Optimized FFT QCA Implementation using Five-input Majority Gate

Rami Akeela*

* Santa Clara University, Santa Clara, CA, USA, rakeela@scu.edu

ABSTRACT

Among the existing nanotechnologies that are getting more attention from researchers is Quantum dot Cellular Automata (QCA), which is a possible substitute for CMOS. It has the potential for higher speed and low area and power consumption. Therefore, researchers are constantly investigating new implementations and applications and striving to optimize them for different criteria. Amongst these applications is the Fast Fourier Transform (FFT) algorithm. A novel architecture for partial parallel FFT processor was presented in [1], which used inefficient older designs of adders and multipliers, which had more gates and occupied more area, and hence, consumed more power.

In this paper, new designs for a Ripple Carry Adder and a Subtractor were developed to cater to the need of a new, more compact multiplier, namely, the Pipelined Array Multiplier, so one can use them in an FFT implementation using QCA. These designs have been designed, optimized, and simulated using QCADesigner, and successfully achieved a higher level of compactness and speed, largely due to the utilization of the Five-input Majority Gate [2].

1 Introduction

Over the years, engineers have been able to continually shrink the size of CMOS transistors and thereby package more of them on the same chip. However, as we approach the physical limits of photolithography as well as device physics, this task has become more expensive and complicated. Studies show that the progress towards increasing the chip complexity while maintaining the speed and constraining the power dissipation has slowed considerably. It is now believed that within the next two decades, the semiconductor industry will have to start using new nanoelectronic devices [3]. The International Roadmap for Semiconductors has enumerated several nanoelectronic alternatives including Resonant Tunneling Diodes (RTD), Quantom-dot Cellular Automata (QCA), and Single Electron Tunneling (SET) [4]. Amongst all these technologies, QCA promises to provide the highest device density with the low power consumption ans high switching speeds [5]. In addition, QCA uses the same technology to build both, the logic gates and the wires carrying logic signals.

Even though QCA has attractive properties, building large QCA architectures has not been very successful. The primary limitation to QCA is the availability of only two basic building blocks: an inverter and a three-input majority gate (MAJ_3). This was the motivation to introduce a new powerful building block for QCA technology, a five-input majority gate (MAJ_5) shown in Fig. 1 [2]. In that paper, its uses were demonstrated and provided an example of a bit-serial adder.

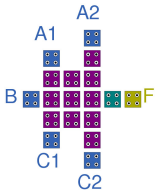


Fig. 1: A Five-input Majority Gate.

This addition to the library of QCA building blocks inspired going back to complex arithmetic circuits, like adders and multipliers, and study the effect of using MAJ_5 on both area and speed. In doing so, it was obvious these new designs for adders and multipliers should be part of larger implementations to utilize the full effect of the new compact gate. Such applications include the Fast Fourier

Transform (FFT) algorithm, which is an important tool in Digital Signal Processing. M. Awais *et al* in [1] discussed such an application and introduced a novel approach to designing a partial parallel FFT processor, and then implemented it using QCA technology. Their implementation used designs for an adder and a multiplier that are inefficient in terms of speed and cell count, which defeats the purpose of an FFT. In this paper, new designs for the aforementioned implementations are introduced, then simulated using QCADesigner.

2 Ripple Carry Adder

In this section, a QCA full adder that is coplanar and based on MAJ_5 developed earlier is introduced. In this design, at first the carry value is calculated by a three-input majority gate (MAJ_3), and then the carry value is inverted and used as the two inputs of the five-input majority gate (MAJ_5).The output of MAJ_5 gate generates the sum value:

$$SUM = MAJ_5(a, b, c_{in}, \overline{c}_{out}, \overline{c}_{out})$$

$$C_{out} = MAJ_3(a, b, c_{in}).$$

The full adder was simulated by QCADesigner, following the same set of parameters adopted for the bit-serial adder in [2], and using the coherence vector, as well.

A new wire-crossing technique developed by Sang-Ho Shin *et al* [6] is adopted and used in the design of the QCA full adder. This technique is based on the state of cells, and the relation between the locked and relaxed states, and hence achieves a more efficient wire-crossing, without adding more cost to the design, in terms of delays or number of cells.

The new design for the full adder can be seen in Fig. 2, where the value for $Carry_{out}$ is read through a wire that does not use cell rotation or translation. Implementation of this design with the new technique is done with minimum complexity and number of cells, and with a simplification of connections, while keeping delays the same (two clock cycles), and consequently same processing speed.

To design a 4-bit Ripple Carry Adder, the Full adder is used. In this design, the LSB bits are added first and the $Carry_{in}$. The next adder in

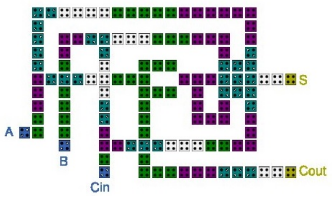


Fig. 2: The Full Adder using MAJ_5.

the circuit then takes the next significant bits and so on. Thus, four full adders need to be cascaded, and connect the carry out of one adder to the carry in for the next full adder. This is an ineffective method for addition of large number of bits, but for small number of bits, it is the least complicated and easy-to-implement design. The layout of an optimized RCA adder can be seen in Fig. 3.

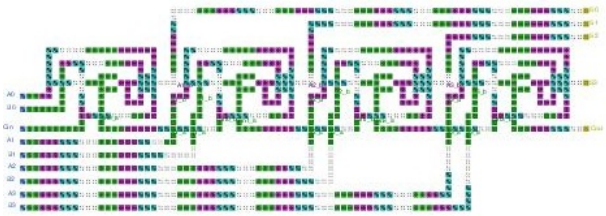


Fig. 3: 4-bit Ripple Carry Adder (RCA) Adder using MAJ_5.

It should be noted that the design uses a fewer number of cells and is more dense due to the fact that wire-crossing no longer needs rotated or shifted cells. It is also more straightforward to feed the inputs to the adder without having to worry about correct number of inversions (rotated cells).

3 Subtractor

After designing and optimizing the ripple carry adder (RCA), a 4-bit Subtractor is designed and optimized. The circuit is shown in Fig. 4.

The subtractor circuit was designed by adding the complementary circuit. In subtraction, the complement has been used in such a manner that if we want to obtain $A-B$, we should calculate $2's$ complement of number B and add it to number A. $2's$ complement is calculated by obtaining $1's$ complement of the word and adding it to 1. $1's$ comple-

ment is obtained by the inverter, and number 1 is added through the carry input.

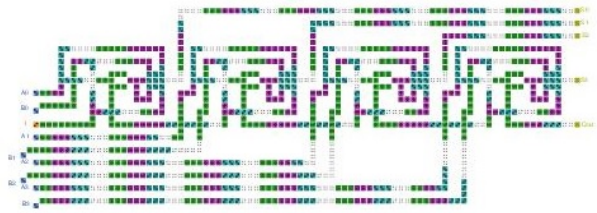


Fig. 4: 4-bit Subtractor using a MAJ_5, derived from RCA adder in Fig. 3.

In Fig. 4, the B word's bits were inverted each by using an inverter for each one of them, and the $Carry_{in}$ was fixed at the polarization value of 1, which is binary 1, in order to obtain B's 2's complement.

4 Pipelined Array Multiplier

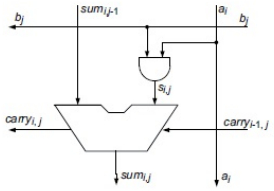
The array multiplier, by definition, is formed by a regular lattice of identical functional units (building blocks), following the paper-and-pencil multiplication algorithm with n-bit binary input operands $A = (a_{n1}, \ldots, a_1, a_0)$ and $B = (b_{n1}, \ldots, b_1, b_0)$, resulting in a 2n-bit output value $M = (m_{2n1}, \ldots, m_1, m_0)$, where a_0, b_0, and m_0 are the least significant bits, respectively. The smallest unit implementing all the basic array multiplier functionality has three-bit operand values, and maps the following computation straight on the hardware:

Fig. 5b shows the logical structure of the multiplier cell, which computes a single bit multiplication of bits a_i and b_j using an AND-gate, which is a MAJ_3, forming a sum $s_{i,j}$, which is then combined by the full adder designed earlier with a previous sum output $s_{i,j1}$ from a cell above, and a previous carry output $carry_{i1,j}$ from a cell to the right [7]. The QCA implementation of the multiplier cell is shown in Fig. 6.

This cell acts as a functional unit and a building block to the 2-bit Array Multiplier. Corresponding to the rows in the middle section of the equation in Fig. 5a, each row in the multiplier array shown in Fig. 7 forms a partial product, and sums it with the output of the row above. The final result is available in parallel form on the outputs $Sum_{i,j}$

$$
\begin{array}{cccc}
 & & a_1 & a_0 \\
\times & & b_1 & b_0 \\
\hline
 & & a_1b_0 & a_0b_0 \\
+ & a_1b_1 & a_0b_1 & \\
\hline
m_3 & m_2 & m_1 & m_0
\end{array}
$$

(a)



(b)

Fig. 5: (a) Multiplication Operation and (b) Multiplier Cell Schematic.

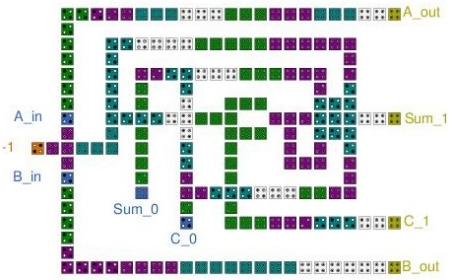


Fig. 6: Array Multiplier Cell.

of the bottom cells in each column, and the most significant bit (MSB) appears on the last carry output $Carry_{n-1,n-1}$. The computation's logic structure is purely combinatorial, but performance reasons usually suggest introducing some pipelining. The characteristics of QCA technology prevent designing large sections of combinatorial logic, because the reliable unclocked logic block size seems to be limited below a primitive gate (MAJ_3) [8]. Thus, the multiplier has to be divided into very fine-grained pipeline stages and operated with the four-phase QCA clocking.

The QCA implementation of the pipelined array multiplier shown in Fig. 7 is depicted in Fig. 8.

5 Conclusion

In this work, new optimized architectures for Ripple Carry Adder, Subtractor, and Pipelined Ar-

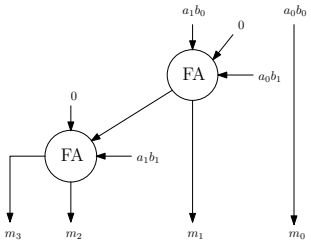


Fig. 7: A 2×2 Array Multiplier for operands a_1a_0 and b_1b_0.

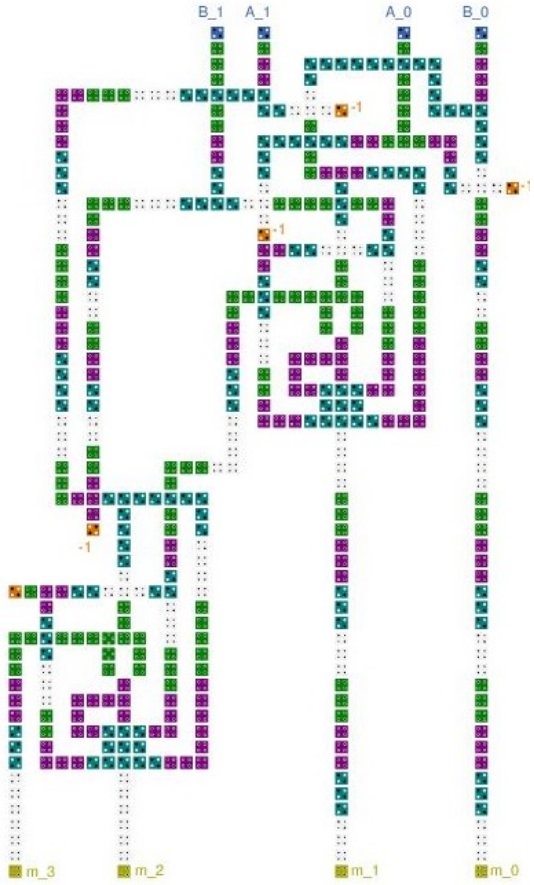


Fig. 8: A QCA implementation of a 2×2 Array Multiplier for operands a_1a_0 and b_1b_0.

ray Multiplier were presented. The designs were based on the Five-input Majority gate introduced in an earlier work. They proved to be more compact, *i.e.* fewer cells, and faster, *i.e.* fewer clock phases, and hence consumed less power. The aforementioned designs were introduced as more efficient alternatives to the designs adopted by the FFT QCA implementation in [1]. Due largely to MAJ_5, the FFT implementation should achieve higher speeds and compactness, which is main purpose of an FFT. Extensions of the work reported here can lead to a wide variety of applications in QCA technology.

REFERENCES

[1] M. Awais, M. Vacca, M. Graziano, and G. Masera, "Fft implementation using qca," in *Electronics, Circuits and Systems (ICECS), 2012 19th IEEE International Conference on*, pp. 741–744, Dec 2012.

[2] R. Akeela and M. Wagh, "A five-input majority gate in quantum-dot cellular automata," NSTI-Nanotech, 2011.

[3] D. A. Antonelli, D. Z. Chen, T. J. Dysart, and X. S. Hu, "Quantum-dot cellular automata (QCA) circuit partitioning: Problem modeling and solutions," in *Proc. of Design Auto. Conf.*, (San Diego, CA), June 2004.

[4] "The international technology roadmap for semiconductors: Emerging research devices." http://www.itrs.net/, 2005.

[5] R. Zhang, K. Walus, W. Wang, and G. A. Jullien, "A majority reduction technique for adder structures in quantum-dot cellular," in *Proceedings of SPIE 5559*, pp. 91–100, 2004.

[6] S.-H. Shin, J.-C. Jeon, and K.-Y. Yoo, "Design of wire-crossing technique based on difference of cell state in quantum-dot cellular automata.," *International Journal of Control & Automation*, vol. 7, no. 4, 2014.

[7] I. Hanninen and J. Takala, "Pipelined array multiplier based on quantum-dot cellular automata," in *Circuit Theory and Design, 2007. ECCTD 2007. 18th European Conference on*, pp. 938–941, Aug 2007.

[8] K. Kim, K. Wu, and R. Karri, "Towards designing robust qca architectures in the presence of sneak noise paths," in *Design, Automation and Test in Europe, 2005. Proceedings*, pp. 1214–1219 Vol. 2, March 2005.