes, while the EXROCISM4 minimization produced 426 cubes. The large negative results for outputs 0, 2 and 3 validates our approach to investigate each output individually as these negative results would clearly mask the benefit of ESOP implementation for outputs 1,4,5, and 6.

TABLE 2
COMPARISON OF ALU4.PLA – 14 INPUTS/8 OUTPUTS

| **Out** | **SOP prds** | **SOP size** | **SOP delay** | **ESOP prds** | **ESOP size** | **ESOP invs** | **ESOP delay** | **GAIN ESOP** |
|---|---|---|---|---|---|---|---|---|
| 0 | 16 | 67 | 7 | 31 | 218 | 45 | 7 | -292% |
| 1 | 12 | 51 | 6 | 5 | 15 | 6 | 6 | 58% |
| 2 | 50 | 247 | 9 | 51 | 466 | 75 | 9 | -119% |
| 3 | 72 | 423 | 10 | 130 | 1327 | 195 | 10 | -259% |
| 4 | 186 | 1594 | 12 | 53 | 481 | 78 | 12 | 64% |
| 5 | 90 | 614 | 10 | 25 | 161 | 36 | 10 | 67% |
| 6 | 36 | 206 | 9 | 9 | 37 | 12 | 9 | 76% |
| 7 | 182 | 1927 | 12 | 210 | 2183 | 315 | 12 | -29% |

The *VG2.PLA* benchmark shown in Table 3 demonstrates a case where ESOP synthesis should not be used at all. In fact, for this 25 input and 8 output benchmark, ESPRESSO minimized it to 110 cubes while EXORCISM4 minimized to 184 cubes.

TABLE 3
COMPARISON OF VG2.PLA – 25 INPUTS/8 OUTPUTS

| Out | SOP prds | SOP size | SOP delay | ESOP prds | ESOP size | ESOP invs | ESOP delay | GAIN ESOP |
|---|---|---|---|---|---|---|---|---|
| 0 | 5 | 46 | 6 | 7 | 74 | 9 | 6 | -80% |
| 1 | 10 | 173 | 9 | 14 | 263 | 21 | 9 | -64% |
| 2 | 5 | 46 | 6 | 7 | 73 | 9 | 6 | -78% |
| 3 | 10 | 103 | 8 | 22 | 249 | 33 | 8 | -173% |
| 4 | 40 | 235 | 9 | 84 | 813 | 126 | 9 | -299% |
| 5 | 5 | 16 | 4 | 7 | 31 | 9 | 4 | -150% |
| 6 | 30 | 161 | 8 | 66 | 550 | 99 | 8 | -303% |
| 7 | 5 | 16 | 4 | 7 | 31 | 9 | 4 | -150% |

The remaining examples for the 5XP1.PLA and the DC2.PLA clearly indicate that there is a potential when selectively using ESOP minimizations.

TABLE 4
COMPARISON OF 5XP1.PLA – 7 INPUTS/10 OUTPUTS

| Out | SOP prds | SOP size | SOP delay | ESOP prds | ESOP size | ESOP invs | ESOP delay | GAIN ESOP |
|---|---|---|---|---|---|---|---|---|
| 0 | 12 | 50 | 6 | 8 | 37 | 12 | 6 | 2% |
| 1 | 12 | 54 | 7 | 12 | 62 | 18 | 7 | -48% |
| 2 | 19 | 86 | 8 | 14 | 57 | 21 | 8 | 9% |
| 3 | 15 | 62 | 6 | 8 | 27 | 12 | 6 | 37% |
| 4 | 10 | 38 | 6 | 6 | 17 | 9 | 6 | 31% |
| 5 | 5 | 15 | 4 | 3 | 6 | 3 | 4 | 40% |
| 6 | 8 | 34 | 6 | 4 | 7 | 6 | 6 | 61% |
| 7 | 2 | 3 | 0 | 2 | 3 | 3 | 0 | -100% |
| 8 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0% |
| 9 | 3 | 10 | 3 | 3 | 14 | 3 | 3 | -70% |

TABLE 5
COMPARISON OF DC2.PLA – 8 INPUTS/7OUTPUTS

| Out | SOP prds | SOP size | SOP delay | ESOP prds | ESOP size | ESOP invs | ESOP delay | GAIN ESOP |
|---|---|---|---|---|---|---|---|---|
| 0 | 4 | 17 | 5 | 4 | 22 | 6 | 5 | -64% |
| 1 | 7 | 40 | 6 | 9 | 49 | 12 | 6 | -52% |
| 2 | 10 | 52 | 7 | 11 | 57 | 15 | 7 | -38% |
| 3 | 12 | 70 | 7 | 15 | 84 | 21 | 7 | -50% |
| 4 | 9 | 49 | 6 | 7 | 31 | 9 | 6 | 18% |
| 5 | 8 | 44 | 6 | 6 | 31 | 9 | 6 | 9% |
| 6 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0% |

We summarize our results with various benchmarks in Table 6. Column 4 indicates the total number of cubes in the ESPRESSO minimization of the benchmark, while column 5 shows the minimization results with EXORCISM4. In column 5 we show how many of the outputs can be implemented based on the ESOP minimization with a gain over the results with SOP minimization. We also show the best gain and the worst loss achieved during the processing of all the outputs individually.

TABLE 6
POTENTIAL GAIN IN USING ESOP MINIMIZATION PRIOR TO QCA MAJORITY GATE IMPLEMENTATION

| **Benchmark** | **Inputs** | **Outputs** | **ESPR. Minim.** | **EX-4 Minim.** | **ESOP GAINS** | **Best gain** | **Worst loss** |
|---|---|---|---|---|---|---|---|
| 5xp1.pla | 7 | 10 | 65 | 31 | 6 of 10 | 61% | -100% |
| dc2.pla | 8 | 7 | 39 | 32 | 3 of 7 | 18% | -64% |
| alu1.pla | 12 | 8 | 19 | 16 | 0 of 8 | 0% | -150% |
| seq.pla | 41 | 35 | 336 | 246 | 6 of 35 | 74% | -333% |
| vg2.pla | 25 | 8 | 110 | 184 | 0 of 8 | none | -303% |
| alu4.pla | 14 | 8 | 575 | 426 | 4 of 8 | 76% | -259% |
| 9sym.pla | 9 | 1 | 86 | 51 | 1 of 1 | 8% | none |
| c8.pla | 28 | 18 | 79 | 50 | 4 of 18 | 31% | -166% |
| z5xp1.pla | 7 | 10 | 65 | 33 | 9 of 10 | 89% | -20% |
| tial.pla | 14 | 8 | 581 | 428 | 4 of 4 | 76% | -317% |
| x6dn | 39 | 5 | 82 | 95 | 0 of 5 | none | -120% |
| vda | 17 | 39 | 93 | 87 | 4 of 39 | 12% | -2820% |

It is important to note that QCAexor was able to perform the entire analysis for all our benchmarks at a fraction of a second. For this reason, we did not report timing in the experimental results. Therefore, combining our tool with EXORCISM4 and ESRESSO will not significantly extend the CPU time needed for the initial SOP or ESOP minimization step.

## 5. Conclusions and Future Improvements

We have investigated the methodical construction of multi-variable XOR, AND, and OR gates using native 3-input majority gates in QCA-based architectures. While we have not proved that our constructions are optimal, they clearly provide easily computable upper bounds for the size and delay expected from QCA complex structures.

We have developed an estimation tool, QCAexor, that can determine if a given output should be minimized using SOP or ESOP forms prior to actual QCA synthesis. We demonstrated that significant size reduction can be achieved in most standard benchmarks for at least some of the outputs. The estimation time to predict if a given output should be implement based on ESOP or SOP minimization is extremely low.

In the future, we plan to investigate the potential of using spectral techniques instead of (or in addition to) the general EXOR minimization used in this research.

## ACKNOWLEDGMENT

The authors thank Alan Mischenko for providing the EXCORSICM4 tool used in this research.

## References

[1] I. Amlani, A.O. Orlov, G.L. Snider and C.S. Lent, "Demonstration of a functional quantum-dot cellular automata cell", *J. Vac. Sci. Technol.*, B, 16, 1998, pp. 3795-3799.

[2] R.K. Brayton, G.D. Hachtel, C.T. McMullen and A.L.M. Sangiovanni-Vincentelli, **Logic Minimization Algorithms for VLSI Synthesis**, Kluwer Academic Publishers, Botson, 1984.

[3] B.J. Falkowski and S. Kannurao, "Spectral theory of disjunctive decomposition for balanced Boolean functions", *13th Int'l Conf on VLSI Design, 2000*, pp. 506-511.

[4] H. Fleisher, M. Tavel, J. Yeager, "A computer algorithm for minimizing Reed-Muller canonical forms", *IEEE Trans. Comp.* Vol. 36. No. 2, February 1987, pp. 247-250.

[5] E. Goto, "The parametron: a digital computer element which utilized parametric oscillation", In *Proc. IRE*, Vol. 47, No. 8, 1959, pp. 1304-1316.

[6] J. Huang, M. Momenzadeh, L. Schiano, M. Ottavi, and F. Lombardi, "Tile-based QCA design using majority-like logic primitives", *J. Emerging Tech. in Comp. Sys. (JETC)*, Vol. 1, No. 3, Oct. 2005, pp. 163-185.

[7] C.S. Lent, P.D. Tougaw, and W. Porod, "Bistable saturation of in coupled quantum dot for quantum cellular automata", In *Appl. Phys. Letters*, Vol. 62, No. 7, February 1993, pp. 714-716.

[8] A. Mischenko and M. Perkowski, "Fast heuristic minimization of exclusive-sums-of-products", *5th Int'l Reed-Muller Workshop*, 2001, pp. 242-250.

[9] F. Miyata, "Realization of arbitrary logical functions using majority element", *IEEE TEC*, Vol. EC-12, No. 3, 1963, pp. 183-191.

[10] H. Qi, S. Sharma, Z.H. Li, G.L. Snider, A. Orlov, C.S. Lent and T.P. Fehlner, "Molecular quantum cellular

automata cells", In *J. Am. Chem. Soc*. 2003, 125, pp 15250-15259.

[11] T. Sasao, **Switching Theory for Logic Synthesis**, Kluwer Academic Publishers, 1999.

[12] T. Sasao and Ph. W. Besslich, "On the complexity of MOD-2 sum PLA's", *IEEE Trans. On Computers*, Vol. 34, No. 2, 1990, pp. 262-266.

[13] R.S. Stanković, C. Moraga and J.T. Astola, "Reed-Muller expressions in the previous decade", In *5th Int'l Reed-Muller Workshop*, 2001, pp. 7-26.

[14] P.D. Tougaw, C.S. Lent, and W. Porod, "Bistable saturation in coupled quantum dot cells", In *J. Appl. Phys*, Vol. 74, No. 5, September 1993, pp. 3558-3566.

[15] 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. on Nanotechnology ,* Vol. 3, No. 1, March 2004, pp. 26 – 31.

[16] K, Walus and G.A. Jullien, "Design tools for an emerging SoC technology: quantum-dot cellular automata", *Proc. of the IEEE*, Vol. 94, No. 6, June 2006, pp. 1225 – 1244.

[17] K. Walus, G. Schulhof, G., G.A. Jullien, R. Zhang, and W. Wang, "Circuit design based on majority gates for applications with quantum-dot cellular automata", *38th Asilomar Conference on Signals, Systems and Computers*, Vol. 2, Nov. 2004, pp. 1354 - 1357.

[18] S.H. Xiaobo, M Crocker, M. Niemier, Y. Minjun and G. Bernstein, "PLAs in quantum-dot cellular automata", *Proc. of Emerging VLSI Technologies and Architectures (ISVLSI'06)*, March 2006, 6 pp.

[19] R. Zhang, K. Walus, W. Wei, and G.A. Jullien, "A method of majority logic reduction for quantum cellular automata", In *IEEE Trans.on Nanotechnology*, Vol, 3, No. 4, Dec. 2004, pp 443 – 450.

[20] R. Zhang and N.K. Jha, "Threshold/majority logic synthesis and concurrent error detection targeting nanoelectronic implementations", In *GLSVLSI'06*, April 2006, pp. 8-13.

.