2017

Exploration of Majority Logic Based Designs for Arithmetic Circuits

Carson Labrado
University of Kentucky, carson_labrado@hotmail.com
Digital Object Identifier: https://doi.org/10.13023/ETD.2017.147

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained needed written permission statement(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine) which will be submitted to UKnowledge as Additional File.

I hereby grant to The University of Kentucky and its agents the irrevocable, non-exclusive, and royalty-free license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless an embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's thesis including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

Carson Labrado, Student

Dr. Himanshu Thapliyal, Major Professor

Dr. Cai-Cheng Lu, Director of Graduate Studies

EXPLORATION OF MAJORITY LOGIC BASED DESIGNS FOR ARITHMETIC CIRCUITS

THESIS

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Electrical Engineering
in the College of Engineering
at the University of Kentucky

By
Carson Labrado
Lexington, Kentucky
Director: Dr. Himanshu Thapliyal
Lexington, Kentucky
2017

ABSTRACT OF THESIS

EXPLORATION OF MAJORITY LOGIC BASED DESIGNS
FOR ARITHMETIC CIRCUITS

Since its inception, Moore's Law has been a reliable predictor of computational power. This steady increase in computational power has been due to the ability to fit increasing numbers of transistors in a single chip. A consequence of increasing the number of transistors is also increasing the power consumption. The physical properties of CMOS technologies will make this powerwall unavoidable and will result in severe restrictions to future progress and applications. A potential solution to the problem of rising power demands is to investigate alternative low power nanotechnologies for implementing logic circuits. The intrinsic properties of these emerging nanotechnologies result in them being low power in nature when compared to current CMOS technologies. This thesis specifically highlights quantum dot celluar automata (QCA) and nanomagnetic logic (NML) as just two possible technologies. Designs in NML and QCA are explored for simple arithmetic units such as full adders and subtractors. A new multilayer 5-input majority gate design is proposed for use in NML. Designs of reversible adders are proposed which are easily testable for unidirectional stuck at faults.

KEYWORDS: Nanomagnetic logic, quantum dot cellular automata, reversible logic

Carson Labrado

May 2, 2017

EXPLORATION OF MAJORITY LOGIC BASED DESIGNS
FOR ARITHMETIC CIRCUITS

By

Carson Labrado

Dr. Himanshu Thapliyal

(Director of Thesis)

Dr. Cai-Cheng Lu

(Director of Graduate Studies)

May 2, 2017

(Date)

Table of Contents

Table of Contents iii

List of Figures v

List of Tables vii

1 Introduction 1

1.1 Contribution of Thesis 3

1.2 Outline of Thesis 4

2 Background 5

2.1 Quantum Dot Cellular Automata 5

2.2 Nanomagnetic Logic 9

2.3 Reversible Logic 11

2.3.1 Conservative Reversible Logic 13

2.3.2 Conservative Reversible Fredkin Gate 13

3 Design of Adder and Subtractor Circuits in QCA 16

3.1 Design of Proposed Full Adder 17

3.2 Design of Proposed Ripple Carry Adder 19

3.3 Design of Proposed Full Subtractor 20

3.4 Design of Proposed Ripple Borrow Subtractor 21

3.5 Conclusion 22

4 Proposed 5-Input Majority Gate in NML Computing 28

4.1 Design Verification Framework 29

4.1.1 NML Generation Program 29

4.2 Proposed 5-Input Majority Gate 32

4.2.1 Design Verification 33

4.3 Design of Proposed Full Adder 34

4.3.1 Design Verification 35

4.4 Proposed Full Adder Design Comparisons 35

4.5 Design of Proposed Full Subtractor 36

4.5.1 Design Verification 37

4.6 Conclusion 37

5 Design of Testable Adder Circuits for NML Computing 43

5.1 Design Methodology 1 of Proposed Testable Reversible Ripple Carry Adder 43

5.2 Design Methodology 2 of Proposed n-bit Testable Reversible Ripple Carry Adder 47

5.3 Comparison of Proposed n-bit Ripple Carry Adder Design Methodologies 49

5.4 Conclusion 50

6 Conclusions 51

References 53

Vita 57

List of Figures

1.1 Approaches for maintaining technology scaling by category [1] (© 2015 IEEE). 2

2.1 QCA Logic 6
2.2 Rotated QCA Logic Cell 6
2.3 QCA Crosswire 7
2.4 QCA Cell Code 8
2.5 QCA Data Propagation 8
2.6 3-Input Majority Gate 9
2.7 5-Input Majority Gate 9
2.8 Logic Values in NML Computing 10
2.9 NML Cell Alignment 10
2.10 3-Input Majority Gate 11
2.11 XOR Gate 12
2.12 Feynman Gate 12
2.13 Fredkin Gate 14
2.14 NML Implementation of Fredkin Gate 15

3.1 Proposed Full Adder 18
3.2 Simulation of Proposed QCA Full Adder 23
3.3 Proposed 4-bit Ripple Carry Adder QCA Implementation 24

3.4 Proposed Full Subtractor 25
3.5 Simulation of Proposed QCA Full Subtractor 26
3.6 Proposed 4-bit Ripple Borrow Subtractor QCA Implementation . . . 27

4.1 3D 5-Input Majority Gate 34
4.2 5-Input Majority Gate Simulation 38
4.3 3D Full Adder 39
4.4 Full Adder Simulation 40
4.5 Proposed 3D Full Subtractor 41
4.6 Full Subtractor Simulation 42

5.1 CRTB 1 45
5.2 CRTB 2 45
5.3 CR Full Adder Method 1 45
5.4 CR 4-bit Ripple Carry Adder Method 1 47
5.5 CR Full Adder Method 2 48
5.6 CR 4-bit Ripple Carry Adder Method 2 48

List of Tables

2.1 A table beside a figure 12

2.2 Feynman Gate Truth Table 12

2.3 Fredkin Gate Truth Table 14

3.1 QCADesigner Simulation Settings 17

3.2 QCA Values for Proposed Full Adder 19

3.3 QCA Values for Proposed 4-bit Ripple Carry Adder 20

3.4 QCA Values for Proposed Full Subtractor 21

3.5 QCA Values for Proposed 4-bit Ripple Borrow Subtractor 22

4.1 3D Cell Model Color Codes 32

4.2 Proposed 3D Full Adder Comparison 36

5.1 CRTB 1 Truth Table 44

5.2 CRTB 2 Truth Table 44

5.3 Fredkin Cost and Delay Comparison 50

Chapter 1

Introduction

Moore's Law states that the number of transistors on a chip will double roughly every two years [2]. This trend has served as a reliable predictor of increases in computational power. A main driver of this trend has been the ability to continually shrink the sizes of transistors themselves. Insurmountable physical limitations have caused this rate to slow as time has gone on. One significant issue facing CMOS technology is there is a physical limitation to how small a transistor can be. A second major hurdle is the increase in transistors also results in an increase in power consumption. This impending powerwall will place severe restrictions on the future progress and applications of CMOS technology. In the last couple of decades new technology fields have emerged which require computational power while maintaining strict constrols on power consumption. The proliferation of mobile devices, the emergence of the Internet of Things (IoT), and the need for large data centers to power the cloud are just a few of the fields which place a large emphasis on reducing power consumption. The physical limitations of CMOS technology indicate that future computing paradigms must move beyond CMOS if they are to continue the trend in increasing computational power first established by Moore's Law. Research into moving beyond CMOS can be divided into three main categories. The first category covers with the creation

of new devices whether they be new transistor designs or new materials. The second category covers the creation of new architectures which could have applications in existing CMOS technology or in newly proposed devices. The final category covers the introduction of completely new computing paradigms that are not based around digital logic. These three categories are not mutually exclusive. Proposed approaches to maintain the scaling of technology can fall under multiple categories. Figure 1.1 from [1] (© 2015 IEEE) shows just a few of the approaches that have been proposed. Each axis in the figure corresponds with one of the previously mentioned categories.

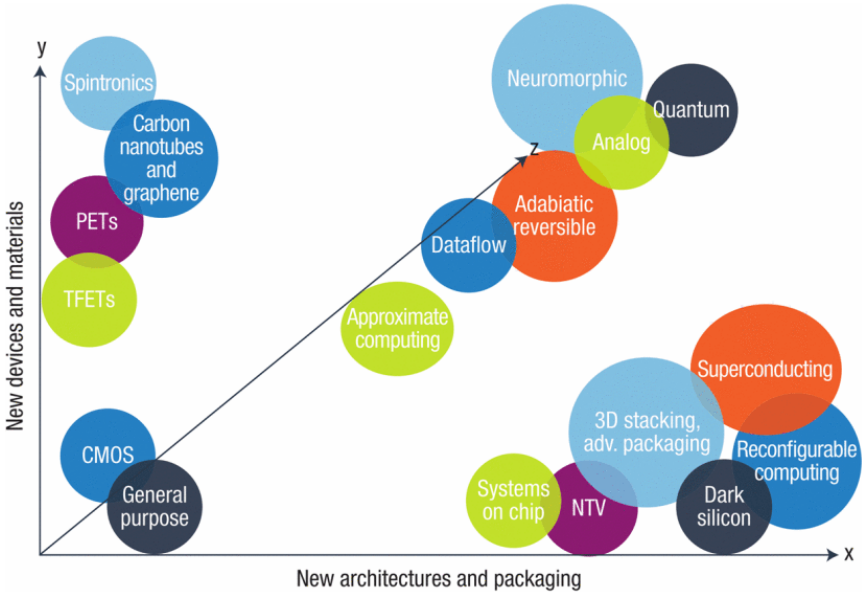


Figure 1.1: Approaches for maintaining technology scaling by category [1] (© 2015 IEEE).

The focus of this thesis is on emerging nanotechnologies which would fall under the category of new devices. Emerging nanotechnologies offer an intriguing alternative as they are low power in nature. These emerging nanotechnologies tend to be majority logic based. One such emerging nanotechnology is quantum dot cellular automata (QCA)[3]. QCA uses field coupled cells to allow the propagation of electrical signals through a circuit. The signal propagation through magnetic forces is what causes QCA to be considered low power when compared to conventional CMOS circuits.

A second emerging nanotechnology is nanomagnetic logic(NML). NML shares some similarities with QCA, such as also being majority logic based. NML makes use of nanomagnetic cells whose polarities correspond with logic states. Arrays of cells are used to form wires where the signals are able to propagate due to magnetic forces. Again, this signal propagation property is what makes NML low power in nature. More in depth information about these technologies will be presented in future chapters. Emerging technologies such as NML and QCA are majority logic based and therefore current CMOS designs are not easily reproducible in these technologies. The basic gates available in these technologies ensures that a straight imitation of CMOS gates would be inefficient. For that reason completely new design methodologies must be used to create familiar functional units that are used in CMOS technology. This thesis explores designs of some basic arithmetic circuits for use in majority logic based technologies. While NML and QCA are provided as sample technologies for design implementations, the designs themselves can be expanded to other similar majority logic based emerging nanotechnologies.

1.1 Contribution of Thesis

This thesis presents arithmetic logic circuit designs for majority logic based technologies. Specific attention is given to quantum dot cellular automata (QCA) and nanomagnetic logic (NML).

1. Full adder and ripple carry adder in QCA
2. Full subtractor and ripple borrow subtractor in QCA
3. A multilayer 5-input majority gate in NML
4. Multilayer full adder in NML
5. Multilayer full subtractor in NML

6. Two designs of testable reversible adders

1.2 Outline of Thesis

Chapter 2 provides an overview of QCA technology, NML technology, and reversible logic. Chapter 3 presents designs of a full adder, ripple carry adder, full subtractor, and ripple borrow subtractor implemented in QCA. Chapter 4 presents designs of a multilayer 5-input majority, full adder, and full subtractor in NML. Chapter 5 presents two designs for testable reversible adders in NML. Chapter 6 concludes the thesis. Portions of Chapters 2 and 3 were previously published in [4] (Reproduced by permission of the Institution of Engineering & Technology). Portions of Chapters 2 and 4 were previously published in [5] (Reproduced by permission of the Institution of Engineering & Technology). Portions of Chapters 2 and 5 were previously published in [6] ().

Chapter 2

Background

This chapter will cover any background information needed to understand the successive chapters. The main focus will be on the operation of Quantum Dot Cellular Automata (QCA) circuits, the operation of Nanomagnetic Logic (NML) circuits, and a description of conservative reversible logic.

2.1 Quantum Dot Cellular Automata

Quantum dot cellular automata (QCA) is an emerging field-coupled nanotechnology that is made of cells containing electrons[3]. The position of the electrons determines the logic state. Figure 2.1(a) shows the possible electron locations and Figure 2.1(b) the cell states corresponding to logic 0 and logic 1. Cells in QCA can have their electron locations rotated by 45°. These rotated cells are shown in Figure 2.2. Despite the change in electron location, rotated cells have the same operation as normal cells.

Wires are created by placing a series of cells side by side. Signals in wires formed from normal cells remain constant while wires created from rotated cells will have the signal invert with each successive cell. These qca wire properties are best demonstrated by the crosswire shown in Figure 2.3. Note that the signals are allowed to cross without any interference.

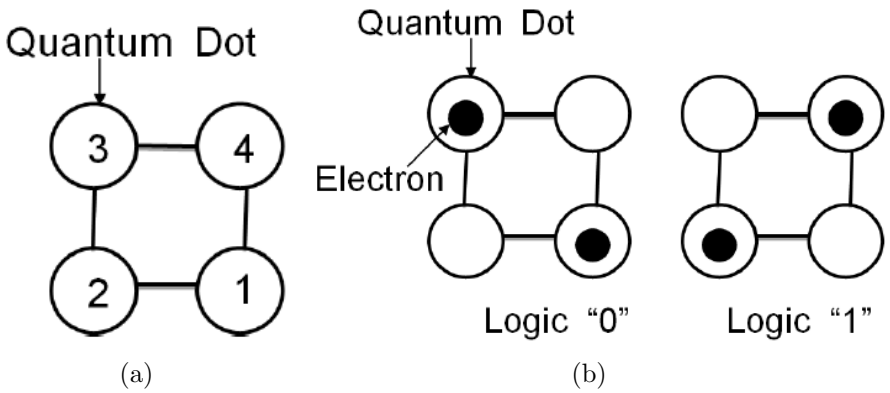


Figure 2.1: QCA Logic
(a) QCA 4 Dots
(b) QCA cell working as logic '0' and logic '1'

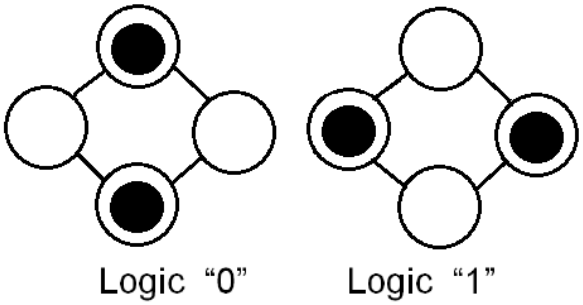


Figure 2.2: Rotated QCA Logic Cell

A four-phase clocking system is used to help facilitate the propagation of data through the logic circuits. The four phases are switch, hold, release, and relax. Cells in the switch phase are able to transition to their appropriate logic value. Cells in the hold phase maintain their current logic value. Cells in the release phase are able to transition back to a neutral state. Cells in the relax phase stay in their neutral state. The color of a cell corresponds to which clock zone a cell is in. The various types of cells can be seen in Figure 2.4 and the direction of propagation of data through these clock zones is shown in Figure 2.5.

The basic logic gates of QCA are the inverter and the 3-input majority gate. The

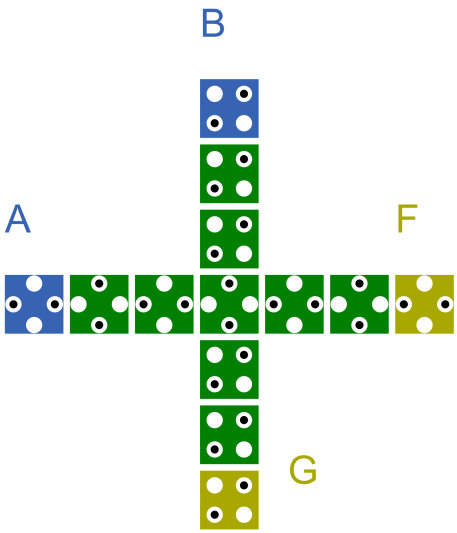


Figure 2.3: QCA Crosswire

output of a majority gate will be equivalent to the majority of its inputs. Figure 2.6 shows the QCA implementation of a 3-input majority gate. If the inputs to the majority gate are defined as A, B, C, then the equation for its output F can be expressed by the following equation:

$$F = MAJ_3(A, B, C) \tag{2.1}$$

$$F = AB + BC + AC \tag{2.2}$$

It is possible to create a larger majority gate which has 5 inputs. This larger majority gate can be seen in Figure 2.7. The 5-input majority gate works in the same manner as the 3-input majority gate. If you define the inputs as A, B, C, D, E, then

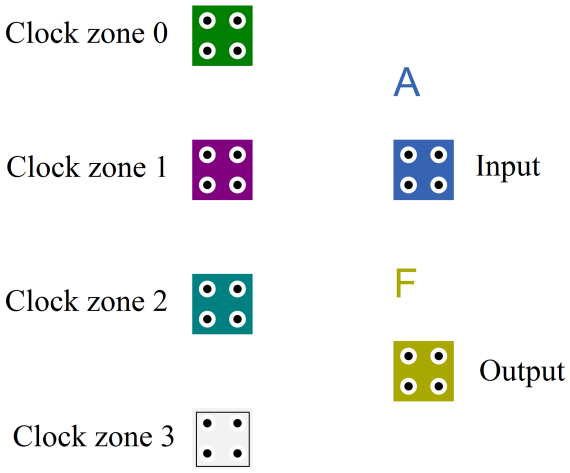


Figure 2.4: QCA Cell Code

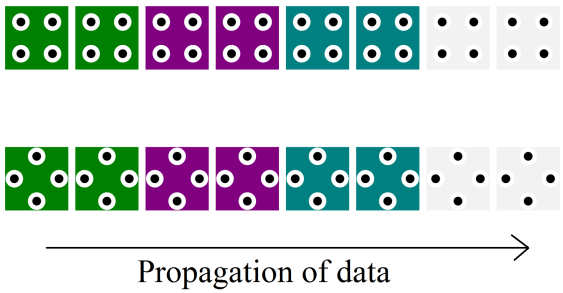


Figure 2.5: QCA Data Propagation

the equation for its output F can be expressed by the following equation:

$$F = MAJ_5(A, B, C, D, E) \tag{2.3}$$

$$F = ABC + ABD + ABE + ACD + ACE \tag{2.4}$$

$$+ ADE + BCD + BCE + BDE + CDE \tag{2.5}$$

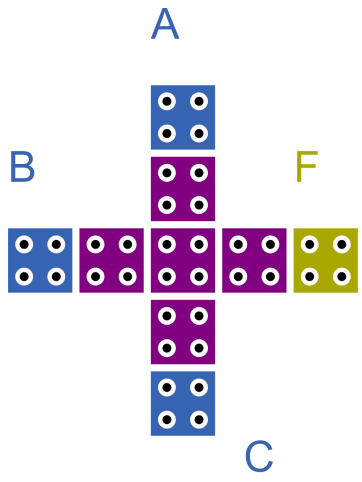


Figure 2.6: 3-Input Majority Gate

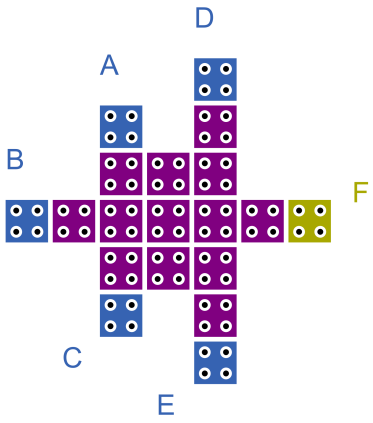


Figure 2.7: 5-Input Majority Gate

2.2 Nanomagnetic Logic

Nanomagnetic logic (NML) is based around nano-scale magnetic cells [7]. The magnetic polarization of each cell determines if the cell can be considered logic 0 or logic 1. Polarized up is considered logic 1 and polarized down is considered logic 0. Figure 2.8 shows an example of these polarizations with their corresponding logic values.

Wires in NML can be created by cascading multiple cells. The magnetic properties of the cells actually allow for two different types of wires. Placing cells end to end will

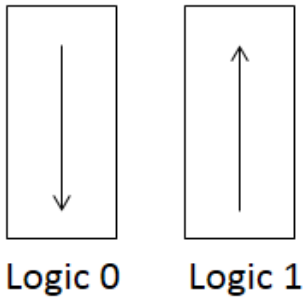
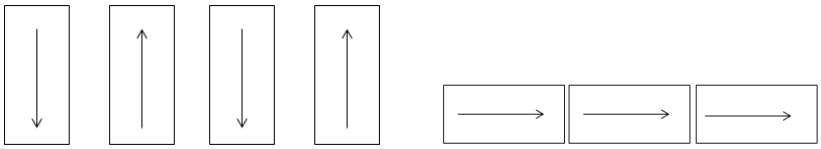


Figure 2.8: Logic Values in NML Computing

result in the signal remaining constant as it travels through the cells. Wires created by placing side by side will actually have the signal invert as it propagates through each successive cell. These two types of NML cell alignment are shown in Figure 2.9(a) and Figure 2.9(b), respectively.



(a) Side-by-side (b) End-to-end

Figure 2.9: NML Cell Alignment

The main gates in NML are the majority voter and the inverter. Inverters are created by simply placing cells side by side as their magnetic properties will cause the signal to invert. The majority voter is also known as a 3-input majority gate. This gate is represented by the equation:

$$F = MAJ_3(A, B, C) \tag{2.6}$$

$$F = AB + BC + AC \tag{2.7}$$

where A, B, and C are inputs, and F is the output. This simply means that the output F will have the same logic value as the majority of the inputs. Figure 2.10 shows an example implementation of a 3-input majority gate. In the example, Inputs

A and B are logic 1 while Input C is logic 0. Output F has a value of logic 1 because a majority of the inputs are logic 1.

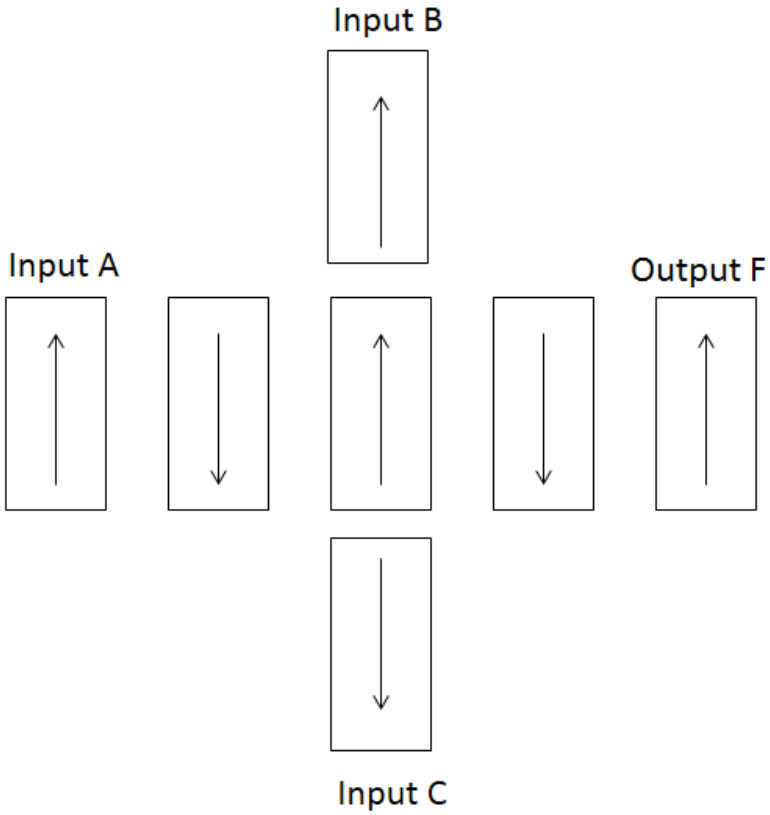


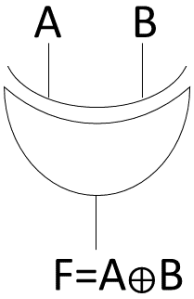
Figure 2.10: 3-Input Majority Gate

A three-phase clocking system is employed to control the propagation of information through NML circuits. The phases are RESET, SWITCH, and HOLD. Cells in the RESET phase remain in a neutral state that is neither logic 1 nor logic 0. Cells in the SWITCH phase are allowed to transition to the correct logic state. Cells in the HOLD phase maintain their current logic state for the duration of the phase. More information on the clocking system and NML computing can be found in [7],[8],[9],[10].

2.3 Reversible Logic

Logic gates are considered reversible when there is a one-to-one mapping between their inputs and their outputs. The one-to-one mapping means there are no repeated

output permutations. The fact that all output permutations are unique allows the associated input value to be determined for any given output value. As an example, consider a simple two input XOR gate as shown in Figure 2.11.



A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

Figure 2.11 & Table 2.1: XOR Gate

The truth table shows that it is impossible to determine what the input value was when given an output value. An output value of 0 could have been caused by an input of all 0's or of all 1's. Similarly, an output of 1 is caused by an input combination of 1 and 0. Which input is 1 and which input is 0 does not matter. It is possible to introduce an extra output to the XOR which allows it to maintain its normal behavior while also maintaining a one-to-one mapping between the inputs and outputs. This gate is called a Feynman gate and is shown in Figure 2.12.

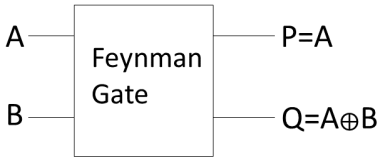


Figure 2.12: Feynman Gate

Table 2.2: Feynman Gate Truth Table

A	B	P	Q
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Output Q is equivalent to the output of a standard XOR gate. Output P allows one to differentiate between the input combinations which would result in an output of 1 in a standard XOR gate and thereby implements the previously described one-to-one mapping. This property can been seen in the associated truth table in Table 2.2 and shows that the gate is reversible. All of the outputs are unique and as a result the input value can be immediately determined for any given output value.

2.3.1 Conservative Reversible Logic

Logic gates are considered conservative reversible (CR) when they have a one-to-one mapping between their inputs and their outputs and also have the same number of 1's in their output as they have in their input. Functional units constructed from CR logic gates will maintain their conservative reversible property. Spintronics based nanomagnetic logic (NML) computing is a promising platform to implement these types of circuits. There are high error rates associated with nanoscale manufacturing. For that reason it is important to place a focus on reducing device error rates. CR logic gates can be easily tested for unidirectional stuck at faults. This is accomplished by comparing the number of 1's in the input vectors and output vectors. The CR nature of the gate guarantees they should be equal. Offline test vectors of all 0's and all 1's would be the only test vectors required to detect unidirectional stuck at faults. The comprehensive proof of this property can be referred to in [11–13]. This paper proposes two methodologies for the design of ripple carry adders that only require test vectors of all 0's and all 1's to detect all unidirectional stuck at faults.

2.3.2 Conservative Reversible Fredkin Gate

One type of conservative reversible logic gate is the Fredkin Gate [14] which is shown in Figure 2.13. Define the inputs as A, B, C and the outputs as P, Q, R. The outputs can be expressed by the following expressions:

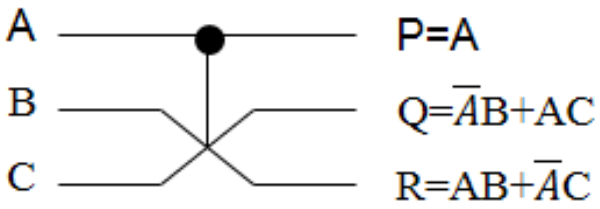


Figure 2.13: Fredkin Gate

$$P = A \tag{2.8}$$

$$Q = \overline{A}B + AC \tag{2.9}$$

$$R = AB + \overline{A}C \tag{2.10}$$

Table 2.3 contains the truth table for the Fredkin Gate.

Table 2.3: Fredkin Gate Truth Table

A	B	C	P	Q	R
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	1	1	1

The truth table proves the conservative reversible property of the Fredkin gate. Each output has the same number of 1's as its corresponding input and there is a one-to-one mapping between the inputs and the outputs. The benefit of this property is it allows stuck at faults to be easily detected. Stuck at 1 faults will be detected when an input vector of all 0's does not return an output vector of all 0's. Likewise, stuck at 0 faults will be detected when an input vector of all 1's does not return an

output vector of all 1's. Figure 2.14 shows how a Fredkin gate can be implemented in NML computing. The designs proposed in this chapter are constructed from Fredkin gates and thus have potential applications in NML computing.

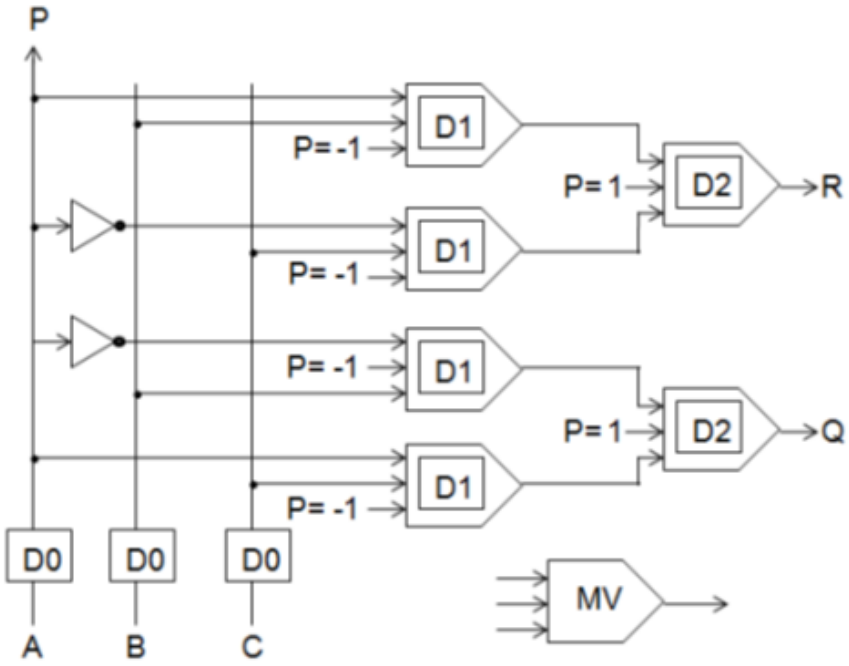


Figure 2.14: NML Implementation of Fredkin Gate

Chapter 3

Design of Adder and Subtractor Circuits in QCA

The implementation of arithmetic logic circuits using QCA is explored in works such as [15], [16], and [17]. Some existing designs make use of multiple layers to boost performance. A method to implement single layer QCA designs as multilayer has been described in [18]. Questions still remain about the feasibility of being able to manufacture multilayer designs. This fact leaves open the possibility of many designs being rendered invalid. The designs we are proposing are able to side-step this issue by being restricted to a single layer. The designs we are proposing are better than existing single layer designs. In addition, our proposed designs are competitive with and in most cases better than existing multilayer designs. Our proposed designs make improvements in the number of cells, circuit area, and the latency of the circuit. The designs proposed in this chapter consist of a full adder, a 4-bit ripple carry adder, a full subtractor, and a 4-bit ripple borrow subtractor. QCADesigner was used to implement and verify all of the proposed designs. The simulation results were only included for the full adder and full subtractor as the simulation results for the 4-bit units were much too large to include. All simulations were performed in QCADesigner

used the settings outlined in Table 3.1.

Table 3.1: QCADesigner Simulation Settings

Setting	Value
Number of Samples	12800
Cell Size	18.0 nm x 18.0 nm
Dot Diameter	5.0 nm x 5.0 nm
Distance Between Cells	2.0 nm
Radius of Effect (nm)	41.0
Relative Permittivity	12.9
Clock High	9.8e-22
Clock Low	3.8e-23
Clock Shift	0.0
Clock Amplitude Factor	2.0
Layer Separation	11.5
Maximum Iterations Per Sample	100

3.1 Design of Proposed Full Adder

Define the inputs as A, B, C_{in} where A and B are the values being added and C_{in} is the input carry value. Define the outputs as C_{out}, Sum where Sum is the sum of the three inputs and C_{out} is the output carry. The equations for the full adder outputs can be expressed as follows:

$$C_{out} = AB + BC_{in} + AC_{in} \tag{3.1}$$

$$Sum = A \oplus B \oplus C_{in} \tag{3.2}$$

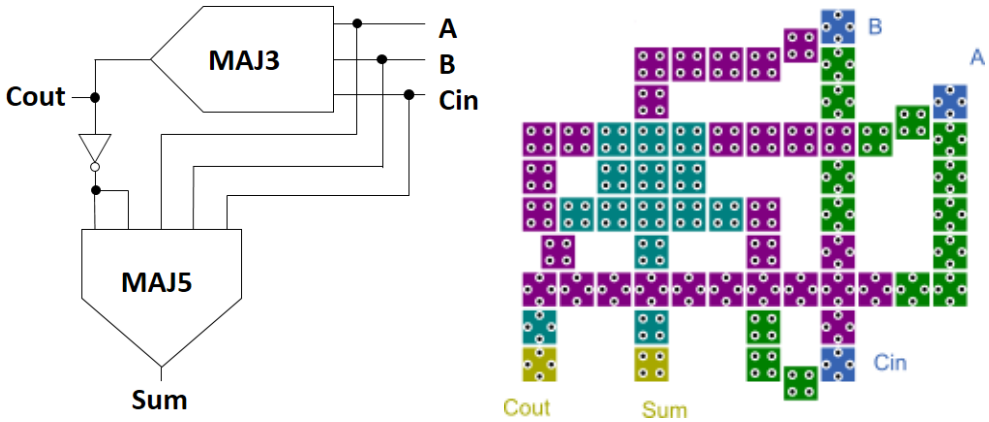
By making use of majority gates, the C_{out} equation can be rewritten as:

$$C_{out} = MAJ3(A, B, C_{in}) \tag{3.3}$$

where $MAJ3$ denotes a 3-input majority gate. The equation for Sum can also be rewritten by using a 5-input majority gate (MAJ5) with the value of the output C_{out} also being used as an input. The equation for Sum then becomes:

$$Sum = MAJ5(A, B, C_{in}, \overline{C_{out}}, \overline{C_{out}}) \tag{3.4}$$

Figure 3.1(a) is the circuit diagram of the proposed full adder. The use of a 5-input majority gate results in a circuit that is much simpler than designs that use only 3-input majority gates and inverters. Figure 3.1(b) contains the QCA implementation of the full adder. A simulation of the circuit using QCADesigner was used to verify the correct function of the proposed full adder. The results of this simulation can be seen in Figure 3.2.



(a) (b)

Figure 3.1: Proposed Full Adder
(a) Full Adder Circuit
(b) Full Adder QCA Implementation

Table 3.2 contains a comparison between our proposed full design and some existing designs. The cost value was determined by the following equation:

$$Cost = Area * Latency^2 \tag{3.5}$$

where $Area$ is the size of the design in $um\hat{}2$ and $Latency$ is the number of clock cycles. Our proposed design is better in all categories than the single layer design from [21].

Table 3.2: QCA Values for Proposed Full Adder

Design	Cells	Area ($um\hat{}2$)	Latency	Cost
[19]*	86	0.10	0.75	0.056
[20]*	73	0.04	0.75	0.0225
[21]	69	0.07	1	0.07
Proposed	63	0.05	0.75	0.028

*Multilayer design

Our proposed design also competitive with multilayer designs. Our proposed design requires fewer cells than [20] and has a lower cost than [19]. It should be noted than better designs exist in [15] and [22]. Both of the full adder designs from those works were omitted due to being multilayer designs.

3.2 Design of Proposed Ripple Carry Adder

The proposed design of the ripple carry adder is very straightforward. It is simply four full adders connected in series so that the carry out C_{out} from one adder gets passed into the carry in C_{in} of the next adder. Define the input values as A_3, A_2, A_1, A_0 and B_3, B_2, B_1, B_0 and define their sum as S_3, S_2, S_1, S_0. Figure 3.3 contains the QCA implementation of a 4-bit ripple carry adder.

Table 3.3 contains a comparison between our proposed 4-bit ripple carry adder and existing designs. Our proposed design has the lowest cell count and cost of the designs compared. [23] has a lower area than our proposed design. The design we proposed is able to counteract that by having a lower latency.

Table 3.3: QCA Values for Proposed 4-bit Ripple Carry Adder

Design	Cells	Area ($um\hat{}2$)	Latency	Cost
[24]*	651	1.20	4.25	21.68
[19]*	371	0.40	1.50	0.911
[23]*	339	0.25	1.75	0.766
Proposed	295	0.30	1.50	0.675

*Multilayer design

3.3 Design of Proposed Full Subtractor

Define the inputs as X, Y, Z where Y is the value being subtracted from X and Z is the borrow input. Define the outputs as $Diff, B$ where $Diff$ is the difference of $X - Y - Z$ and B is the borrow output. The equations for the full subtractor outputs can be expressed as follows:

$$Diff = X \oplus Y \oplus Z \tag{3.6}$$

$$B = \overline{X}(Y + Z) + YZ \tag{3.7}$$

The equation for B can be expanded into:

$$B = \overline{X}Y + \overline{X}Z + YZ \tag{3.8}$$

Doing so makes it possible to express B as the output of a single 3-input majority gate.

$$B = MAJ3(\overline{X}, Y, Z) \tag{3.9}$$

By using the same process as for the proposed full adder, the equation for $Diff$ can be rewritten with a 5-input majority gate (MAJ5). In this case the value of the

output B is used as an input.

$$Diff = MAJ5(X, \overline{Y}, \overline{Z}, B, B) \tag{3.10}$$

Figure 3.4(a) is the circuit diagram of the proposed full subtractor. The use of a 5-input majority gate results in a circuit that is much simpler than designs that use only 3-input majority gates and inverters. Figure 3.4(b) contains the QCA implementation of the proposed full subtractor design. A simulation of the circuit using QCADesigner was used to verify the correct function of the proposed full subtractor. The results of this simulation can be seen in Figure 3.2.

Table 3.4 contains a comparison between our proposed full subtractor design and existing designs. Our proposed design is superior in all categories compared to the existing designs.

Table 3.4: QCA Values for Proposed Full Subtractor

Design	Cells	Area ($um\hat{}2$)	Latency	Cost
[25]*	186	0.132	2	0.528
[16]	90	0.11	1	0.22
Proposed	63	0.05	0.75	0.028

*Multilayer design

3.4 Design of Proposed Ripple Borrow Subtractor

It is possible to cascade n copies of the proposed full subtractor to create an n-bit ripple borrow subtractor. This is done by using the borrow output B from one full subtractor as the barrow input Z of the next full subtractor. By that method a 4-bit ripple borrow subtractor can be constructed with the inputs defined as X_3, X_2, X_1, X_0 and Y_3, Y_2, Y_1, Y_0 and the borrow input defined as Z. Their difference can defined as $Diff_3, Diff_2, Diff_1, Diff_0$ and the borrow input can be defined as B. The design

of the proposed 4-bit ripple borrow subtractor is shown in Figure 3.6. Table 3.5 contains a comparison between our proposed 4-bit ripple borrow subtractor design and existing designs. Our proposed design is better in all categories than the single layer design presented in [16]. The multilayer design from [26] has the same latency, but is otherwise worse in all categories than our proposed design.

Table 3.5: QCA Values for Proposed 4-bit Ripple Borrow Subtractor

Design	Cells	Area ($um\hat{}2$)	Latency	Cost
[16]	N/A	0.688	4	11.008
[26]*	410	0.43	1.5	0.968
Proposed	295	0.38	1.5	0.855

*Multilayer design

3.5 Conclusion

In this chapter we have proposed single-layer QCA designs for some basic arithmetic circuits using 5-input majority gates. Our proposed designs for the full adder, 4-bit ripple carry adder, full subtractor, and 4-bit ripple borrow subtractor are better than existing single layer designs according to our evaluation metrics. In addition, our proposed designs are competitive with existing multilayer designs and are actually better in some cases.

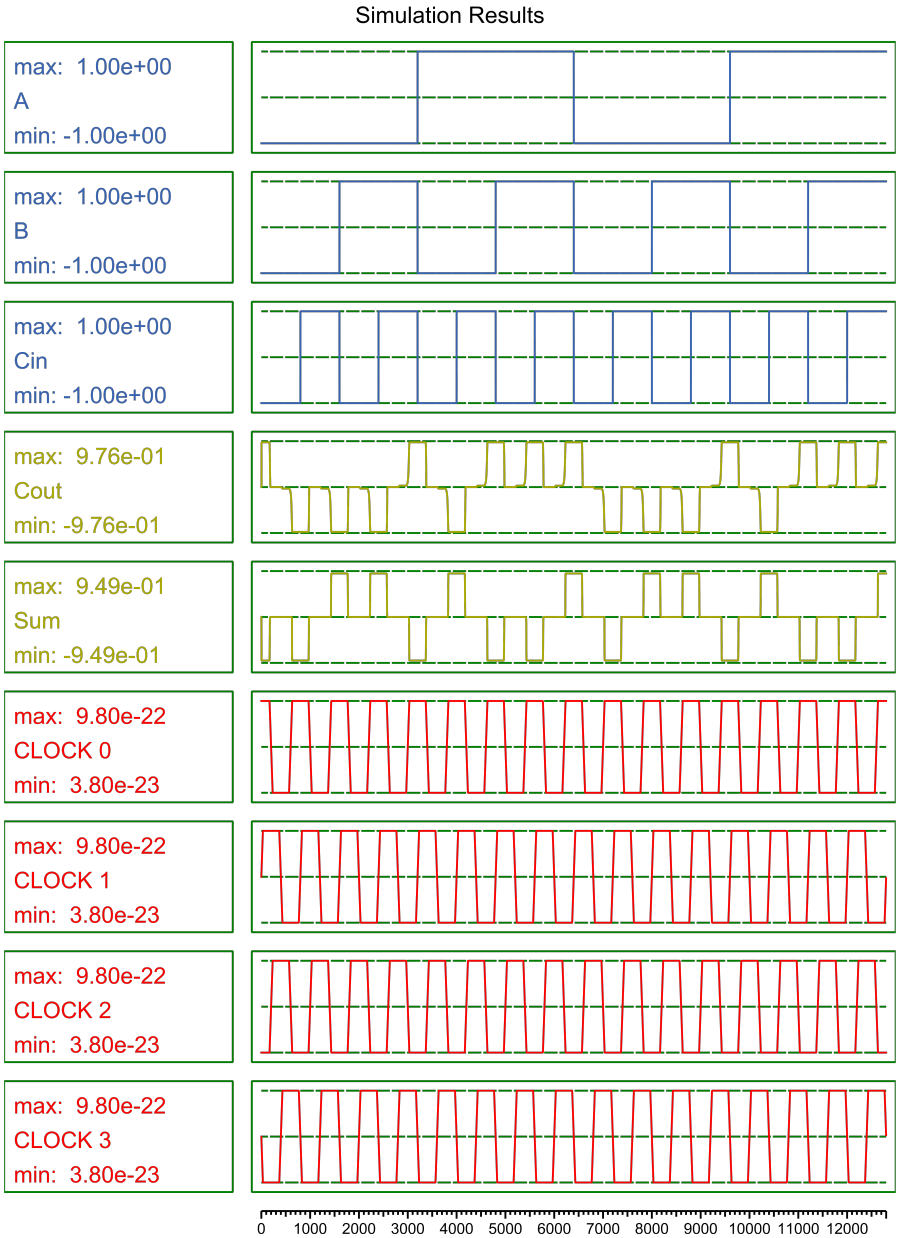


Figure 3.2: Simulation of Proposed QCA Full Adder

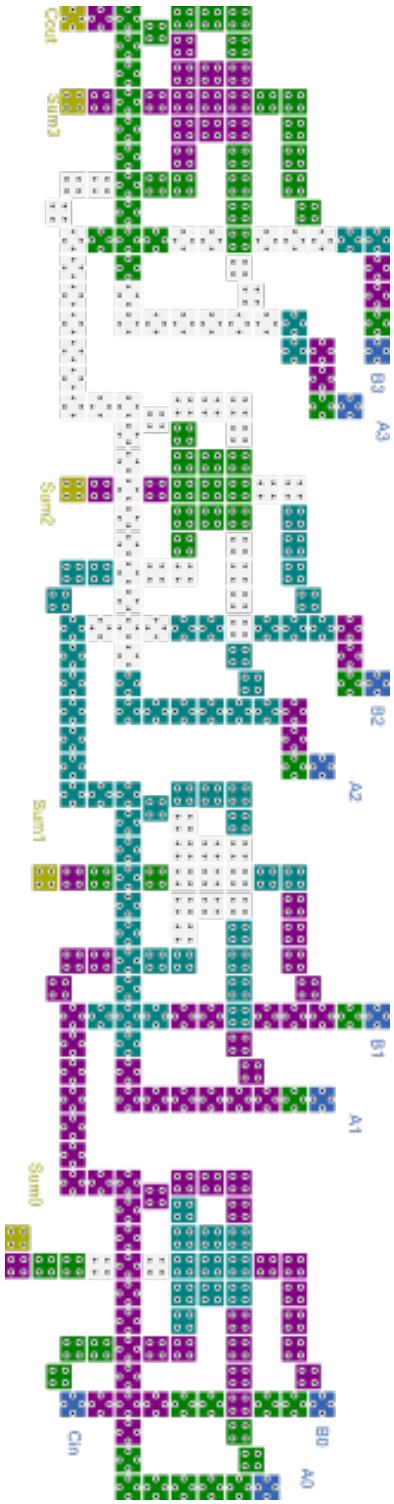
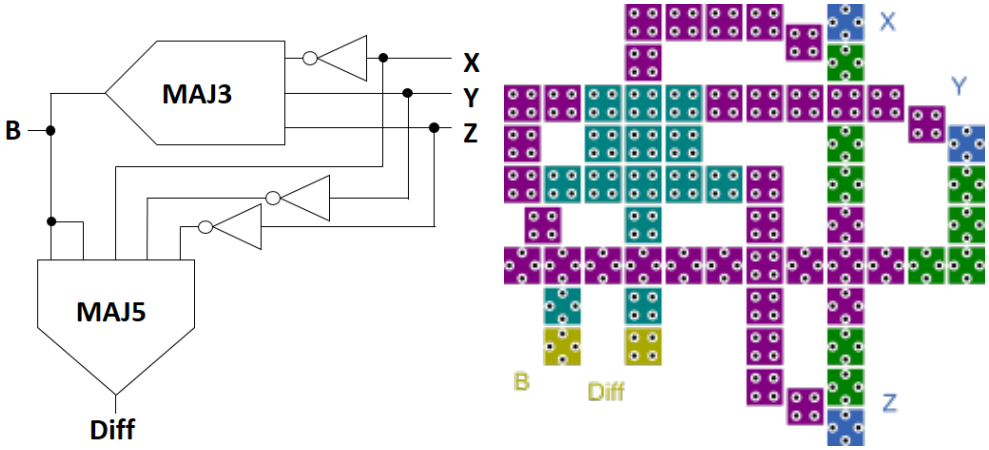


Figure 3.3: Proposed 4-bit Ripple Carry Adder QCA Implementation



(a) (b)

Figure 3.4: Proposed Full Subtractor
(a) Full Subtractor Circuit
(b) Full Subtractor QCA Implementation

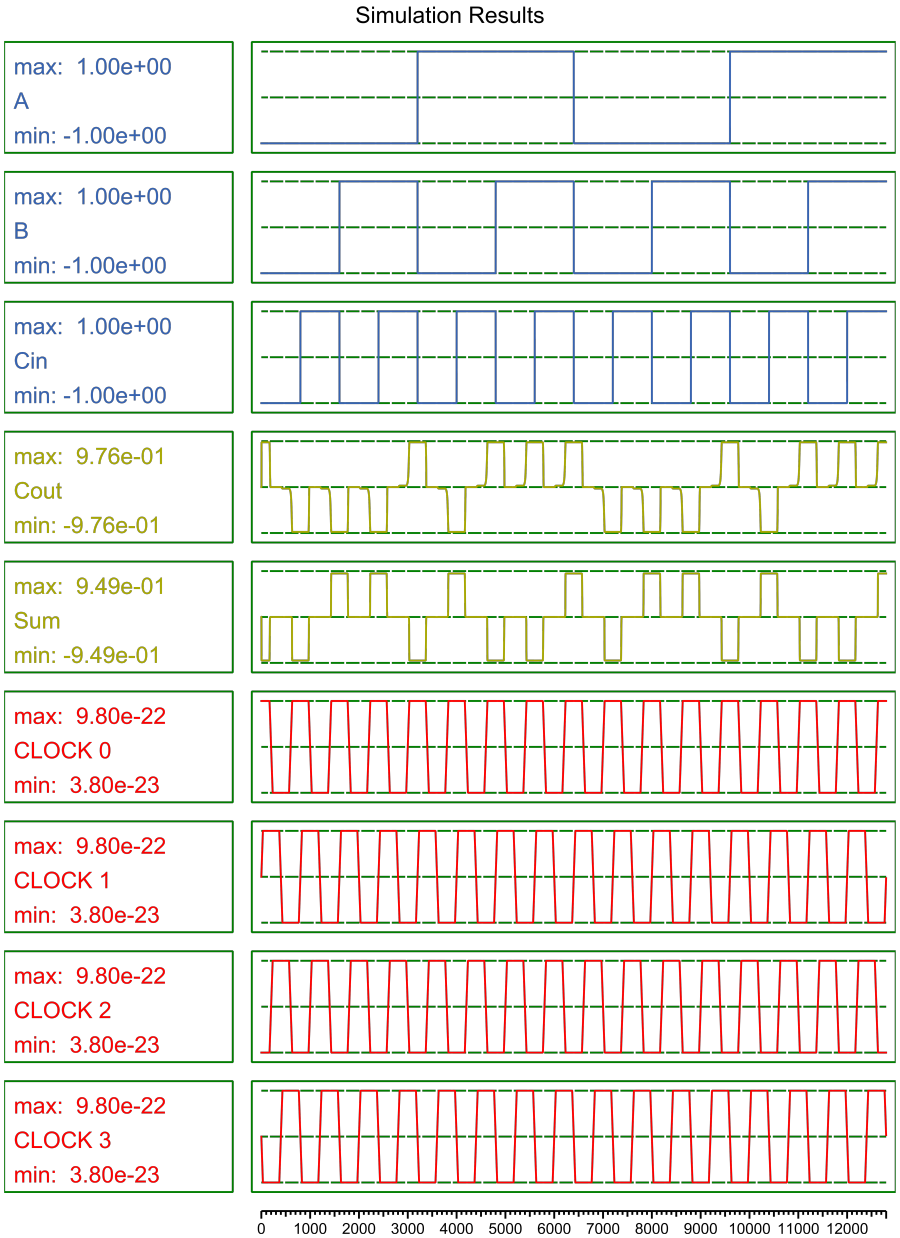


Figure 3.5: Simulation of Proposed QCA Full Subtractor

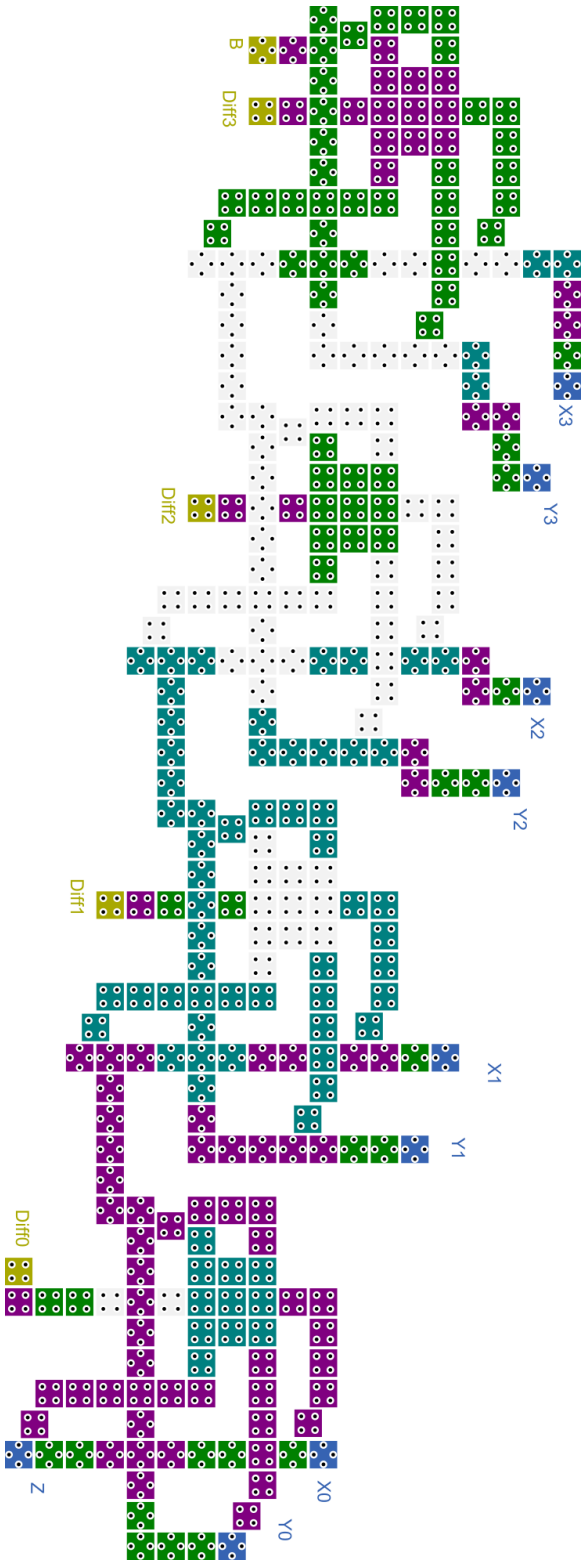


Figure 3.6: Proposed 4-bit Ripple Borrow Subtractor QCA Implementation

Chapter 4

Proposed 5-Input Majority Gate in NML Computing

Some researchers [27],[28],[29],[30] have proposed using multiple layers to create three dimensional NML circuits and structures. Specifically [29] and [30] have used the stacking of multiple layers of NML cells to create circuits. These designs utilize magnetic vias to allow signals to propagate between logic planes. The logic planes are essentially identical to normal single layer designs that contain logic gates. A single NML cell can be used as a via to allow signals to propagate between logic planes.

Researchers have already proposed some designs for full adders in NML. One such design [31] requires multiple copies of each input to ensure correct logic function of the circuit. A separate work [29] contains a full adder NML design that uses multiple layers to avoid the need for multiple copies of each inputs. By using our proposed 5-input majority gate we are able to improve on the number of cells in the design from [29] without requiring additional input copies such as in [31]. We wanted our designs to consist of uniform symmetric cells so other existing designs such as [32], [27], [28], etc., were not considered in our comparisons. To our knowledge, there are

no existing NML designs of a full subtractor.

4.1 Design Verification Framework

Verilog was used to verify the designs proposed in this chapter from a logic standpoint. An existing MQCA library was modified to allow inputs to any cell from above and below the cell in addition to the four standard cardinal directions. A module simulates the behavior of each individual cell in the circuit. The program OpenSCAD was used to generate 3D models of the circuits themselves to aid in verifying cell placements. A program was created to speed up the verification process. By doing so the process of simulating these multilayer circuits has been reduced to specifying the location of each cell. Simulation of the designs in Verilog was determined to be a better approach for our purposes instead of using an existing simulation tool such as OOMMF [33].

4.1.1 NML Generation Program

A program was created to aide in the simulation of our proposed NML designs. The generation program was written in C using the tools lex and yacc to create a lexical analyzer and parser. The program takes an input file of cell locations and generates equivalent Verilog and OpenSCAD descriptions. The program itself was not designed to be robust. It does not perform any error checking of the circuit layout, nor does the program check syntax. The program works under the assumption that the input file is completely valid. The cell locations can be visualized as X,Y,Z coordiantes on a grid. Inputformat.txt is a general example of the input file syntax expected by the program. The file must contain the following tags in order: module, parameters, inputs, cells, outputs, end.

module

A module name must be given after "module". This name will be the name of the created Verilog module in addition to the actual Verilog and OpenScad files. For that reason the given module name must be a valid option across each format.

parameters

After "parameters", three numbers are used to specify the number of inputs, the number of outputs, and the number of layers in the desired NML circuit. As an example, (3,1,3) would denote a circuit with three inputs, one output, and three layers of NML cells.

inputs

After "inputs", each individual circuit input must be specified by its name and position. The input name must be a valid input name in Verilog. The position is denoted by X, Y, and Z coordinates. The range for each coordinate is any whole number from 1 to 127. All inputs are assumed to be in clock zone 0.

cells

After "cells", each individual cell must be listed by its X, Y, and Z coordinates and the clock zone it is in. Each entry can be further modified to denote the center cell in a majority gate or the location of an inverter. The use of "m" specifies a majority gate and "inv" specifies an inverter. The inverter actually fits four cells into the space normally occupied by three cells. For that reason there must not be a cell located on either side of the inverter along its X axis.

outputs

After "outputs", each individual circuit output must be specified by its name, position, and clock zone. The output name must be a valid output name in Verilog. The position is denoted by X, Y, and Z coordinates. Specifying the outputs is very similar to the method used to specify the inputs. The only difference is it possible to place an output in any clock zone.

end

The presence of "end" denotes the end of the file.

Inputformat

```
module

module_name


parameters

(number_of_inputs,number_of_outputs,number_of_layers)


//X,Y,Z coordinate range is 1-127

inputs

(Name,X,Y,Z)

(Name,X,Y,Z)

(Name,X,Y,Z)


cells

(X,Y,Z,clock_zone)


//Majority gate

m(X,Y,Z,clock_zone)
```

```
//Inverter
//No other cells at X-1 and X+1
inv(X,Y,Z,clock_zone)

outputs
(Name,X,Y,Z,clock_zone)
(Name,X,Y,Z,clock_zone)

end
```

Each cell in the OpenSCAD file is color coded to differentiate between inputs, outputs, and standard cells. Table 4.1 contains complete explanation of the different colors.

Table 4.1: 3D Cell Model Color Codes

Cell Color	Cell Type
White	Input
Black	Output
Blue	Clock Zone 0
Red	Clock Zone 1
Green	Clock Zone 2

4.2 Proposed 5-Input Majority Gate

A newer type of majority gate is the 5-input majority gate. It functions in much the same way as a standard 3-input majority gate. The main difference between the gates is there are now five inputs instead of three inputs. A 5-input majority gate

can be represented by the equation:

$$F = MAJ_5(A, B, C, D, E) \tag{4.1}$$

$$F = ABC + ABD + ABE + ACD + ACE \tag{4.2}$$

$$+ ADE + BCD + BCE + BDE + CDE \tag{4.3}$$

where A, B, C, D, and E are inputs and F is the output. The general shape and magnetic properties of NML cells make it difficult to efficiently implement a majority gate that has more than 3 inputs. One possible design for a 5-input majority gate has already been proposed [27]. That design would require the inputs to be specially designed and fabricated on a circuit-by-circuit basis. Our proposed design uses multiple layers to eliminate the need for special cells. We are able to achieve this goal by treating the space above and below a given cell as possible inputs and outputs. In total there are 5 layers where layers 1, 3, and 5 contain inputs and outputs, and layers 2 and 4 are vias connecting the inputs and outputs on layers 1 and 5. Figure 4.1 shows a 3D model of this implementation in which the output is located on layer 3.

4.2.1 Design Verification

An exhaustive test was performed on the equivalent Verilog description of the proposed 5-input majority gate. The simulation results are shown in Figure 4.2. The sections of blue in the waveform for Output F denote high impedance. High impedance occurs when a cell is in RESET phase. The slight delay in the value of F is due to the fact that all of the inputs are in clock zone 0 while the output is in clock zone 1. As the signal propagates through the circuit, F first transitions into the RESET phase before then assuming its proper value. F only shows logic 1 when at least three of the inputs are also logic 1. The simulation verified that the 5-input majority gate

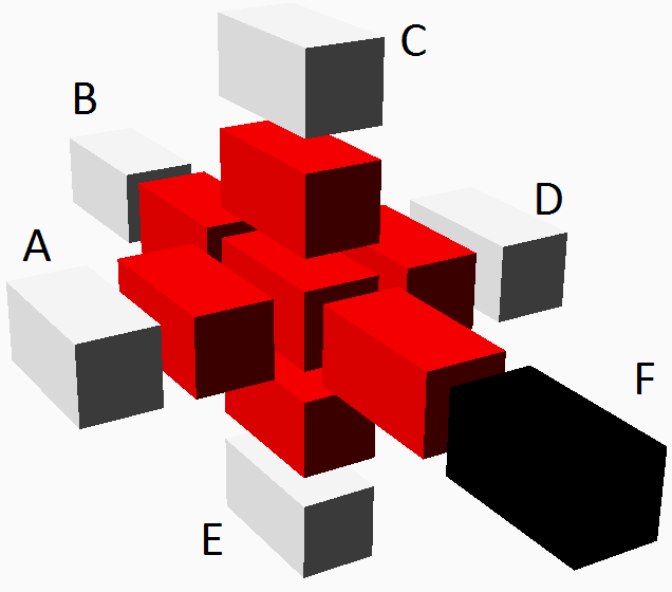


Figure 4.1: 3D 5-Input Majority Gate

follows the correct behavior outlined in the previous equations.

4.3 Design of Proposed Full Adder

The newly proposed 5-input majority gate can be used to construct multilayer circuits. As an example we have presented the design of a full adder. Define the inputs as A, B, and C_{in} where C_{in} is the input carry and A and B are the values to be added. The outputs can be defined as C_{out} and Sum where C_{out} is the carry output and Sum is the Sum of the inputs. The outputs of a standard full adder are expressed by the following equations:

$$C_{out} = AB + BC_{in} + AC_{in} \tag{4.4}$$

$$Sum = A \oplus B \oplus C_{in} \tag{4.5}$$

These equations can be rewritten to make use of the majority gates. Doing so

allows the equation for C_{out} to be rewritten as:

$$C_{out} = MAJ_3(A, B, C_{in}) \tag{4.6}$$

The equation for Sum can be rewritten to make use of the 5-input majority gate. Doing so requires the value of C_{out} to be used as an input to the gate. The new equation for Sum becomes:

$$Sum = MAJ_5(A, B, C, \overline{C_{out}}, \overline{C_{out}}) \tag{4.7}$$

A 3D representation of the full adder can be seen in Figure 4.3.

4.3.1 Design Verification

An exhaustive test was performed on the equivalent Verilog description of the proposed full adder. The simulation results are shown in Figure 4.4. Just like with the simulation for the 5-input majority gate, there is a delay between a change in the inputs being reflected in the outputs. The inputs A, B, and C_{in} are in clock zone 0 while the outputs C_{out} and Sum are in clock zone 2. The simulation verified that the proposed full adder operated correctly.

4.4 Proposed Full Adder Design Comparisons

Table 4.2 contains a comparison of our proposed design with some existing works. Percent difference was used to compare the existing designs to our proposed design. We did not calculate a percent difference in the number of layers used in our proposed design and the design by [31] due to their original design being only a single layer. By using multiple layers, our proposed design is able to avoid the need for extra inputs such as required by the design proposed in [31]. The use of a 5-input majority gate in

Table 4.2: Proposed 3D Full Adder Comparison

Metric	[31]	[29]	Proposed	% Diff to [31]	% Diff to [29]
Inputs	6	3	3	66.7	0
Layers	1	3	5	-	50
Cells	31	90	66	72.2	30.8

our proposed design allows us to use fewer cells than the multilayer design proposed in [29].

4.5 Design of Proposed Full Subtractor

Define the inputs as X, Y, and Z where Z is the borrow input and Y is the value being subtracted from X. Define the outputs as B and $Diff$ where B is the borrow output and $Diff$ is the difference of $X - Y - Z$. The equations for the outputs can be expressed as follows:

$$B = \overline{X}Y + \overline{X}Z + YZ \tag{4.8}$$

$$Diff = X \oplus Y \oplus Z \tag{4.9}$$

Just like with the full adder, the output equations can be rewritten to make use of majority gates. Doing so produces the following equations:

$$B = MAJ_3(\overline{X}, Y, Z) \tag{4.10}$$

$$Diff = MAJ_5(X, \overline{Y}, \overline{Z}, B, B) \tag{4.11}$$

A 3D representation of the proposed full subtractor can be seen in Figure 4.5. The proposed design requires 76 cells and only one instance of each input.

4.5.1 Design Verification

An exhaustive test was performed on the equivalent Verilog description of the proposed full subtractor. The simulation results are shown in Figure 4.6. Just like with the previously presented simulations, there is a delay between a change in the inputs being reflected in the outputs. The inputs X, Y, and Z are in clock zone 0 while the outputs B and $Diff$ are in clock zone 2. The simulation verified that the proposed full adder operated correctly.

4.6 Conclusion

In this chapter we proposed a method for implementing a 5-input majority gate in NML computing. Further, we illustrated the utility of our proposed gate by showing how it could be used in the design of a full adder and a full subtractor in NML. Verilog was used to verify the correct operation of the proposd circuits. going forward, this proposed gate should allow for the implementation of less complex functional units in NML. The use of identical cells in the proposed five-input majority gate should somewhat simplify the design process for any functional units that use the gate. The proposed designs can be expanded to create multi-bit arithmetic circuits.

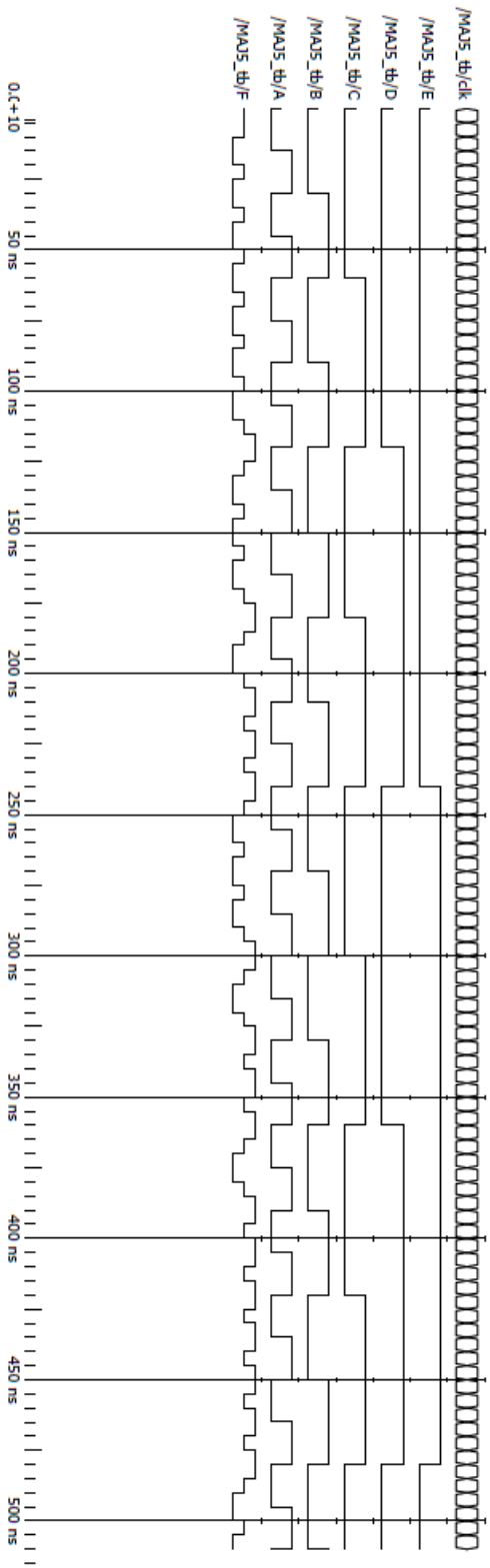


Figure 4.2: 5-Input Majority Gate Simulation

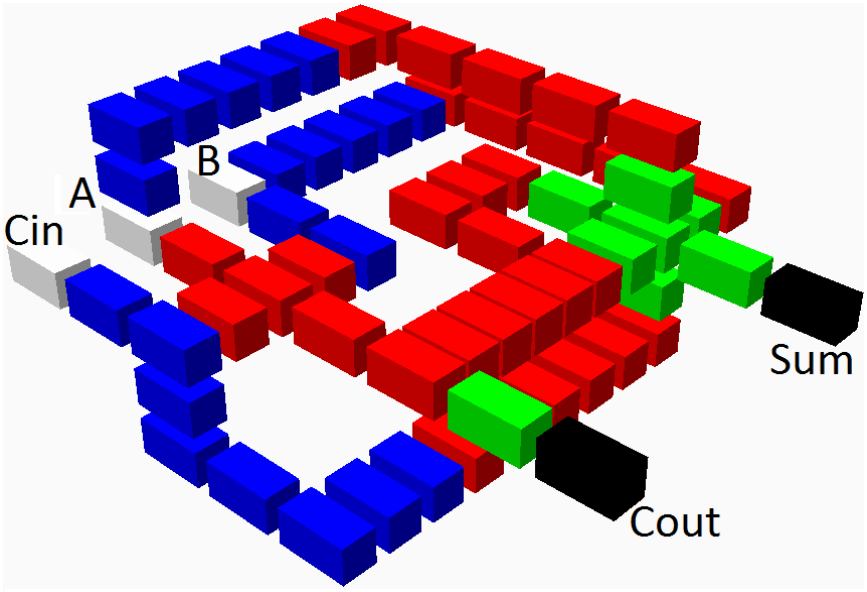


Figure 4.3: 3D Full Adder

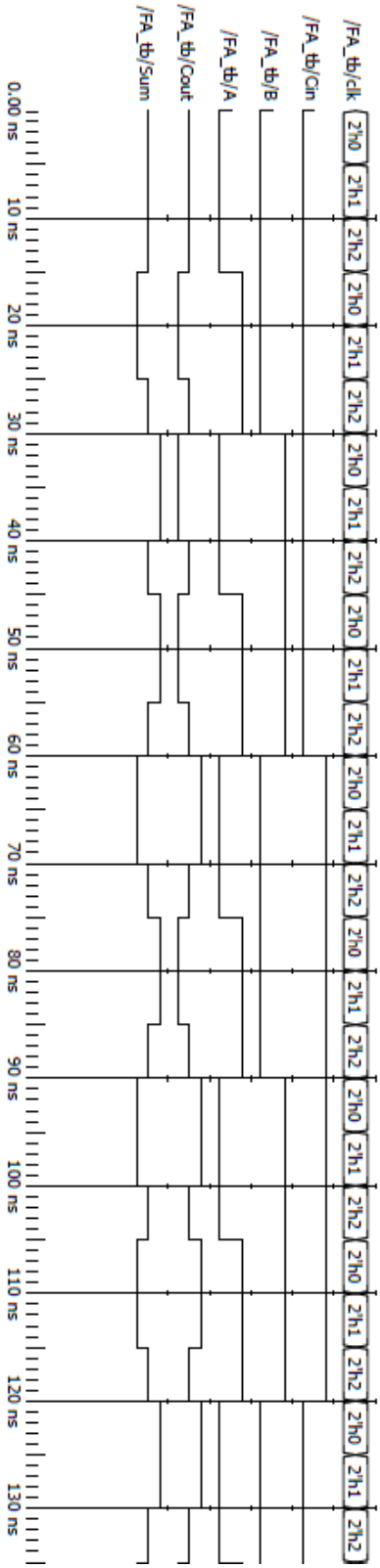


Figure 4.4: Full Adder Simulation

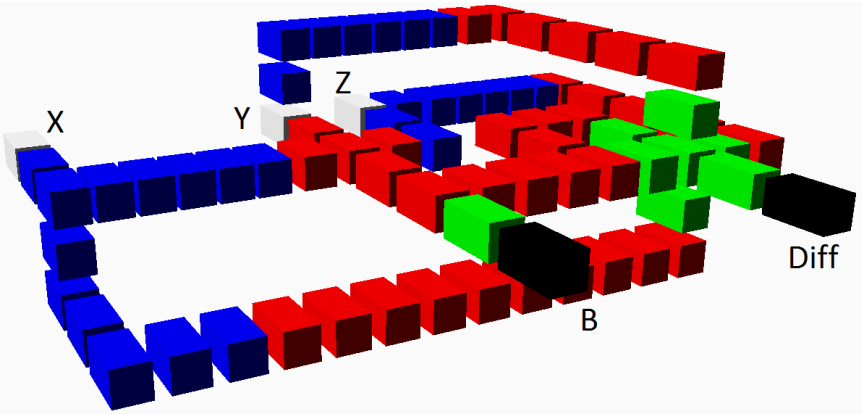


Figure 4.5: Proposed 3D Full Subtractor

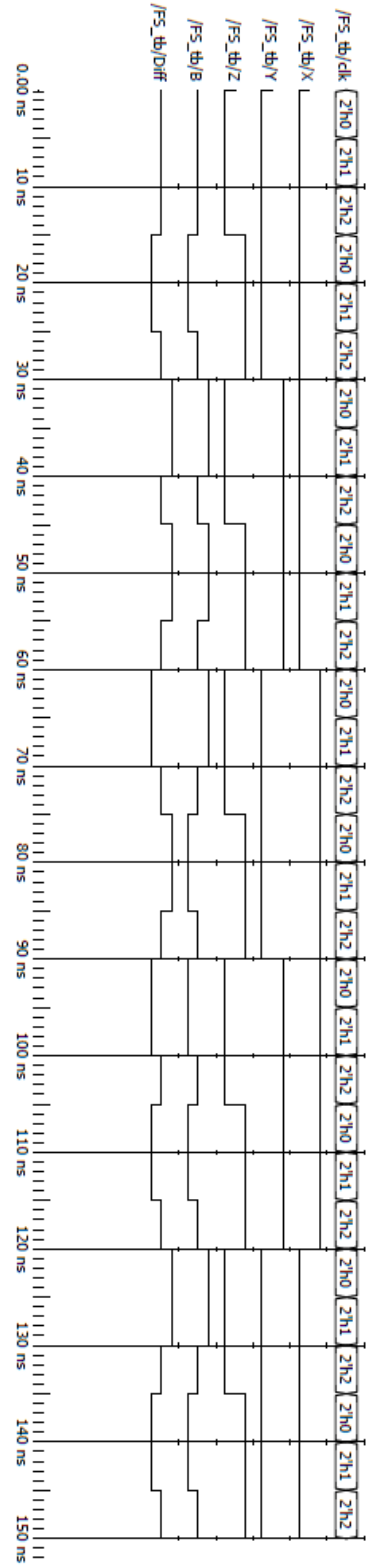


Figure 4.6: Full Subtractor Simulation

Chapter 5

Design of Testable Adder Circuits for NML Computing

This chapter will show how the principles of reversible logic can be applied to the design of adder circuits in NML computing. Two different design methodologies are used to create the proposed testable reversible adders.

5.1 Design Methodology 1 of Proposed Testable Reversible Ripple Carry Adder

This design methodology uses an approach similar to those of the designs proposed in [34, 35]. Implementing the design using the previously described Fredkin gate will allow the design to be applicable in NML computing. This design requires the creation of two smaller functional units. Cascading n copies of both functional units produces an n-bit adder. In this paper the units will be referred to as conservative reversible test block 1 (CRTB 1) and conservative reversible test block 2 (CRTB 2). The Fredkin gate based implementations of CRTB 1 and CRTB 2 are shown in Figures 5.1 and 5.2, respectively. The truth table of CRTB 1 is shown in 5.1 and the

truth table of CRTB 2 is shown in 5.2.

Table 5.1: CRTB 1 Truth Table

A	B	C	$A \oplus B$	$A\overline{B} + (\overline{A \oplus B})C$	$AB + (A \oplus B)C$
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	1	0	1
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	0	0	1
1	1	1	0	1	1

Table 5.2: CRTB 2 Truth Table

A	B	C	$\overline{A}B + \overline{B}C$	$AB + \overline{B}C$	$A\overline{B} + BC$
0	0	0	0	0	0
0	0	1	1	1	0
0	1	0	1	0	0
0	1	1	1	0	1
1	0	0	0	0	1
1	0	1	1	1	1
1	1	0	0	1	0
1	1	1	0	1	1

In both logic blocks the primary inputs are defined as A, B, C and the primary outputs are defined as P, Q, R. The primary outputs of CRTB 1 are expressed by $P = A \oplus B, Q = A\overline{B} + (\overline{A \oplus B})C, R = AB + (A \oplus B)C$ and the primary outputs of CRTB 2 are expressed by $P = \overline{A}B + \overline{B}C, Q = AB + \overline{B}C, R = A\overline{B} + BC$. These inputs and outputs are referred to as primary because they must be connected in a specific way to create an adder. The use of Fredkin gates introduces a number of ancilla inputs and garbage outputs. Ancilla inputs are constant inputs. There are certain Fredkin gate inputs that must be set to a constant logic 0 or logic 1 to produce the desired behavior. Garbage outputs are outputs that serve no purpose other than to maintain the one to one mapping in the circuit. These outputs are denoted by a

T and serve no purpose beyond testing for universal stuck at faults.

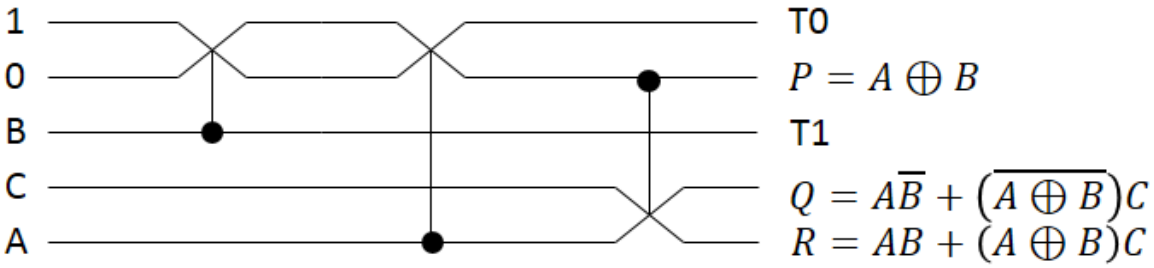


Figure 5.1: CRTB 1

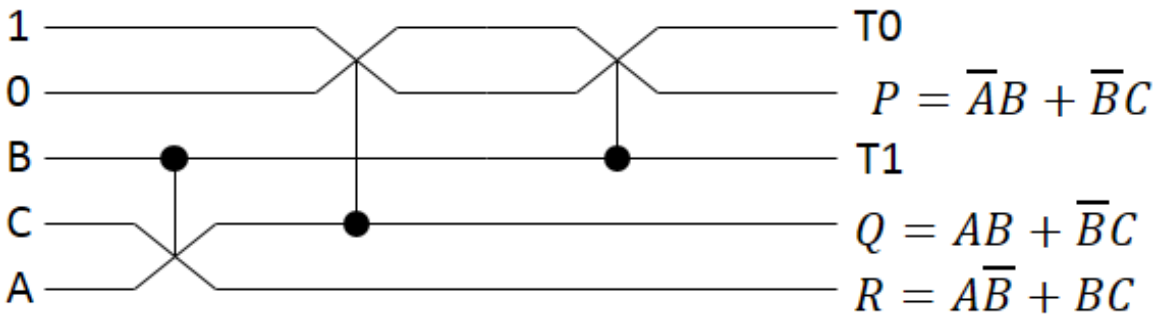


Figure 5.2: CRTB 2

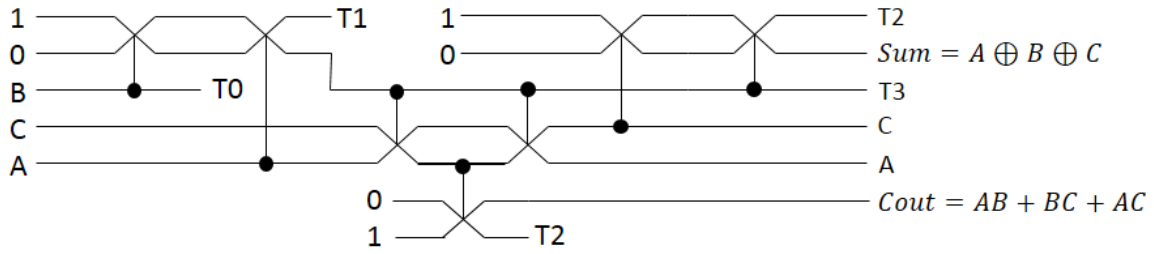


Figure 5.3: CR Full Adder Method 1

Cascading the primary outputs of CRTB 1 into the primary inputs of CRTB 2 produces the full adder shown in Figure 5.3. The inputs are defined as A, B, C. The outputs are defined as Sum and C_{out} where Sum is the sum of the inputs A, B, C and C_{out} is he carry output of the inputs A, B, C. The output R of CRTB 1 is the carry output C_{out}. That value is needed as an input to CRTB 2 in addition to being an output of the overall adder. An additional Fredkin gate is used to make a copy of the carry output C_{out} without introducing fanout.

An n-bit reversible full adder can be constructed by cascading n copies of Unit 1 with n copies of Unit 2. The resulting adder maintains the stuck at fault testability and potential application in NML computing of the full adder because it too is constructed solely from Fredkin gates. Define the inputs of the n-bit adder as $A_{n-1}A_{n-2}...A_1A_0$ and $B_{n-1}B_{n-2}...B_1B_0$ with a carry input C_{in}. The methodology for constructing the n-bit adder is explained below.

1. Cascading of CRTB 1

   (a) For i=0:

   Apply a CRTB 1 such that adder inputs A_0, B_0, C_{in} are passed to CRTB 1 inputs A, B, C, respectively.

   (b) For i=1 to i=n-1:

   Apply a CRTB 1 such that the values passed to CRTB 1 inputs A, B, C are A_i, B_i, R_{i-1}, respectively where R_{i-1} denotes the R output from the previous CRTB 1.

   (c) Further for i=n-1:

   Apply the R output of the previous CRTB 1 to the A input of a Fredkin gate.

2. Cascading of CRTB 2

   (a) For i=n-1:

   Apply a CRTB 2 so that the value passed to the CRTB 2 inputs A, B, C are $P(Fredkin), Q_{n-1}, P_{n-1}(CRTB1)$, respectively where $P(Fredkin)$ is the P output of the Fredkin gate that was added to the $(n-1)$th CRTB 1 and $Q_{n-1}, P_{n-1}(CRTB1)$ are outputs of the $(n-1)$th CRTB 1.

   (b) For i=n-2 to i=0:

   Apply a CRTB 2 so that the value passed to the CRTB 2 inputs A, B, C are R_i, Q_i, P_i, respectively where R_i, Q_i, P_i are the outputs of the ith CRTB 1.

An example of a CR 4-bit adder is shown in Figure 5.4. In the figure define the values being summed as A_3, A_2, A_1, A_0 and B_3, B_2, B_1, B_0 with the carry input defined as C_{in}. The sum is defined as S_3, S_2, S_1, S_0 and the carry output is defined as C_{out}.

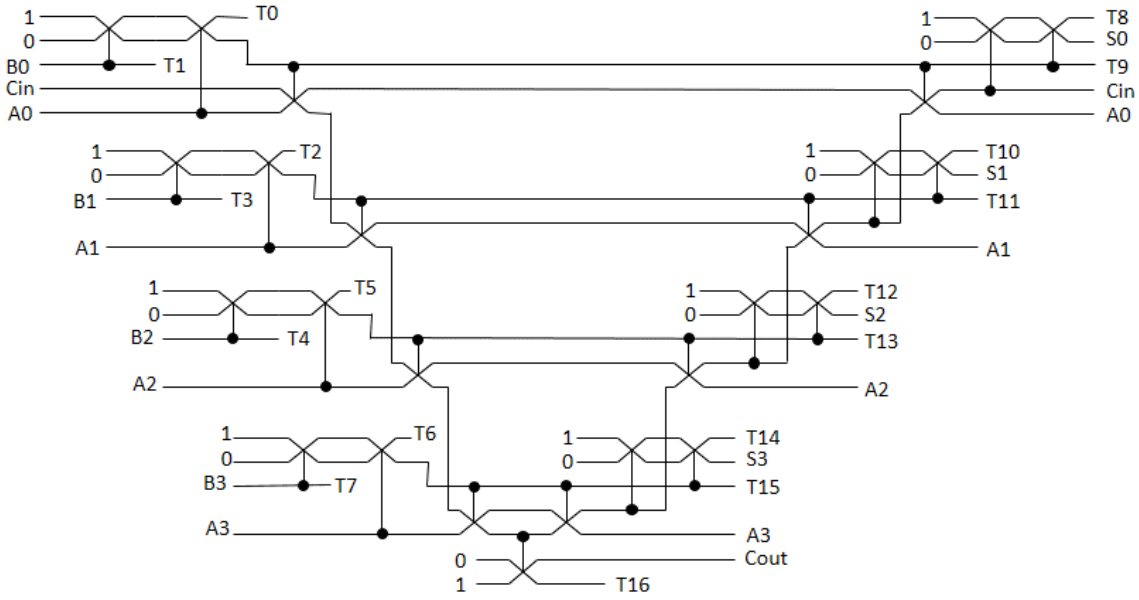


Figure 5.4: CR 4-bit Ripple Carry Adder Method 1

5.2 Design Methodology 2 of Proposed n-bit Testable Reversible Ripple Carry Adder

The second proposed design methodology uses an approach similar to the one used in [36] to implement a testable reversible n-bit ripple carry adder. This design method reduces both the propagation delay and cost in terms of Fredkin gates. Figure 5.5 contains the proposed full adder. The proposed design is made solely from Fredkin gates which means it is conservative reversible and thus only requires test vectors of all 0s and all 1s to detect all unidirectional stuck at faults.

As is typical with ripple carry adders, an n-bit ripple carry adder can be constructed using n copies of the individual full adder. Creating the n-bit ripple carry adder requires cascading the full adders so that the carry output C_{out} of one full adder

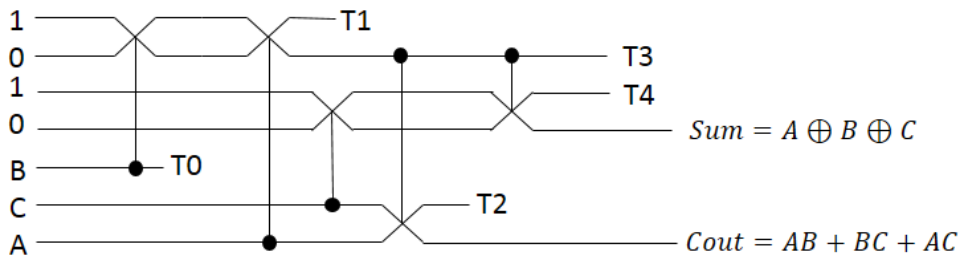


Figure 5.5: CR Full Adder Method 2

is passed to the carry input C_{in} of the next full adder. The created n-bit ripple-carry adder is conservative reversible and therefore only requires test vectors of all 0s and all 1s to detect all unidirectional stuck at faults. An example of a 4-bit CR ripple carry adder is shown in Figure 5.6.

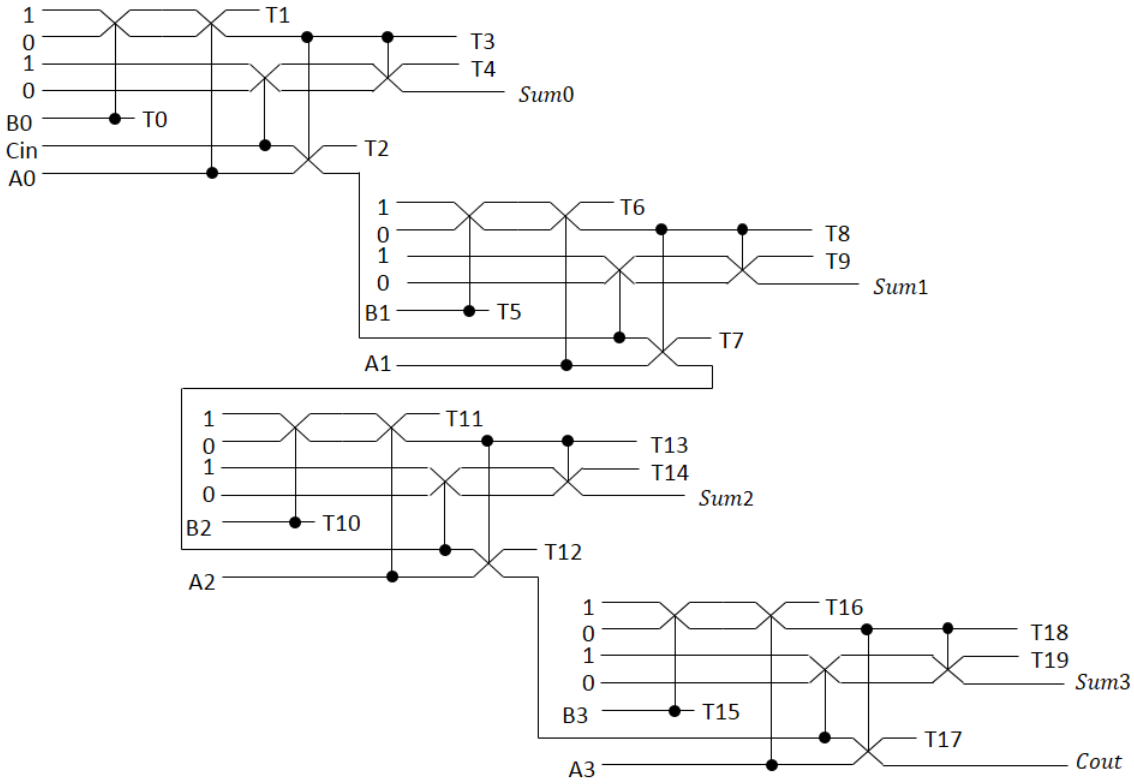


Figure 5.6: CR 4-bit Ripple Carry Adder Method 2

5.3 Comparison of Proposed n-bit Ripple Carry Adder Design Methodologies

A comparison of the cost and delay of the proposed design methodologies shows that our proposed design using method 2 is superior in both categories. The cost is the number of Fredkin gates required to implement the design and the delay is the number of Fredkin gates the input vector must propagate through to obtain the complete output vector. The proposed method 2 design requires fewer Fredkin gates than the method 1 design. For an n-bit adder the proposed method 2 design will require $n + 1$ fewer gates than the method 1 design. In addition, the proposed method 2 design has less delay than the method 1 design. For an n-bit adder the delay of the proposed method 2 design will be $n+2$ lower than in the method 1 design. In the method 1 design a signal has to propagate from the lowest significant bit to the most significant and then propagate back to the least significant bit. Another reason for the reduced delay in the method 2 design is in the design each full adder allows it to partially calculate its outputs before the carry output from the previous adder has been calculated. The second and third Fredkin gates in each full adder are independent of each other and can thus be considered to have a single gate delay. The carry output of each full adder is calculated before the final Fredkin gate and therefore only contributes to the overall delay in the final full adder. The delay for a single bit full adder is only 4 and each additional bit will only increase the overall propagation delay by 2. Table 5.3 shows a comparison between the cost and delay of the two design methodologies. All cost and delay values are in terms of Fredkin gates.

Table 5.3: Fredkin Cost and Delay Comparison

Adder Size	Cost		Delay	
	Method 1	Method 2	Method 1	Method 2
1-bit	7	5	7	4
2-bit	13	10	10	6
3-bit	19	15	13	8
4-bit	25	20	16	10
...	...	...	...	...
n-bit	6n+1	5n	3n+4	2n+2

Note: All values are in terms of the number of Fredkin gates

5.4 Conclusion

In this chapter we present two possible design methodologies to implement an n-bit conservative reversible adder in NML computing. Both design methodologies involve the use of Fredkin gates to make the designs applicable to NML computing. The first design method involves cascading two separate logic blocks to create ripple carry adders. The second design method uses the approach of constructing full adders and then cascading them to form ripple carry adders. The conservative reversible nature of the adders gives them the testability advantage of only requiring test vectors of all 0s and of all 1s to test for all unidirectional stuck at faults.

Chapter 6

Conclusions

In this thesis, multiple designs of arithmetic circuits have been proposed for use in majority logic based computing. Specific technologies targeted in this thesis were quantum dot cellular automata (QCA) and nanomagnetic logic (NML). Proposed designs of the QCA full adder, full subtractor, ripple carry adder, and ripple borrow subtractor were shown to improve on the number of cells, area, latency, and overall cost of existing designs by using five input majority gates in the designs. A multilevel five input majority gate was proposed in NML that was able to be constructed from identical cells. The merits of this gate were proven by the fact that proposed full adder design was able to reduce the number of required NML cells without needing additional copies of the inputs. In addition, a full subtractor NML design was proposed which makes use of the proposed 5-input majority gate. Lastly, the potential of majority logic for implementing conservative reversible (CR) logic designs was explored. Two implementations of n-bit adders were proposed. Both of the adders were created by exclusively using Fredkin gates which are CR in nature. Using only CR gates to create the adders means that the CR property is passed onto the proposed designs. The CR property of the adders allows them to be easily tested for all unidirectional stuck at faults. All of the presented QCA designs were verified through

the use of QCADesigner. All of the presented NML designs were verified through simulations performed on specially created Verilog descriptions.

The designs proposed in this thesis provide a solid foundation for future work. One such direction would be designing larger adder and subtractor circuits by following the same principles outlined in this thesis. More complex functional units such as multipliers, fast Fourier transform (FFT) units, arithmetic logic units (ALUs), and even an entire majority logic based processor can be designed by taking advantage of the designs proposed in this thesis. These larger units could also be designed using conservative reversible logic to make them easily testable for unidirectional stuck at faults. A potential hurdle to designing these larger and more complex units is the previously mentioned issue of a suitable CAD tool for NML circuits. That issue can be somewhat offset by increasing the functionality of program we developed to include such features as power dissipation or actual circuit dimensions. Another feature could be increasing user friendliness by adding a GUI that would allow a designer to make immediate cell location changes instead of having to cross reference the input file and the openSCAD file.

References

[1] John M Shalf and Robert Leland. Computing beyond moore's law. *Computer*, 48(12):14–23, 2015. © 2015 IEE Reprinted, with permission.

[2] Gordon E Moore et al. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82–85, 1998.

[3] Craig S Lent and P Douglas Tougaw. A device architecture for computing with quantum dots. *Proceedings of the IEEE*, 85(4):541–557, 1997.

[4] Carson Labrado and Himanshu Thapliyal. Design of adder and subtractor circuits in majority logic-based field-coupled qca nanocomputing. *Electronics Letters*, 52(6):464–466, 2016. Reproduced by permission of the Institution of Engineering & Technology.

[5] Carson Labrado and Himanshu Thapliyal. Design of a multilayer five-input majority gate and adder/subtractor circuits in nml computing. *Electronics Letters*, 52(19):1618–1620, 2016. Reproduced by permission of the Institution of Engineering & Technology.

[6] Carson Labrado, Himanshu Thapliyal, and Ronald F Demara. Design of testable adder circuits for spintronics based nanomagnetic computing. In *Nanoelectronic and Information Systems (iNIS), 2015 IEEE International Symposium on*, pages 107–111. IEEE, 2015. (c) 2015 IEE Reprinted, with permission.

[7] Marco Vacca, Mariagrazia Graziano, Juanchi Wang, Fabrizio Cairo, Giovanni Causapruno, Gianvito Urgese, Andrea Biroli, and Maurizio Zamboni. Nanomagnet logic: An architectural level overview. *Lecture Notes in Computer Science*, pages 223–256, 2014.

[8] Edit Varga, Alexei Orlov, Michael T. Niemier, Xiaobo Sharon Hu, Gary H. Bernstein, and Wolfgang Porod. Experimental demonstration of fanout for nanomagnet logic. *IEEE Transactions on Nanotechnology*, 9(6):668–670, 2010.

[9] Wolfgang Porod, Gary H Bernstein, György Csaba, Sharon X Hu, Joseph Nahas, Michael T Niemier, and Alexei Orlov. Nanomagnet logic (nml). In *Field-Coupled Nanocomputing*, pages 21–32. Springer, 2014.

[10] Marco Vacca. *Emerging Technologies-NanoMagnets Logic (NML)*. PhD thesis, Politecnico di Torino, 2013.

[11] P. Kartschoke. Implementation issues in conservative logic networks. In *M.S.E.E. Thesis, University of Virginia, Charlottesville VA*, 1992.

[12] G. Swaminathan. Concurrent error detection techniques using parity. In *M.S.E.E. Thesis, University of Virginia, Charlottesville VA*, 1989.

[13] G. Swaminathan, J. Aylor, and B. Johnson. Concurrent testing of vlsi circuits using conservative logic. In *Proc. International Conference on Computer Design (ICCD)*, pages 60–65, Cambridge, MA, September 1990.

[14] E. Fredkin and T Toffoli. Conservative logic. *International J. Theor. Physics*, 21:219–253, 1982.

[15] Sara Hashemi, Mohammad Tehrani, and Keivan Navi. An efficient quantum-dot cellular automata full-adder. *Scientific Research and Essays*, 7(2):177–189, 2012.

[16] Seong-Wan Kim. Design of parallel multipliers and dividers in quantum-dot cellular automata. 2011.

[17] Weiqiang Liu, Liang Lu, Máire O'Neill, and Earl E Swartzlander Jr. Cost-efficient decimal adder design in quantum-dot cellular automata. In *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*, pages 1347–1350. IEEE, 2012.

[18] Aaron Gin, P Douglas Tougaw, and Sara Williams. An alternative geometry for quantum-dot cellular automata. *Journal of Applied Physics*, 85(12):8281–8286, 1999.

[19] Heumpil Cho and Earl E Swartzlander Jr. Adder and multiplier design in quantum-dot cellular automata. *Computers, IEEE Transactions on*, 58(6):721–727, 2009.

[20] Keivan Navi, Razieh Farazkish, Samira Sayedsalehi, and Mostafa Rahimi Azghadi. A new quantum-dot cellular automata full-adder. *Microelectronics Journal*, 41(12):820–826, 2010.

[21] Moein Kianpour, Reza Sabbaghi-Nadooshan, and Keivan Navi. A novel design of 8-bit adder/subtractor by quantum-dot cellular automata. *Journal of Computer and System Sciences*, 80(7):1404–1414, 2014.

[22] Arman Roohi, Hossein Khademolhosseini, Samira Sayedsalehi, and Keivan Navi. A symmetric quantum-dot cellular automata design for 5-input majority gate. *Journal of Computational Electronics*, 13(3):701–708, 2014.

[23] Vikramkumar Pudi and K Sridharan. Low complexity design of ripple carry and brent–kung adders in qca. *Nanotechnology, IEEE Transactions on*, 11(1):105–119, 2012.

[24] Heumpil Cho and Earl E Swartzlander. Adder designs and analyses for quantum-dot cellular automata. *Nanotechnology, IEEE Transactions on*, 6(3):374–383, 2007.

[25] S Karthigai Iakshmi, G Athisha, Madurakavi Karthikeyan, and Chidambar Ganesh. Design of subtractor using nanotechnology based qca. In *Communication Control and Computing Technologies (ICCCCT), 2010 IEEE International Conference on*, pages 384–388. IEEE, 2010.

[26] Vikramkumar Pudi and K Sridharan. Efficient qca design of single-bit and multi-bit subtractors. In *Nanotechnology (IEEE-NANO), 2013 13th IEEE Conference on*, pages 1155–1158. IEEE, 2013.

[27] Stephan Breitkreutz, Irina Eichwald, Josef Kiermaier, Adam Papp, György Csaba, Michael Niemier, Wolfgang Porod, Doris Schmitt-Landsiedel, and Markus Becherer. 1-bit full adder in perpendicular nanomagnetic logic using a novel 5-input majority gate. In *EPJ web of conferences*, volume 75, page 05001. EDP Sciences, 2014.

[28] Robert Perricone, X Sharon Hu, Joseph Nahas, and Michael Niemier. Design of 3d nanomagnetic logic circuits: a full-adder case study. In *Proceedings of the conference on Design, Automation & Test in Europe*, page 119. European Design and Automation Association, 2014.

[29] M Cofano, G Santoro, M Vacca, D Pala, G Causapruno, F Cairo, F Riente, G Turvani, M Roch, M Zamboni, et al. Logic-in-memory: A nano magnet logic implementation. In *VLSI (ISVLSI), 2015 IEEE Computer Society Annual Symposium on*, pages 286–291. IEEE, 2015.

[30] Irina Eichwald, Stephan Breitkreutz, Grazvydas Ziemys, György Csaba, Wolfgang Porod, and Markus Becherer. Majority logic gate for 3d magnetic computing. *Nanotechnology*, 25(33):335202, 2014.

[31] Edit Varga, G Csaba, GH Bernstein, and W Porod. Implementation of a nanomagnetic full adder circuit. In *Nanotechnology (IEEE-NANO), 2011 11th IEEE Conference on*, pages 1244–1247. IEEE, 2011.

[32] Edit Varga, Michael T Niemier, Gyorgy Csaba, Gary H Bernstein, and Wolfgang Porod. Experimental realization of a nanomagnet full adder using slanted-edge magnets. *Magnetics, IEEE Transactions on*, 49(7):4452–4455, 2013.

[33] Michael Joseph Donahue and Donald Gene Porter. *OOMMF User's guide*. US Department of Commerce, Technology Administration, National Institute of Standards and Technology, 1999.

[34] Saurabh Kotiyal, Himanshu Thapliyal, and Nagarajan Ranganathan. Mach-zehnder interferometer based design of all optical reversible binary adder. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 721–726. EDA Consortium, 2012.

[35] Himanshu Thapliyal, Nagarajan Ranganathan, and Saurabh Kotiyal. Design of testable reversible sequential circuits. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 21(7):1201–1209, 2013.

[36] J. W. Bruce, M. A. Thornton, L. Shivakumaraiah, P. S. Kokate, and X. Li. Efficient adder circuits based on a conservative reversible logic gate. In *Proc. IEEE Symposium on VLSI, 2002*, pages 83–88, 2002.

Vita

Carson Labrado

Education

University of Kentucky
Bachelor of Science in Electrical Engineering, May 2014
Bachelor of Science in Computer Engineering, May 2014
Minors in Computer Science and Mathematics

Experience

Graduate Research Assistant
May 2015-Present
University of Kentucky
Lexington, KY

Teaching Assistant
Fall 2015-Present
University of Kentucky
Lexington, KY

Publications

Carson Labrado and Himanshu Thapliyal. "Design of a multilayer five-input majority gate and adder/subtractor circuits in NML computing." Electronics Letters 52.19 (2016): 1618-1620.
Carson Labrado and Himanshu Thapliyal. "Design of adder and subtractor circuits in majority logic-based field-coupled QCA nanocomputing." Electronics Letters 52.6 (2016): 464-466.
Carson Labrado, Himanshu Thapliyal, and Ronald F. Demara. "Design of Testable Adder Circuits for Spintronics Based Nanomagnetic Computing." 2015 IEEE International Symposium on Nanoelectronic and Information Systems. IEEE, 2015.
Himanshu Thapliyal, Carson Labrado, and Ke Chen. "Design procedures and NML cost analysis of reversible barrel shifters optimizing garbage and ancilla lines." The Journal of Supercomputing 72(3) (2016): 1092-1124.