Novel Architectures and Synthesis Methods for Quantum-dot Cellular Automata

Rajeswari D

Department of Computer Science and Engineering

Indian Institute of Technology Delhi

December- 2015

Novel Architectures and Synthesis Methods for Quantum-dot Cellular Automata

by

Rajeswari D

Department of Computer Science and Engineering

Submitted

in fulfillment of the requirements of the degree of Doctor of Philosophy

to the

Indian Institute of Technology Delhi

December- 2015

To

My precious mentor and cherished friend

Kolin Paul

Certificate

This is to certify that the thesis titled ***"Novel Architectures and Synthesis Methods for Quantum-dot Cellular Automata"*** being submitted by ***Rajeswari D*** for the award of the degree of ***Doctor of Philosophy*** in Computer Science and Engineeringis a record of bona-fide work carried out by her under our guidance and supervision at the Department of Computer Science and Engineering, Indian Institute of Technology Delhi. In our opinion, the thesis has reached the standards fulfilling the requirements of the regulations relating to the degree.

The work presented in this thesis has not been submitted elsewhere, either in part or full, for the award of any other degree or diploma.

Kolin Paul
Associate Professor
Dept. of Computer Science and Engineering
Indian Institute of Technology Delhi
New Delhi - 110016

M Balakrishnan
Professor
Dept. of Computer Science and Engineering
Indian Institute of Technology Delhi
New Delhi - 110016

Date :
New Delhi

Acknowledgements

It would be a gross under-statement to say that my PhD of seven long years has been a roller-coaster ride. These years have been colorful, fun, insightful, enriching, frustrating, painful, enlightening, and humbling among other things—in one word, memorable. In a more "zen" mood, I might term it a journey of self-discovery—I discovered parts of myself that I never knew existed. It has been my good fortune that the frustrating and painful portions were in no way caused by the people around me, or by circumstances that could have in any way been foreseen or preempted. It has been my better fortune that all the happy portions have had active contributions from the people around me: my beloved supervisors Dr. Kolin Paul and Prof. M. Balakrishnan, my gang of crackpot friends dotted across the map, and my ever-worried yet indulgent parents.

I tend to take for granted the support that my supervisors have provided me, and continue to provide. They set their own high standards at the very beginning of my PhD, and have never wavered from it. I find it redundant to thank them for being my mentors and grooming me from the enthusiastic but clueless 22 year old that I was, to the (more, much more!) mature researcher that I am today. It suffices to say that though this thesis might be the result of my perseverance, their influence is reflected in each page: the seeds of the thought processes, the constant search for improvement, the work ethics, and all the seemingly tiny but crucial "research" habits. In fact, all the "good" attributes in me as a researcher can easily be traced back to them. I might go so far as to attribute my obsessive attitude towards my work to my professors, especially to Bala Sir and Anshul Sir, but that would be stretching it too far. After all, unlike me, they manage to be efficient

while being meticulous. I hope I do get there eventually. One observes, remembers and learns.

I do not know of many (any?) other PhD students who go gaga over their supervisors like I do. I always have, and will continue to bore anybody who cares to listen about how precious "my" supervisors are. May be, I am a unique crackpot like that. But more likely, *they* are unique like that. They have allowed me space and a lot (emphasis on LOT) of time to work on my own terms and pace, gently nudging me along now and then. In particular, I have taken complete advantage of Kolin Sir (abuse?) on this regard, and he has been insanely patient with me. I remember him with twinkling eyes, animatedly explaining QCA to me, in our first official meeting. He continues to get excited about ideas, to get me on board, and to ensure that I do not let other frustrations (minor or major) dampen my spirits. Though I have not been able to have those lively and exciting research discussions with him in the past three years, I would always cherish the ones I did have. And I hope to have more of those discussions, whenever and however possible.

Speaking of the past three years, no acknowledgments page would be complete without a quick recollection of those days. On one hand, I went from being an obsessive researcher to a lost-and-never-found PhD student. On the other, I was humbled by the trust and patience that my supervisors displayed in waiting for me to find myself again. Kolin Sir took it upon himself to "rehabilitate" me into research one slow step at a time, with too many set-backs to count. Each deadline missed was an opportunity for him to remind me that there are only a "few" steps more to go. Each deadline met was an achievement, for him more than for me. While I found and pulled out resilience from deep within myself, he fought to keep me afloat—both as a researcher, and as an individual. I came to IITD as a student looking for a mentor and some good research problems to work on. I am glad to leave having found an unlikely friend, reinforced confidence in myself as a researcher, and pride to have produced this thesis, under such guidance. I refuse to say anymore, for I believe, it is implicit and understood.

Thank you, all. For being you. I need say no more.

Rajeswari D.

P.S. : All hail the Internet—the one non-human element that has sustained and (literally) nourished me at the click of a button, and saved my sanity in the darkest of times through an endless supply of books, and medical and psychological help that showed me light at the end of the tunnel.

Abstract

Quantum-dot Cellular Automata (QCA) is an emerging nano-scale low-power fabric that is being explored as an alternative to traditional CMOS devices. The two most interesting features of the QCA paradigm are: a) both wires and gates are created using the same basic element (the QCA cell), and the required functionality is achieved by directing data flow using the QCA clocking scheme. b) the 3-input Majority gate, the primary logic element of QCA, is computationally more powerful than the AND/OR gates of traditional fabrics. In this work, we exploit these features to enhance the usability and efficiency of QCA devices through novel architectures and synthesis methods :

Clocked Coplanar Wire Crossings Traditional wire crossings in QCA use the weak coupling between distant cells and are unreliable. We propose a robust crossing by extending the QCA clock to an 8-phase scheme. This exploits the ability of the QCA clock to direct data flow to eliminate all need for complicated cell positioning and orientations.

p-QCA: A Programmable Architecture We develop a programmable device architecture for QCA that exploits the fabric, rather than import concepts from traditional FPGAs. This architecture eliminates the difference between routing and logic elements on a programmable device, and achieves programmability while retaining near-ASIC performances.

***n*-input Majority Algebra and Minimization** Logic synthesis methods for QCA have been focused on searching for patterns that fit well into 3-input Majority gate networks, and do

not consider the fundamental mathematics of Majority. We generalize the 3-input Majority gate into the generic n-input Majority gate, and develop the fundamental algebra for n-input Majority. This new Boolean algebra for the Majority operator is a generalization of the traditional Boolean algebra for sum and product operators. Using this, we propose a specialized logic minimization method for Majority gates.

Technology Mapping for QCA We bridge the gap between technology-independent n-input Majority minimization and the needs of QCA, which uses 3-input Majority (and at most 5-input Majority) gates. We propose an effective method to map unrestricted n-input Majority expressions to those that use only 3-input Majority (or at most k-input Majority) terms, as required by the target technology.

Exploiting Symmetry Majority functions form a special class of symmetric functions that can be used to efficiently realize all types of symmetry in a Boolean graph. We develop a direct method of expressing any symmetric function in terms of n-input Majority-terms. Boolean graph decomposition techniques targeted for large scale Majority synthesis need to be aware of the various types and levels of symmetry seen in Boolean functions, and to be able to exploit such symmetry using Majority-terms. We extend the concept of Binary Decision Diagrams to Symmetry Decision Diagrams (SDD), and take the representation of generic Boolean functions closer to Majorities and symmetric forms. We develop the basic framework for building and manipulating SDDs to perform Majority-friendly Boolean graph decomposition.

The *MajSynth* tool We have built *MajSynth*, a logic synthesis tool that implements our two-step strategy of technology-independent logic minimization to n-input Majority expressions followed by technology mapping to 3-input Majority (or 5-input Majority) expressions. We have included symmetry optimizations and SDD-based symmetry-aware graph decomposition techniques in this tool, making it a comprehensive framework for logic synthesis for Majority gates. *MajSynth* is the first comprehensive tool that addresses generic n-input Majority, and handles the core issues of large scale Majority synthesis using the fundamental mathematics of Majority.

Table of Contents

Certificate i

Acknowledgements iii

Abstract v

Table of Contents vii

List of Figures xiii

List of Tables xvii

List of Algorithms xix

Introduction

1 Quantum-dot Cellular Automata 3

– *An emerging technology and angles of interest*

1.1 QCA Cells, Wires, and Gates 4

1.2 The QCA Clock 7

1.3 The Power of Majority Gates 10

1.4 Exploiting the QCA fabric 12

1.4.1 Programmability with QCA 13

1.4.2 Computing with Majority Gates 14

I Programmable Architecture 17

2 Programmability and QCA 19

– *A survey of literature and possibilities*

2.1 Programmable QCA till now 20

2.1.1 LUTs, CLBs, and Switches 20

2.1.2 Arrays of AND, OR gates 22

2.1.3 Sea of Universal Gates 23

2.1.4 2-input NAND gates among interconnects 26

2.2 Designing Programmable QCA 28

2.2.1 Fabrication Needs 28

2.2.2 Logic Elements 29

2.2.3 Routing Elements 30

2.2.4 Programming Method 30

2.2.5 Granularity and Regularity 31

2.3 Deconstructing Programmable QCA 32

3 Clocked QCA Wire Crossings 35

– *A new method for crossing two QCA wires*

3.1 QCA Wire Crossings till now 36

3.1.1 Rotated-cell Coplanar Wire Crossings 36

3.1.2 Multi-layer Wire Crossings 38

3.1.3 Logical Wire Crossings 39

3.1.4 Signal Distribution by Customized Clocking 39

3.1.5 Eliminating Wire Crossings 41

3.2 Clocking-based QCA Wire Crossing 42

3.2.1 Our 8-phase Clocking Scheme 42

3.2.2 Example : 2-input XOR gate 44

3.2.3 MATLAB Simulation 46

3.3 A Qualitative Analysis 47
3.3.1 Delay 48
3.3.2 Fabrication 48
3.3.3 Reliability and Robustness 50
3.3.4 Area 50

4 The p-QCA Architecture **53**
– *A new architecture for a programmable QCA device*
4.1 Computing with Tiles 54
4.1.1 Tiles for Logic and Routing 54
4.1.2 Tiles for Input 58
4.1.3 A 2-input XOR on p-QCA tiles 60
4.1.4 A Grid of Tiles 63
4.2 Programming with the Clock 64
4.2.1 A Clocking structure for p-QCA 65
4.3 p-QCA Design Flow 67
4.4 An Analysis 68
4.4.1 Area 70
4.4.2 Timing 73
4.4.3 Routing = Logic and Resource Utilization 73

II Logic Synthesis **77**

5 Exploiting the QCA Majority Gate **79**
– *A survey of literature and possibilities*
5.1 The Mathematics of Majority 83
5.2 Majority Logic Synthesis 85
5.3 Threshold Logic Synthesis 87
5.4 Summary 89

6 Boolean Algebra and Majority **91**
– *Extending the laws of Boolean algebra to n-input Majority*

6.1 Majority Gates 92
6.2 The Majority Operator 95
6.2.1 Majority with Constants 96
6.3 Traditional Boolean Algebra and Majority 100
6.4 A New Boolean Algebra for Majority 102
6.4.1 Commutativity to Symmetry 103
6.4.2 Associativity to Flatten 104
6.4.3 Distributivity 106
6.4.4 Annihilation to Absolute Majority 108
6.4.5 Identity to Cancellation 109
6.4.6 Idempotence to Median 110
6.4.7 Absorption 115
6.4.8 De Morgan's Law to Negation 115
6.4.9 Duality 117
6.4.10 Complements to Certainty 122
6.4.11 Shannon's Expansion to Symmetry Expansion 126

7 Advanced Algebra of Majority **135**
– *Deeper into the mathematics of n-input Majority*
7.1 Majority and Unateness 136
7.2 Majority on Two Ordered sets 139
7.3 Majority Expansion 143
7.4 Majority and Prime Implicants 152
7.4.1 Implicants of Majority 154
7.4.2 Majority as Sum of Prime Implicants 159
7.5 Equivalence of Majority Terms 165
7.5.1 Duality of Implicants 167
7.5.2 Detecting Equivalence of Majority-terms 170
7.6 Majority Compaction 173

8 Minimizing to *n*-input Majority **181**

– *Implementing Boolean logic with n-input Majority*

8.1 Minterms to Majority 182

8.1.1 Difference Rule 183

8.1.2 Collapse Rule 184

8.2 Iterative Majority Minimization 185

8.2.1 Distributivity in Minimization 187

8.3 Implementation in MajSynth 190

8.3.1 The Majority-Invert Graph 190

8.3.2 Transformations on Majority-Invert Graph 194

8.3.3 Minimization of Majority-Invert Graph 197

8.4 Efficacy, Consistency and Scalability 201

9 Mapping *n*-input Majority to QCA **205**

– *Downsizing n-input Majority to suit target technology*

9.1 Downsizing using Majority-counts 206

9.2 Downsizing Differences 212

9.3 Implementation in MajSynth 214

9.4 Efficacy, Consistency and Scalability 216

10 Symmetry-aware Majority Synthesis **219**

– *Exploiting Majority gates to implement symmetry in logic*

10.1 Symmetry and Majority 220

10.1.1 Symmetric Functions 221

10.1.2 Partially Symmetric Functions 223

10.2 SDD: Symmetry Decision Diagram 226

10.2.1 Majority Expression of SDD 230

10.3 Implementation in MajSynth 235

10.3.1 Symmetry and Iterative Minimization 236

10.3.2 SDD-based Minimization 237

11 The MajSynth Tool **241**

– *A framework for Majority logic synthesis*

11.1 The MajSynth Framework 241

11.2 Efficacy, Consistency, and Scalability 244

Conclusion

12 The Now and the Next **255**

– *A summary of this work and its significance*

12.1 Programmable Architecture for QCA 255

12.2 Logic Synthesis for Majority gates 256

Bibliography **259**

Publications from the Thesis **267**

Author Biography **269**

List of Figures

Figure 1.1 QCA Cells and Wires 4

Figure 1.2 The Primary Logic Gates in QCA 6

Figure 1.3 4-phase Clocking Scheme for QCA 8

Figure 1.4 The QCA clock directs data flow in a circuit 9

Figure 1.5 3-input XOR using 2-input AND/OR gates Vs 3-input Majority gates 10

Figure 1.6 5-input Majority and 7-input Majority gates in QCA 11

Figure 1.7 3-input and 5-input XOR functions using *n*-input Majority gates 11

Figure 2.1 PLA for QCA proposed by Crocker et al. 22

Figure 2.2 PAL for QCA proposed by Tougaw et al. 23

Figure 2.3 AOI gate for QCA proposed by Huang et al. 24

Figure 2.4 Universal gate for QCA proposed by Kim et al. 25

Figure 2.5 NAND-based FPGA design for QCA proposed by Niemier et al. 26

Figure 2.6 *4-diamond* architecture proposed by Niemier and Kogge 27

Figure 3.1 Rotated-cell Coplanar Wire Crossing in QCA 36

Figure 3.2 Robust rotated-cell wire crossings using redundancy 37

Figure 3.3 Multi-layer Wire Crossing in QCA 38

Figure 3.4 Logical Wire Crossing in QCA 39

Figure 3.5 Crossbar network for signal distribution proposed by Graunke et al. 40

Figure 3.6 Clocking scheme used by Tougaw et al. in signal distribution network 40
Figure 3.7 The new Clocking-based QCA Wire Crossings 43
Figure 3.8 2-input XOR using Rotated-cell Vs Clocked Crossings 45
Figure 3.9 *MATLAB* simulation results for 2-input XOR shown in Figure 3.8b 47
Figure 3.10 Robustness of clocked crossing and MATLAB simulator 49
Figure 3.11 Specialized use of clocked crossings to implement compact 2-input-XOR 51

Figure 4.1 "Clocking= Functionality": Majority gate, fan-out, and wire crossing 55
Figure 4.2 Basic tile set for P-QCA 55
Figure 4.3 Illustration of clocking-based data flow across P-QCA tiles 57
Figure 4.4 Fixed-polarity input blocks for P-QCA 59
Figure 4.5 2-input XOR mapped on to a P-QCA grid 61
Figure 4.6 Simulation results for Figure 4.5 using *MATLAB* engine 62
Figure 4.7 A possible clock distribution scheme for P-QCA 65
Figure 4.8 Design flow for P-QCA 67
Figure 4.9 P-QCA Vs. *4-diamond*: Percentage of total area used for wire crossings 71
Figure 4.10 Simple12 1-bit ALU mapped on to a P-QCA grid 72
Figure 4.11 P-QCA Vs. *4-diamond*: Percentage of Active Programmable Elements 74

Figure 6.1 The n-input Majority logic gate 92
Figure 6.2 A 5-input Majority gate with duplicate inputs 92
Figure 6.3 7-input and 9-input Majority gates with duplicate inputs 93
Figure 6.4 Ordering of Majority-counts and inverse Majority-counts 114
Figure 6.5 Majority expressions of a function, its negation, conjugate, and dual 121

Figure 7.1 Sum of all Prime implicants of $\langle X_2\, Y_2^2\, Z_2^3\, 0^2 \rangle$ and $\langle X_2\, Y_2^2\, Z_2^3\, 1^2 \rangle$ 164
Figure 7.2 Sum of all Prime implicants of $\langle X_2\, Y_2^3\, Z_2^4\, 0^3 \rangle$ 166
Figure 7.3 Equivalence of Majority-terms 172
Figure 7.4 Compaction of a Majority-term 177

Figure 8.1 Iterative minimization of 3-input XOR 186
Figure 8.2 Iterative minimization of a 4-variable function 189

Figure 8.3 Majority-Invert graph for sum of minterms of 3-input XOR 191

Figure 8.4 Majority-Invert graph transformed to minimize 3-input XOR 198

Figure 8.5 Majority-Invert graph transformed to minimize 3-input XOR 199

Figure 9.1 $\langle a\, b\, c\, d\, e\, f\, g \rangle$ rewritten in terms of $\{a, b, c, d\}$, $\{e, f, g\}$ 207

Figure 9.2 $\langle a\, b\, c\, d\, T^3 \rangle$ rewritten in terms of $\{d, T^3\}$, $\{a, b, c\}$ 208

Figure 10.1 Majority expression of the 7-input XOR function 221

Figure 10.2 Majority expression of a 10-input symmetric function 222

Figure 10.3 Variants of Symmetry Decision Diagrams 228

Figure 10.4 Reordering an Ordered Symmetry Decision Diagram 229

Figure 10.5 Partially symmetric functions with ordered Symmetry-cofactors 231

Figure 10.6 Partially symmetric functions with two Symmetry-cofactors 233

Figure 10.7 Partially symmetric functions with exclusive common factors/terms 234

Figure 11.1 *MajSynth*: A framework for Majority logic synthesis 242

Figure 11.2 *MALS* Vs. *MajSynth* with Iterative Minimization 246

Figure 11.3 *MALS* Vs. *MajSynth* with optimized Iterative Minimization 248

Figure 11.4 *MALS* Vs. *MajSynth* with Symmetry-aware graph decomposition 250

List of Tables

Table 2.1 Comparison of different Programmable architectures for QCA 33

Table 4.1 P-QCA Vs *4-diamond*: No. of Processing Elements, Total Area and Latency 70

Table 5.1 3-input Majority implementations of 3-variable functions 80

Table 5.2 *n*-input Majority implementations of 3-variable functions 82

Table 5.3 Comparison of logic synthesis techniques for Majority gates 88

Table 6.1 Functions implemented by 3, 5, 7 and 9-input Majority gates 94

Table 6.2 Basic laws of traditional Boolean Algebra with Majority operator 101

Table 6.3 Basic laws of Boolean Algebra for Majority 132

Table 6.4 Properties of Boolean Algebra for Majority 133

Table 8.1 *n*-input Majority reductions of 4 and 5-variable functions 201

Table 9.1 Downsizing 5-input and 7-input Majority-terms by division of inputs 211

Table 9.2 No. of gates for realizing 5, 7, and 9-input Majority-terms on QCA 217

Table 11.1 *MALS* Vs. *MajSynth*: *k*-decomposed *MCNC* benchmarks 249

Table 11.2 *MALS* Vs. *MajSynth*: Symmetry decomposed *MCNC* benchmarks 250

List of Algorithms

Algorithm 7.1 All Prime 1-implicants and Prime 0-implicants of a Majority-term 163

Algorithm 7.2 Equivalence of two Majority-terms on the same variables 171

Algorithm 7.3 Compact a Majority-term to its smallest multiset using constraints 175

Algorithm 8.1 Create a Majority-Invert graph representing a sum of minterms 193

Algorithm 8.2 Transform Majority-term by Compact, Collapse, Difference, Flatten 195

Algorithm 8.3 Distribute Majority-term over an inner-term if it simplifies expression 196

Algorithm 8.4 Iterative Minimization of a Majority-Invert Graph 200

Algorithm 9.1 Downsize a Majority-Invert graph to use at most k-input Majority gates 215

Algorithm 10.1 Symmetry-optimized Iterative Minimization 238

Algorithm 10.2 SDD-based Majority Minimization of Boolean Functions 239

Algorithm 11.1 *MajSynth*: A Majority synthesis tool based on n-input Majority algebra 243