Weighted Test Pattern Generation for Built-In Self-Test using Cellular Automata

by

© Blake W. Podaima

A thesis
presented to the University of Manitoba
in partial fulfillment of the
requirements for the degree of
Master of Science
in
Electrical Engineering

Winnipeg, Manitoba, 1989

© Blake W. Podaima, 1989

National Library of Canada
Canadian Theses Service
Ottawa, Canada
K1A 0N4

Bibliothèque nationale du Canada
Service des thèses canadiennes

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-51589-9

Canada

WEIGHTED TEST PATTERN GENERATION FOR BUILT-IN
SELF-TEST USING CELLULAR AUTOMATA

BY

BLAKE W. PODAIMA

A thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

© 1989

Permission has been granted to the LIBRARY OF THE UNIVERSITY OF MANITOBA to lend or sell copies of this thesis. to the NATIONAL LIBRARY OF CANADA to microfilm this thesis and to lend or sell copies of the film, and UNIVERSITY MICROFILMS to publish an abstract of this thesis.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

Abstract

In this thesis, the issue of generating probabilistically weighted test patterns, for use in a Built-In Self-Test environment, is addressed. The methodology presented considers the suitability of incorporating Cellular Automata-based structures, as opposed to those based conventionally on the Linear Feedback Shift Register. By adopting this strategy, there are, in general, beneficial improvements noted with respect to both system quality and performance. The means by which this is achieved is partly due to improvements in the statistical properties governing the time evolution mechanism. Some cellular-automata are capable of complex behaviour, which statistically qualify as attractive candidates for use in weighted test pattern generator design. The analysis involved in determining and statistically evaluating these potential models is discussed, and conveyed for the additional purpose of comparison with previous and statistically independent models.

It is shown that the cellular automata based weighted test pattern generator configurations do, indeed, possess better statistical properties (with respect to: density and average density; probability mass function; magnitude spectrum; space-phase cross correlation; and, bit sequence tuple lengths), in an appreciable manner. Due to this, it is possible to incorporate high quality parallel weighted test pattern generation for use in an array of testing schemes. One of the most interesting proposals suggests an extension of a Cellular Automaton Logic Block Observer to include the functionality of a Weighted Test Pattern Generator. In a more directly implementable approach, very efficient, stand alone, selectable Parallel Weighted Test Pattern Generators may be realized. Finally, because of the modularity and simplicity inherent in these cellular automata architectures, there is a reduction in wiring complexity and an anticipated concomitant improvement in test circuit performance.

Acknowledgments

I would like to extend my thanks and appreciation to Dr. Robert D. McLeod for his supervision, assistance, encouragement, and support, during the development of this work.

Moreover, I would also like to express innumerable thanks to those who have contributed in other capacities: Dr. Miroslaw Pawluk; Dr. Peter D. Hortensius; Mr. Robert Nesbitt; as well as many of my other colleagues in the VLSI laboratory.

Financial support from the Natural Sciences and Engineering Research Council of Canada, and the University of Manitoba Academic Development Fund, and equipment loans from the Canadian Microelectronics Corporation, are graciously acknowledged.

I am also greatful to CANCAD Technology Inc. for contributing their plotting software for this work.

To the memory of Father,
whose continuing support will always remain.

Table of Contents

Abstract ii
Acknowledgements iii
Dedication iv
Table of Contents v
List of Figures ix
List of Tables xxii

Chapter 1: Introduction 1

Chapter 2: Design and Test of VLSI 4

2.1. Structured Design Methodologies 4
2.2. Testability of VLSI 5
2.2.1. Test Generation and Verification 6
2.2.2. Design For Test Methodology 7
2.2.3. Fault Modelling 7

Chapter 3: Design For Testability 10

3.1. Introduction to DFT 10
3.2. Built-In Self-Test approach to DFT 11
3.2.1. Parallel Signature Analysis 13
3.2.2. Built In Logic Block Observation 13

Chapter 4: Circuit Testability using Random Test Pattern Generation 19

4.1. Purely Random versus Pseudorandom Testing 20
4.1.1. Expected Fault Coverage 22
4.1.2. Test Confidence 22

4.1.3. Expected Test Length 23
4.1.4. Average Test Length 24
4.1.5. Comparison of Purely Random and Pseudorandom Test Quality Measures 25

Chapter 5: Cellular Automata-An Overview 27

5.1. Introduction to Cellular Automata 27
5.2. Cellular Automata Classification 29
5.3. One-dimensional Cellular Automata 32
5.4. Class and Attractor Behaviour 35
5.5. Summary 37

Chapter 6: Weighted Test Pattern Generation for Built-In Self-Test using Cellular Automata 38

6.1. Weighted Test Pattern Generation for Fault Detectability 39
6.2. Weighted Test Pattern Generation Methodology 40
6.3. Cellular Automata based WTPG Considerations: 44
6.3.1. Using Conventional and Hybrid One-dimensional Cellular Automata 44
6.3.2 Using Driving Engines with External Weighting Logic 46
6.3.2.1. CA based Pseudorandom Driving Engines 48
6.3.2.2. WTPG Logical Configurations 50
6.4. Statistical Evaluation of WTPGs 53
6.4.1. Statistical Estimators 53
6.4.2. State-time Visualization 54
6.4.3. Density and Average Density 55
6.4.4. Probability Mass Function 56
6.4.4.1. Goodness-of-fit Test 58
6.4.5. Magnitude Spectrum 61
6.4.6. Auto and Cross-Correlation Coefficient Functions 64

6.4.6.1. Correlation of Original Driving Engines 66
6.4.6.2. Correlation of Weighted Configurations 70
6.4.7. Bit Sequence Tuple Lengths 71
6.4.7.1. Tuple Profiles-Parallel Weighted Test Pattern Generation 74
6.4.7.2. Tuple Profiles-Concatenated Weighted Test Pattern Generation 76

Chapter 7: Cellular Automata Based Testing Apparatus 7

7.1. Cellular Automaton Logic Block Observer 77
7.2. Practical Considerations 78
7.2. Signature Analysis using Cellular Automata 83
7.3. Weighted Cellular Automaton Logic Block Observer 83

Chapter 8: Conclusions and Future Work 85

8.1. Conclusions 85
8.2. Future Work 88

References 90

Appendix A: Boolean Logic Probability Profiles 95

Appendix B: Statistical Estimators 103

B.1. Estimation of Independent Random Variables 103
B.2. Estimation of Dependent Random Variables 105
B.3. Method of Independent Replications 108

Appendix C: State-Time Evolutions 113

Appendix D: Density and Average Density Evolutions 126

Appendix E: PMF Evaluation-Definitions and Theorems 151

Appendix F: Histogram Evolutions 158

Appendix G: Magnitude Spectrums 169

Appendix H: Space-Phase Correlation 177

Appendix I: Bit Sequence Tuple Profiles 191

Appendix J: Glossary 198

List of Figures

Figure 3.1: *Built In Logic Block Observer (BILBO) register with four mode functionality: W=8* 14

Figure 3.2: *Sequential Network Testing strategy using Built In Logic Block Observation* 15

Figure 3.3: *Concatenated Network Testing strategy using Built In Logic Block Observation* 16

Figure 3.4: *Bus-Oriented Testing strategy using Built In Logic Block Observation* 17

Figure 5.1a): *Von Neumann Neighbourhood (Two-dimensional)* 31

Figure 5.1b): *Moore Neighbourhood (Two-dimensional)* 31

Figure 5.1c): *Local Neighbourhood (One-dimensional)* 31

Figure 5.2a): *One-dimensional CA with Cyclic boundary conditions* 33

Figure 5.2b): *One-dimensional CA with Null boundary conditions* 33

Figure 5.3: *Four different classes of CA behaviour* 42

Figure 6.1: *Nested AND-gate Tree with Three-input Logic* 42

Figure 6.4a) *Parallel WTPG Sharing Configuration: [S]* 51

Figure 6.4b): *Parallel WTPG Zero Spacing Configuration: [J=0]* 51

Figure 6.4c): *Parallel WTPG One Spacing Configuration: [J=1]* 52

Figure 6.4d): *Parallel WTPG N/2 Configuration: [J=(N/2)-1]* 52

Figure 7.1: *Non-linear Rule 30 CA CALBO Cell* 80

Figure 7.2a): *Linear Rule 90 CA CALBO Cell* 81

Figure 7.2b): *Linear Rule 150 CA CALBO Cell* 81

Appendix A: Boolean Logic Probability Profiles

Figure A.1a): *Probability Profile for Inverter* 96

Figure A.1b): *Probability Profile for AND gate: equal input probabilities* 96

Figure A.1c): *Probability Profile for NAND gate: equal input probabilities* 97

Figure A.1d): *Probability Profile for OR gate: equal input probabilities* 97

Figure A.1e): *Probability Profile for NOR gate: equal input probabilities* 98

Figure A.1f): *Probability Profile for XOR gate: equal input probabilities* 98

Figure A.1g): *Probability Profile for XNOR gate: equal input probabilities* 99

Figure A.2a): *Probability Profile for AND gate: variable input probabilities, Pin A and Pin B.* *100*

Figure A.2b): *Probability Profile for NAND gate: variable input probabilities, Pin A and Pin B* 100

Figure A.2c): *Probability Profile for OR gate: variable input probabilities, Pin A and Pin B* 101

Figure A.2d): *Probability Profile for NOR gate: variable input probabilities, Pin A and Pin B* 101

Figure A.2e): *Probability Profile for XOR gate: variable input probabilities, Pin A and Pin B* 102

Figure A.2f): *Probability Profile for XNOR gate: variable input proities, Pin A and Pin B* 102

Appendix C: State-Time Evolutions

Figure C.1: *Space-Time Evolution of test patterns for NLFSR: W=53; L=400* 114

Figure C.2: *Space-Time Evolution of test patterns for LFSR: W=53; L=400* 114

Figure C.3: *Space-Time Evolution of test patterns for Rule 90/150 HCA: W=53; L=400* 115

Figure C.4: *Space-Time Evolution of test patterns for Rule 30 CA: W=53; L=400* 115

Figure C.5a): *Space-Time Evolution of test patterns for NLFSR weighted to 25% (AND-gate bank): [S]; W=52; L=400* 116

Figure C.5b): *Space-Time Evolution of test patterns for NLFSR weighted to 75% (OR-gate bank): [S]; W=52; L=400* 116

Figure C.6a): *Space-Time Evolution of test patterns for LFSR weighted to 25% (AND-gate bank): [S]; W=52; L=400* 117

Figure C.6b): *Space-Time Evolution of test patterns for LFSR weighted to 75% (OR-gate bank): [S]; W=52; L=400* 117

Figure C.7a): *Space-Time Evolution of test patterns for Rule 90/150 HCA weighted to 25% (AND-gate bank): [S]; W=52; L=400* 118

Figure C.7b): *Space-Time Evolution of test patterns for Rule 90/150 HCA weighted to 75% (OR-gate bank): [S]; W=52; L=400* 118

Figure C.8a): *Space-Time Evolution of test patterns for Rule 30 CA weighted to 25% (AND-gate bank): [S]; W=52; L=400* 119

Figure C.8b): *Space-Time Evolution of test patterns for Rule 30 CA weighted to 75% (OR-gate bank): [S]; W=52; L=400* 119

Figure C.9a): *Space-Time Evolution of test patterns for NLFSR weighted to 25% (AND-gate bank): [J=0]; W=26; L=400* 120

Figure C.9b): *Space-Time Evolution of test patterns for NLFSR weighted to 75% (OR-gate bank): [J=0]; W=26; L=400* 120

Figure C.10a): *Space-Time Evolution of test patterns for LFSR weighted to 25% (AND-gate bank): [J=0]; W=26; L=400* 121

Figure C.10b): *Space-Time Evolution of test patterns for LFSR weighted to*

75% (OR-gate bank): [J=0]; W=26; L=400 121

Figure C.11a): *Space-Time Evolution of test patterns for Rule 90/150 HCA weighted to 25% (AND-gate bank): [J=0]; W=26; L=400* 122

Figure C.11b): *Space-Time Evolution of test patterns for Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=0]; W=26; L=400* 122

Figure C.12a): *Space-Time Evolution of test patterns for Rule 30 CA weighted to 25% (AND-gate bank): [J=0]; W=26; L=400* 123

Figure C.12b): *Space-Time Evolution of test patterns for Rule 30 CA weighted to 75% (OR-gate bank): [J=0]; W=26; L=400* 123

Figure C.13a): *Space-Time Evolution of test patterns for LFSR weighted to 25% (AND-gate bank): [J=25]; W=26; L=400* 124

Figure C.13b): *Space-Time Evolution of test patterns for LFSR weighted to 75% (OR-gate bank): [J=25]; W=26; L=400* 124

Figure C.14a): *Space-Time Evolution of test patterns for Rule 90/150 HCA weighted to 25% (AND-gate bank): [J=25]; W=26; L=400* 125

Figure C.14b): *Space-Time Evolution of test patterns for Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=25]; W=26; L=400* 125

Appendix D: Density and Average Density Evolutions

Figure D.1a): *Density Evolution of test patterns for NLFSR: W=30; L=500* 127

Figure D.1b): *Average Density Evolution of test patterns for NLFSR: W=30; L=1,000* 127

Figure D.2a): *Density Evolution of test patterns for LFSR: W=30; L=500* 128

Figure D.2b): *Average Density Evolution of test patterns for LFSR: W=30; L=1,000* 128

Figure D.3a): *Density Evolution of test patterns for Rule 90/150 HCA: W=30; L=500* 129

Figure D.3b): *Average Density Evolution of test patterns for Rule 90/150 HCA: W=30; L=1,000* 129

Figure D.4a): *Density Evolution of test patterns for Rule 30 CA: W=30; L=500* 130

Figure D.4b): *Average Density Evolution of test patterns for Rule 30 CA: W=30; L=1,000* 130

Figure D.5a): *Density Evolution of test patterns for NLFSR weighted to 25% (AND-gate bank): [S]; W=29; L=500* 131

Figure D.5b): *Average Density Evolution of test patterns for NLFSR weighted to 25% (AND-gate bank): [S]; W=29; L=1,000* 131

Figure D.6a): *Density Evolution of test patterns for LFSR weighted to 25% (AND-gate bank): [S]; W=29; L=500* 132

Figure D.6b): *Average Density Evolution of test patterns for LFSR weighted to 25% (AND-gate bank): [S]; W=29; L=1,000* 132

Figure D.7a): *Density Evolution of test patterns for Rule 90/150 HCA weighted to 25% (AND-gate bank): [S]; W=29; L=500* 133

Figure D.7b): *Average Density Evolution of test patterns for Rule 90/150 HCA weighted to 25% (AND-gate bank): [S]; W=29; L=1,000* 133

Figure D.8a): *Density Evolution of test patterns for Rule 30 CA: weighted to 25% (AND-gate bank): [S]; W=29; L=500* 134

Figure D.8b): *Average Density Evolution of test patterns for Rule 30 CA*

weighted to 25% (AND-gate bank): [S]; W=29; L=1,000 134

Figure D.9a): *Density Evolution of test patterns for NLFSR weighted to 75% (OR-gate bank): [S]; W=29; L=500* 135

Figure D.9b): *Average Density Evolution of test patterns for NLFSR weighted to 75% (OR-gate bank): [S]; W=29; L=1,000* 135

Figure D.10a): *Density Evolution of test patterns for LFSR weighted to 75% (OR-gate bank): [S]; W=29; L=500* 136

Figure D.10b): *Average Density Evolution of test patterns for LFSR weighted to 75% (OR-gate bank): [S]; W=29; L=1,000* 136

Figure D.11a): *Density Evolution of test patterns for Rule 90/150 HCA weighted to 75% (OR-gate bank): [S]; W=29; L=500* 137

Figure D.11b): *Average Density Evolution of test patterns for Rule 90/150 HCA weighted to 75% (OR-gate bank): [S]; W=29; L=1,000* 137

Figure D.12a): *Density Evolution of test patterns for Rule 30 CA: weighted to 75% (OR-gate bank): [S]; W=29; L=500* 138

Figure D.12b): *Average Density Evolution of test patterns for Rule 30 CA weighted to 75% (OR-gate bank): [S]; W=29; L=1,000* 138

Figure D.13a): *Density Evolution of test patterns for NLFSR weighted to 25% (AND-gate bank): [J=0]; W=15; L=500* 139

Figure D.13b): *Average Density Evolution of test patterns for NLFSR weighted to 25% (AND-gate bank): [J=0]; W=15; L=1,000* 139

Figure D.14a): *Density Evolution of test patterns for LFSR weighted to 25% (AND-gate bank): [J=0]; W=15; L=500* 140

Figure D.14b): *Average Density Evolution of test patterns for LFSR weighted to 25% (AND-gate bank): [J=0]; W=15; L=1,000* 140

Figure D.15a): *Density Evolution of test patterns for Rule 90/150 HCA weighted to 25% (AND-gate bank): [J=0]; W=15; L=500* 141

Figure D.15b): *Average Density Evolution of test patterns for Rule 90/150 HCA weighted to 25% (AND-gate bank): [J=0]; W=15; L=1,000* 141

Figure D.16a): *Density Evolution of test patterns for Rule 30 CA: weighted to 25% (AND-gate bank): [J=0]; W=15; L=500* 142

Figure D.16b): *Average Density Evolution of test patterns for Rule 30 CA weighted to 25% (AND-gate bank): [J=0]; W=15; L=1,000* 142

Figure D.17a): *Density Evolution of test patterns for NLFSR weighted to*

75% (OR-gate bank): [J=0]; W=15; L=500 143

Figure D.17b): *Average Density Evolution of test patterns for NLFSR weighted to 75% (OR-gate bank): [J=0]; W=15; L=1,000* 143

Figure D.18a): *Density Evolution of test patterns for LFSR weighted to 75% (OR-gate bank): [J=0]; W=15; L=500* 144

Figure D.18b): *Average Density Evolution of test patterns for LFSR weighted to 75% (OR-gate bank): [J=0]; W=15; L=1,000* 144

Figure D.19a): *Density Evolution of test patterns for Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=0]; W=15; L=500* 145

Figure D.19b): *Average Density Evolution of test patterns for Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=0]; W=15; L=1,000* 145

Figure D.20a): *Density Evolution of test patterns for Rule 30 CA: weighted to 75% (OR-gate bank): [J=0]; W=15; L=500* 146

Figure D.20b): *Average Density Evolution of test patterns for Rule 30 CA weighted to 75% (OR-gate bank): [J=0]; W=15; L=1,000* 146

Figure D.21a): *Density Evolution of test patterns for LFSR weighted to 25% (AND-gate bank): [J=14]; W=15; L=500* 147

Figure D.21b): *Average Density Evolution of test patterns for LFSR weighted to 25% (AND-gate bank): [J=14]; W=15; L=1,000* 147

Figure D.22a): *Density Evolution of test patterns for Rule 90/150 HCA weighted to 25% (AND-gate bank): [J=14]; W=15; L=500* 148

Figure D.22b): *Average Density Evolution of test patterns for Rule 90/150 HCA weighted to 25% (AND-gate bank): [J=14]; W=15; L=1,000* 148

Figure D.23a): *Density Evolution of test patterns for LFSR weighted to 75% (OR-gate bank): [J=14]; W=15; L=500* 149

Figure D.23b): *Average Density Evolution of test patterns for LFSR weighted to 75% (OR-gate bank): [J=14]; W=15; L=1,000* 149

Figure D.24a): *Density Evolution of test patterns for Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=14]; W=15; L=500* 150

Figure D.24b): *Average Density Evolution of test patterns for Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=14]; W=15; L=1,000* 150

Appendix F: Histogram Evolutions

Figure F.1a): *Binomial Coefficient Combination Profile: W=30* 159

Figure F.1b): *Binomial Probability Mass Function (50%): W=30* 159

Figure F.2a): *Binomial Coefficient Combination Profile: W=29* 160

Figure F.2b): *Binomial Probability Mass Function (25% & 75%): W=29* 160

Figure F.3a): *Binomial Coefficient Combination Profile: W=15* 161

Figure F.3b): *Binomial Probability Mass Function (25% & 75%): W=15* 161

Figure F.4: *Probability Mass Function (Histogram) of NLFSR (50%): W=30* 162

Figure F.5: *Probability Mass Function (Histogram) of LFSR (50%): W=30* 162

Figure F.6: *Probability Mass Function (Histogram) of Rule 90/150 HCA (50%): W=30* 163

Figure F.7: *Probability Mass Function (Histogram) of Rule 30 CA (50%): W=30* 163

Figure F.8: *Probability Mass Function (Histogram) of NLFSR weighted to 75% (OR-gate bank): [S]; W=29* 164

Figure F.9: *Probability Mass Function (Histogram) of LFSR weighted to 75% (OR-gate bank): [S]; W=29* 164

Figure F.10: *Probability Mass Function (Histogram) of Rule 90/150 HCA weighted to 75% (OR-gate bank): [S]; W=29* 165

Figure F.11: *Probability Mass Function (Histogram) of Rule 30 CA weighted to 75% (OR-gate bank): [S]; W=29* 165

Figure F.12: *Probability Mass Function (Histogram) of NLFSR weighted to 75% (OR-gate bank): [J=0]; W=15* 166

Figure F.13: *Probability Mass Function (Histogram) of LFSR weighted to 75% (OR-gate bank): [J=0]; W=15* 166

Figure F.14: *Probability Mass Function (Histogram) of Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=0]; W=15* 167

Figure F.15: *Probability Mass Function (Histogram) of Rule 30 CA*

weighted to 75% (OR-gate bank): [J=0]; W=15 167

Figure F.16: *Probability Mass Function (Histogram) of LFSR weighted to 75% (OR-gate bank): [J=14]; W=15* 168

Figure F.17: *Probability Mass Function (Histogram) of Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=14]; W=15* 168

Appendix G: Magnitude Spectrums

Figure G.1: *Magnitude Spectrum of NLFSR (50%): W=30; L=8,192* 170

Figure G.2: *Magnitude Spectrum of LFSR (50%): W=30; L=8,192* 170

Figure G.3: *Magnitude Spectrum of Rule 90/150 HCA (50%): W=30; L=8,192* 171

Figure G.4: *Magnitude Spectrum of Rule 30 CA (50%): W=30; L=8,192* 171

Figure G.5: *Magnitude Spectrum of NLFSR weighted to 75% (OR-gate bank): [S]; W=29; L=8,192* 172

Figure G.6: *Magnitude Spectrum of LFSR weighted to 75% (OR-gate bank): [S]; W=29; L=8,192* 172

Figure G.7: *Magnitude Spectrum of Rule 90/150 weighted to 75% (OR-gate bank): [S]; W=29; L=8,192* 173

Figure G.8: *Magnitude Spectrum of Rule 30 CA weighted to 75% (OR-gate bank): [S]; W=29; L=8,192* 173

Figure G.9: *Magnitude Spectrum of NLFSR weighted to 75% (OR-gate bank): [J=0]; W=15; L=8,192* 174

Figure G.10: *Magnitude Spectrum of LFSR weighted to 75% (OR-gate bank): [J=0]; W=15; L=8,192* 174

Figure G.11: *Magnitude Spectrum of Rule 90/150 weighted to 75% (OR-gate bank): [J=0]; W=15; L=8,192* 175

Figure G.12: *Magnitude Spectrum of Rule 30 CA weighted to 75% (OR-gate bank): [J=0]; W=15; L=8,192* 175

Figure G.13: *Magnitude Spectrum of LFSR weighted to 75% (OR-gate bank): [J=14]; W=15; L=8,192* 176

Figure G.14: *Magnitude Spectrum of Rule 90/150 weighted to 75% (OR-gate bank): [J=14]; W=15; L=8,192* 176

Appendix H: Space-Phase Correlation

Figure H.1: *Space-phase Cross-correlation of NLFSR (50%): W=30; L=10,000; Reference n=0* 178

Figure H.2: *Space-phase Cross-correlation of LFSR (50%): W=30; L=10,000; Reference n=0* 178

Figure H.3a): *Space-phase Cross-correlation of Rule 90/150 HCA (50%): W=30; L=10,000; Reference n=0* 179

Figure H.3b): *Space-phase Cross-correlation of Rule 90/150 HCA (50%): W=30; L=10,000; Reference n=1* 179

Figure H.3c): *Space-phase Cross-correlation of Rule 90/150 HCA (50%): W=30; L=10,000; Reference n=2* 180

Figure H.4: *Space-phase Cross-correlation of Rule 30 CA (50%): W=30; L=10,000; Reference n=0* 180

Figure H.5: *Space-phase Cross-correlation of NLFSR weighted to 75% (OR-gate bank): [S]; W=29; L=10,000; Reference n=0* 181

Figure H.6: *Space-phase Cross-correlation of LFSR weighted to 75% (OR-gate bank): [S]; W=29; L=10,000; Reference n=0* 181

Figure H.7a): *Space-phase Cross-correlation of Rule 90/150 HCA weighted to 75% (OR-gate bank): [S]; W=29; L=10,000; Reference n=0* 182

Figure H.7b): *Space-phase Cross-correlation of Rule 90/150 HCA weighted to 75% (OR-gate bank): [S]; W=29; L=10,000; Reference n=1* 182

Figure H.7c): *Space-phase Cross-correlation of Rule 90/150 HCA weighted to 75% (OR-gate bank): [S]; W=29; L=10,000; Reference n=2* 183

Figure H.7d): *Space-phase Cross-correlation of Rule 90/150 HCA weighted to 75% (OR-gate bank): [S]; W=29; L=10,000; Reference n=3* 183

Figure H.8: *Space-phase Cross-correlation of Rule 30 CA weighted to 75% (OR-gate bank): [S]; W=29; L=10,000; Reference n=0* 184

Figure H.9: *Space-phase Cross-correlation of NLFSR weighted to 75% (OR-gate bank): [J=0]; W=15; L=10,000; Reference n=0* 184

Figure H.10: *Space-phase Cross-correlation of LFSR weighted to 75% (OR-gate bank): [J=0]; W=15; L=10,000; Reference n=0* 185

Figure H.11a): *Space-phase Cross-correlation of Rule 90/150 HCA weighted*

to 75% (OR-gate bank): [J=0]; W=15; L=10,000; Reference n=0 185

Figure H.11b): *Space-phase Cross-correlation of Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=0]; W=15; L=10,000; Reference n=1* 186

Figure H.11c): *Space-phase Cross-correlation of Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=0]; W=15; L=10,000; Reference n=2* 186

Figure H.11d): *Space-phase Cross-correlation of Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=0]; W=15; L=10,000; Reference n=6* 187

Figure H.12: *Space-phase Cross-correlation of Rule 30 CA weighted to 75% (OR-gate bank): [J=0]; W=15; L=10,000; Reference n=0* 187

Figure H.13a): *Space-phase Cross-correlation of LFSR weighted to 75% (OR-gate bank): [J=14]; W=15; L=10,000; Reference n=0* 188

Figure H.13b): *Space-phase Cross-correlation of LFSR weighted to 75% (OR-gate bank): [J=14]; W=15; L=10,000; Reference n=6* 188

Figure H.14a): *Space-phase Cross-correlation of Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=14]; W=15; L=10,000; Reference n=0* 189

Figure H.14b): *Space-phase Cross-correlation of Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=14]; W=15; L=10,000; Reference n=1* 189

Figure H.14c): *Space-phase Cross-correlation of Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=14]; W=15; L=10,000; Reference n=2* 190

Appendix I: Bit Sequence Tuple Profiles

Figure I.1: *Bit Sequence Tuple Profile of NLFSR weighted to 75% (OR-gate bank): [J=0], [J=14]; W=15; L=8,192; Reference n=0* 192

Figure I.2: *Bit Sequence Tuple Profile of LFSR weighted to 75% (OR-gate bank): [J=0], [J=14]; W=15; L=8,192; Reference n=0* 192

Figure I.3a): *Bit Sequence Tuple Profile of Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=0]; W=15; L=8,192; Reference n=(0,1,2)* 193

Figure I.3b): *Bit Sequence Tuple Profile of Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=0]; W=15; L=8,192; Reference n=(3,4,5)* 193

Figure I.3c): *Bit Sequence Tuple Profile of Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=0]; W=15; L=8,192; Reference n=(6,7,8)* 194

Figure I.4a): *Bit Sequence Tuple Profile of Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=14]; W=15; L=8,192; Reference n=(0,1,2)* 194

Figure I.4b): *Bit Sequence Tuple Profile of Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=14]; W=15; L=8,192; Reference n=(3,4,5)* 195

Figure I.4c): *Bit Sequence Tuple Profile of Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=14]; W=15; L=8,192; Reference n=(6,7,8)* 195

Figure I.5: *Bit Sequence Tuple Profile of Rule 30 CA weighted to 75% (OR-gate bank): [J=0], [J=14]; W=15; L=8,192; Reference n=0* 196

Figure I.6: *Concatenated Bit Sequence Tuple Profiles for finite state machines weighted to 75% (OR-gate bank): [S]; W=29; L=1,092* 196

Figure I.7: *Concatenated Bit Sequence Tuple Profiles for finite state machines weighted to 75% (OR-gate bank): [J=0]; W=15; L=1,092* 197

Figure I.8: *Concatenated Bit Sequence Tuple Profiles for finite state machines weighted to 75% (OR-gate bank): [J=14]; W=15; L=1,092* 197

Chapter 1
Introduction

Associated with the recent advancements in VLSI technology is the ensuing problem of testability. To circumvent the difficulty encountered when providing for testability, there has been a concentrated effort to establish new and improved Design For Testability (DFT) methodologies. A most promising and active field of DFT is that of Built-In Self-Test (BIST). Because this strategy has the capability of infiltrating networks-under-test, right at the chip-level, there is a greater likelihood that any potentially expensive defects will be exposed before the cost of verification is allowed to escalate.

One BIST strategy which has emerged and is gaining wide acceptance, because of its ability to conform to the restrictions imposed upon by VLSI, and its comparatively low fault detection cost, is Pseudorandom Test Pattern Generation (PTPG). There are, however, some detriments to the incorporation of this technique: the largest being its inability to exercise pseudorandom-resistant circuits. Fortunately, there is an approach known as Weighted Test Pattern Generation (WTPG), which was initially introduced in an attempt to expose the faults contained in pseudorandom-resistant circuits. Although there are many ways to ''weight'' typically ''unbiased'' pseudorandom test patterns, there are certain statistical attributes which must be maintained to ensure testability.

The basic motivation of this thesis is to investigate alternative test pattern generation techniques, which may prove conducive as integration levels increase. Specifically, it is of particular interest to generate probabilistically weighted test

patterns with improved statistical, and, hence, fault detection capabilities. The majority of the effort is directed on a scheme which involves supplying unbiased pseudorandom patterns, from a discrete-time stochastic source, to an array of logic gates performing a weighting operation. The main focus is on using Cellular Automata (CA), as opposed to Linear Feedback Shift Registers (LFSRs), as primary ''driving engines'' for the derivation of high quality Unbiased Pseudorandom Test Pattern Generation (UPTPG). But, because, they too evolve around a deterministic statistical evolution mechanism, only a certain degree of ''apparent'' statistical independence is, at best, attainable in the associated Weighted Test Pattern Generator (WTPG) function. However, it is anticipated that this level of behaviour should suffice for many BIST applications. Some of the arguments which are believed to be germane in building ''good'' WTPGs are discussed, especially those pertaining to statistical characteristics, wiring complexities, and performance. The accompanying analysis which follows is intended to provide support to these initial suppositions.

The beginning few chapters are intended to provide some background information generalizing and categorizing DFT in VLSI. By abstracting some general points, a ''grounding'' is, in effect, laid out which will shed some light on the ''core'' of the thesis, that being WTPG. Chapter 2 (Design and Test of VLSI) presents an overview to the field of VLSI, from the design outset to testing, and Chapter 3 (Design for Testability) discusses some of the present, and past, DFT methodologies. The emphasis is on the Built In Logic Block Observer (BILBO), due to its ability to conform to BIST implementations. Chapter 4 details the effectiveness of pseudorandom testing, with the assumption of a single stuck-at fault model. Test quality measures of Chin, McCluskey, and Wagner (see [Chin87] and [Wagner87]), and there differences with respect to pseudorandom as opposed to purely random test modelling, are discussed. Chapter 5 (Cellular Automata-An overview) is an overview of the theory of Cellular Automata (CA). Included is a discussion on one-dimensional CA characterizations of class and

behaviour. It is the sole intention of this chapter to overview the various classes of behaviour attainable by such simply constructed machines, and their possible utilization.

Chapter 6 (Weighted Test Pattern Generation for Built-In Self-Test using Cellular Automata), as mentioned previously, is the core of the thesis. Outlined are some of the potential architectures for WTPG, and their associated behaviours. Examination of the proposed class of WTPGs are made possible by some common statistical estimators. Also mentioned are some of the potential hazards of using known ''biased'' estimators, and how confidence intervals are consequently affected. There are several aspects of WTPG behaviour investigated. The six which were considered as having significant impact on the rating of such generators include: state-time visualization; density and average density; probability mass function (histogram); space-phase cross correlation; and, bit sequence tuple lengths. Extensive results are included comparing CA based WTPGs with those based on the LFSR.

Chapter 7 (Cellular Automata based Testing Apparatus) presents a variation of the well known BILBO, known as a CALBO, so as to achieve higher levels of fault coverage with the added benefit of improved circuit performance. Because of the importance of WTPG to BIST, an implied Weighted CALBO (WCALBO) is also justified.

Finally, in Chapter 8 (Conclusions and Future Work), some inferences are made in regards to the practicality of the proposed class of WTPGs. Also mentioned, in closing, are some ideas on future directions for WTPG design, and variations on implementation.

Chapter 2
Design and Test of VLSI

2.1. Structured Design Methodologies

With the advent of Very Large Scale Integration (VLSI), design complexities have emerged which compel design engineers to adopt structured design methodologies. Structured approaches to design offer the means by which problems, existing in present and future VLSI developments, may be dealt with. These approaches include hierarchical, modular, regular, and local design strategies at the integrated circuit layout level (see [Weste85]).

The use of *hierarchy* in design layout reduces the complexity of modules by dividing these further into sub-modules, as to create sufficiently workable levels of detail. A direct consequence of hierarchical design is the creation of ''well-formed'' modules. *Modularity* of design generates easily managed constructs with well defined physical interfaces; that is, references to position, name, layer, size and signal type, of any external interconnections, such as power, ground, inputs, and outputs, is implied in each module. The final placement of the external interfaces of each module is crucial in providing for the regularity of structures.

Hierarchical and Modular design techniques, associated with the development of semi-custom integrated circuits, have been incorporated to rid the designer from the burden of low-level circuit design [Aylor86]. The result of this design exercise, from the perspective of a digital designer, is that previously designed macros, or modules,

may be used to develop complex integrated circuits. Moreover, several designs, from many different designers, may be grouped together to form an entire cell library. This tends to aid the digital designer, with limited knowledge of integrated circuit design and fabrication, in the task of custom design. Such a design environment has proven successful at the University of Manitoba's VLSI Design Lab, where a custom CMOS library provides a strong foundation for which to build from, especially for the strictly digital designer. Also, developmental attributes allow for several designs, from many designers, to be fabricated on a single silicon wafer as a way of reducing prototype fabrication costs. This approach to implementation is practised at the Canadian Microelectronics Corporation's VLSI Implementation Centre (VLSIIC) facility.[2.1]

2.2. Testability of VLSI

Along with the advances of integrated circuit technology has occurred problems of determining, in a cost-effective manner, whether or not a component has been successfully manufactured. It has been established that the cost associated with integrated circuit testing grows proportionally, according to a cubic power relation, to the total number of devices on an integrated circuit [Feugate88]. With this relation in mind, it is not surprising that the cost of manufacturing is steadily decreasing with respect to the cost of testing.

Recently, concern has arisen that the limit to the attainable functional density of an integrated circuit will not be substantiated by any physical barrier, but will, rather, occur as a result of the integrated circuit becoming untestable. The point at which a

[2.1] Location: Canadian Microelectronics Corporation
Carruthers Hall, Queen's University
Kingston, Ontario, Canada K7L 3N6

component can no longer be confirmed free of any faults is the point at which it is untestable [Aylor86]. An interesting metric depicting the cost incurred by not adopting testability at the circuit level is as follows: if a fault goes undetected at the circuit level, it will cost one order of magnitude, two orders of magnitude, and three orders of magnitude, at the printed circuit board level, system level, and in the field, respectively, times the fault detection cost involved at the circuit level. It is, therefore, necessary for the design engineer to think as a test engineer, and vice-versa, so that their duties will overlap to acknowledge testability as a means of increasing production yield.

Test generation, test verification, and design for testability are three main areas which comprise testability in VLSI. *Test generation* refers to the process of producing test stimuli to establish the correctness of a circuit's behaviour; whereas, *test verification* attempts to find measures of the effectiveness of the actual test stimuli (i.e., fault coverage performance, etc.) resulting from the test generation process. Finally, *design for testability* (DFT) involves designing test circuits, from the outset, to be used in conjunction with existing hardware for the purpose of ensuring testability.

2.2.1. Test Generation and Verification

It is of great importance for the test engineer to reduce the difficulty of test generation. This can be accomplished by considering the testability of a circuit early in the design process and making the necessary adjustments to improve its testability. *Testability analysis* is a method which assists in the identification of areas of poor testability for the commitment of improvement. To do so requires a *testability measure* to quantify the testability of a circuit. Furthermore, the accuracy and information content of the testability measure must be adequate enough to justify the redesigning of a circuit as a way of improving its testability [Fujiwara85].

There are two testability measures which are available to provide estimates of testability for particular circuits: *controllability* and *observability*. The degree to which internal circuitry, containing low external accessibility, can be controlled from its primary inputs, is a measure of its controllability. Because test generation requires the ability to present arbitrary test sets to certain portions of a network, at different instances of time, good controllability is necessary. Just as important to test generation is the concept of good observability. Observability measures the ease with which internal circuitry can be observed at its primary outputs so that circuit verification, in the form of pass or fail, can be made.

2.2.2. Design For Test Methodology

Including DFT features involves modifying a design according to some criterion. This criterion is guided by various design goals and constraints. They include: area overhead, effects on circuit speed, fault coverage, number of I/O pins, time needed for testing, and availability of testing equipment [Abadir85]. Basically, the relative cost of implementing DFT depends on the magnitude of the desired level of testability. Because there is no comprehensive theory relating design constraints with levels of testability, immediate assessment of the consequences of design decisions is rather difficult. Thus, the rules of the DFT criterion are somewhat flexible and subjective.

2.2.3. Fault Modelling

A fault model which has proven to be quite successful, in practise, is the *stuck-at-fault model*. This simple but effective fault modelling device attempts to model different physical faults, resulting from the fabrication process, by assuming lines in a logic model to be permanently stuck at the 0 or 1 level.

For circuits containing just one fault, a *single stuck-at-fault model* is often used. If there are K lines in a logical circuit, then there are $2K$ possible single stuck-at-faults. For circuits with more than one fault, a multiple stuck-at-fault model is used. In this case, there are $3^K - 1$ possible faults since any line may be fault free, stuck-at-0, or stuck-at-1[2.2] [Fujiwara85].

A stuck-at-fault model proves quite effective when *line bridging* to ground or power lines cause errors. For situations where *open failures* cause intermittent faults, the simple stuck-at-fault model can no longer prove to be as reliable. In such circumstances, a more complicated model is necessary to ensure a high degree of testability. As often is the case with CMOS, an open error may cause both pull-up and pull-down sections to be nonconductive. This can place the output mode into a floating state which will cause the previous value to be retained. As a result, a once combinational circuit is turned into a sequential one. For this case, it is necessary to use a *stuck-open-fault model* rather than a stuck-at-fault model, since the former takes into account the time dependent nature of a faulty circuit (see [Sudhakar86]). The exhaustive testing, under the stuck-open fault model, of an n-input CMOS combinational circuit has been shown to require the application of $n \cdot 2^{n+1}$ test patterns [Bate87]. This represents rather a large increase in number over that associated with the single stuck-at-fault model.

The simple single stuck-at-fault model has maintained wide acceptance as a basis for modelling faults in combinational networks, due to its ability to model intrinsic faults at a logic level. Also, to a lesser extent, it makes possible the statistical analysis of test quality measures for pseudorandom testing and allows for the application of test set reduction algorithms (eg., the D-algorithm). When a single stuck-at-fault model is

[2.2] The factor (-1) is included in the expression so as to exclude the possibility of having no faults at all.

used, a test vector which detects one particular fault may, generally speaking, detect many other stuck-at-faults as well. For instance, it has been determined by Agarwal and Fung [Agarwal81], that for every complete single fault detection test set in any internal fan-out free combinational circuit, at least 98% of all multiple faults made up of six or fewer faults are covered. This is because many multiple faults are functionally equivalent to single faults and are thus readily detectable. It has further been conjectured that other kinds of faults, besides the stuck-at type, may manifest themselves detectable by the application of a test set devised solely from a single stuck-at-fault model.

Chapter 3
Design For Testability

3.1. Introduction to DFT

Categorically, there are three techniques which comprise DFT: ad hoc, structured, and self-test. The *ad hoc* approach to DFT has, until recently, been the simplest and least expensive means of providing for testability. It is contained at the board level, and is, therefore, heuristic rather than systematic in nature [Fujiwara85]. Since this method does not directly investigate on-chip circuits, and is not generally applicable to all designs, test generation and fault simulation is in most respects difficult. Some of the more common ad hoc methods of testability include partitioning, test-points, and signature analysis.

On the other hand, *structured* approaches attempt to deal with the problems of testability by applying general design methodologies. In doing so, a systematic set of rules employed early in the design stage will, in essence, guarantee both controllability and observability. Structured DFT methods achieve higher degrees of controllability and observability by introducing testability right at circuit design time. This practise ensures on-chip circuit testing which was, otherwise, unattainable by conventional ad hoc testing techniques. By allowing internal latch variables to be controlled and observed, the testing of sequential networks can be altered to that of testing combinational networks. Thus, once configured into a stable combinational form, a network can be applied test sets (controllability), from a test generation algorithm, while the

resulting data is verified (observability). Structured techniques are well established and are widely used in practise. There are many implementations and variations, all going by different names, but similar, nevertheless: IBM calls it Level Sensitive Scan Design (LSSD); NEC calls it Scan Path; Unisys (formerly, Sperry-Univac division) calls it Scan/Set Logic; and, Fujitsu calls it Random-access Scan [Williams84].

3.2. Built-In Self-Test approach to DFT

The Built-In Self-Test (BIST) approach to DFT is based on self-contained test application methods. Self-testing generally involves internal testing practises, where test patterns are applied and the resulting responses compacted, without the use of external test equipment. (This is referred to as a BIST strategy.) Prior to the development of this technique, available conventional structured DFT methods relied on *off-line* external test methods [Fujiwara85, Williams86]. When incorporated for sequential circuit testing, test patterns delivered to the network-under-test were generated from an external source. Furthermore, all test response data had to be compared with golden data, also external to the network-under-test, for substantiation of correctness. The time taken to scan the test vectors in, and the network responses out, is, in itself, enough of an incentive to find alternative means for achieving testability.

There are several advantages to be gained by employing BIST methods. The more important and readily apparent are the following: huge amounts of test patterns and responses are no longer required to be stored for subsequent comparison; testing speed is significantly increased; external pin count, dedicated for testing, is drastically reduced; and, several networks may be simultaneously tested. Because of these attributes, BIST techniques have become one of the most promising of all the DFT methods. As VLSI advances, large complex circuits with high speed operation will

necessitate new developments in BIST structures.

BIST structures are designed to provide the entire testing mechanism for the system-under-test. Some structures are included for the testing of strictly combinational networks, while other structures replace system latches in order to transform a difficult-to-test sequential network into a readily testable combinational network. The methods of testing addressed here assume that the networks-under-test are either entirely combinational or have original sequential circuit functions, which may be broken up into its system latches and combinational circuitry.

BIST forms of testing are known as Insitu Self-testing. This technique is exclusively dedicated to on-chip testing and abide by the assumption of sequential circuit testing stated previously. Insitu structures often use system latches to participate in both test pattern application and data compaction. In light of this, BIST techniques facilitate improvements in controllability and observability beyond those attainable by adopting either ad hoc or structured DFT methods. Consequently, they are included in much of the current DFT research.

In Insitu Self-testing, the system latches of a network-under-test are used in conjunction with test pattern generation and signature analysis. A popular Insitu method which incorporates both aspects, within the same structure, is Built In Logic Block Observation. Although there are a wide variety of other Insitu techniques which reap the benefits of pseudorandom test pattern generation, Built In Logic Block Observation is not exclusively dedicated to exhaustive testing as are many of the others. In fact, it is normally used for those instances where exhaustive testing is inappropriate. Because of this, it serves best to demonstrate the advantages of BIST. Thus, all principals applied to this technique, herein contained in this work, are also equally applicable to those not specifically addressed.

3.2.1. Parallel Signature Analysis

For on-chip circuit testing, it is far more practical to propose *parallel* rather than serial signature analysis. To meet the demands of testing dense VLSI circuits, with large numbers of primary inputs and outputs, Multiple Input Signature Registers (MISRs) were developed (see [David86]). These structures offer the same capabilities as the better known serial signature analyzers, but are more efficient. The realization of MISRs significantly reduce logic overhead and increases the rate at which data compaction may be accomplished. Simply stated, MISRs conform to VLSI better than their singular predecessors.

3.2.2. Built In Logic Block Observation

A Testing technique that combines Scan Path testing with signature analysis is called Built In Logic Block Observation. This technique is based entirely on the Built In Logic Block Observer, or BILBO. The BILBO is a multi-functional register designed with conventional digital logic components. There are two similar implementations which are most often used. One consists of an assortment of D-type flip-flops (D), NOR, EXOR, and AND gates, and a 2X1 multiplexer (M) (see Figure 3.1). The other has, as its only difference in design, the placement of OR instead of NOR gates. Both versions have two select lines, S_1 and S_2, several inputs $\{I_1, I_2, \ldots, I_n\}$ and outputs $\{Q_1, Q_2, \ldots, Q_n\}$ (for an n-bit BILBO), and a scan-in, D_{in}, and scan-out, D_{out}, line.

There are four different functional modes the BILBO of Figure 3.1, using select lines S_1 and S_2, may be configured:

- First, a *parallel latch mode* allows for normal system operation. When $S_1 = 1$ and $S_2 = 1$, the inputs $\{I_1, I_2, \ldots, I_n\}$ may be latched in so that their values are

retained at the outputs $\{Q_1, Q_2, \ldots, Q_n\}$.

- Secondly, When $S_1 = 0$ and $S_2 = 0$, a *scan path mode* is created via a linear shift register, thereby, allowing data to be scanned into the primary input, D_{in}, and scanned out the primary output D_{out}.
- Thirdly, an *MISR/PRNG mode* is established when $S_1 = 1$ and $S_2 = 0$. In this mode, the BILBO is turned into a maximal-length parallel LFSR. Thus, it can act as a parallel signature analyzer or maximal-length pseudorandom number generator (PRNG).[3.1] In order to separate one function from another, the inputs $\{I_1, I_2, \ldots, I_n\}$ must be controllable.
- Finally, the fourth function mode of the BILBO is the *reset*. When $S_1 = 0$ and $S_2 = 1$, all of the D-type flip-flops are cleared to zero; that is, $Q_i = 0$ for $i \in \{1, 2, \ldots, n\}$. This is used as an option prior to the commencement of serial or parallel signature analysis.

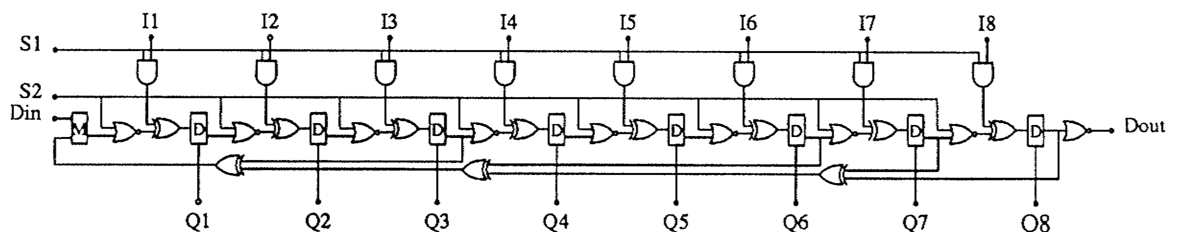

Figure 3.1: *Built In Logic Block Observer (BILBO) register with four mode functionality: W=8; M=2X1 multiplexer; D=D-flip flop.*

[3.1] Pseudorandom sequences are produced by deterministically generated, therefore, cyclic random sequences. Bardell and McAnney [Bardell86] have shown that a maximal length LFSR is a poor choice for a parallel PRNG.

There are a variety of self-testing schemes where the BILBO may be applied. For instance, BILBO testing may be used in a similar capacity to that of LSSD (see Figure 3.2). The BILBO structure simply replaces the Shift Register Latches (SRLs) in the feedback of the sequential network-under-test. To test the combinational logic, the BILBO is initially configured as a linear shift register, by placing $S_1 = 0$ and $S_2 = 0$ so that an initial test pattern may be loaded into its registers. Then the BILBO is configured as an MISR by placing $S_1 = 1$ and $S_2 = 0$. In this mode, parallel signature analysis may be performed on the combinational logic, since pseudorandom test patterns are applied to the input while the corresponding circuit responses are compacted. After some predetermined number of test patterns, or clock cycles, the BILBO is once again turned into a linear shift register, so that the final signature may be scanned out and checked for good machine compliance.

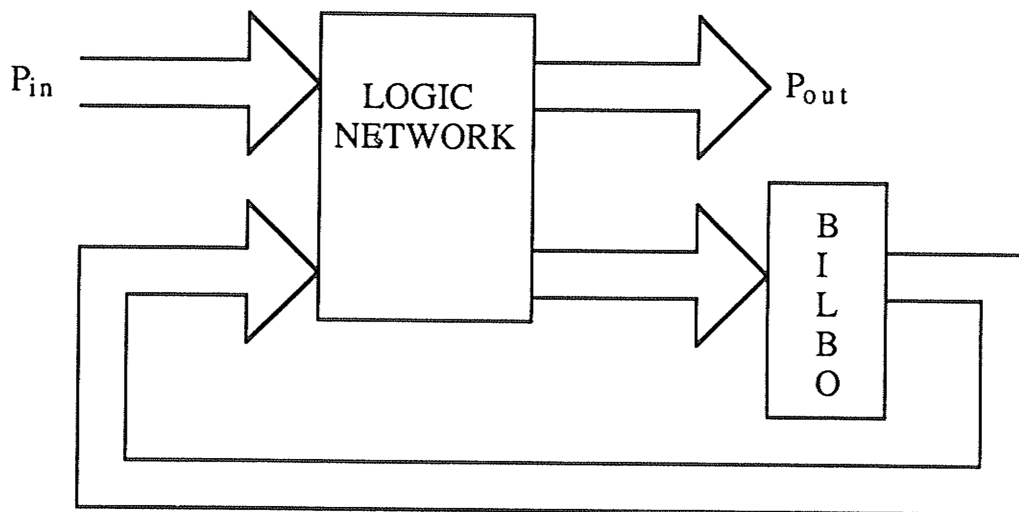

Figure 3.2: *Sequential Network Testing strategy using Built In Logic Block Observation.*

Because of the favorable testing capability a BILBO has to offer, it may also be used in the testing configuration depicted by Figure 3.3. In order to test combinational Network 1, BILBO 1 must be used as a PRNG to provide test patterns into combinational Network 2, while BILBO 2 performs the parallel signature analysis. Likewise, in order to test combinational Network 2, BILBO 2 must be used as a PRNG to provide test patterns into combinational Network 2, while BILBO 1 performs the parallel signature analysis. Additional control circuitry must be included so that when a BILBO is required to function as a PRNG, all of its inputs are maintained isolated from the outputs of the previous combinational network/BILBO stage (i.e., $I_i = 0$ for $i \in \{1, 2, \ldots, n\}$ is necessary for a BILBO to act as a PRNG).

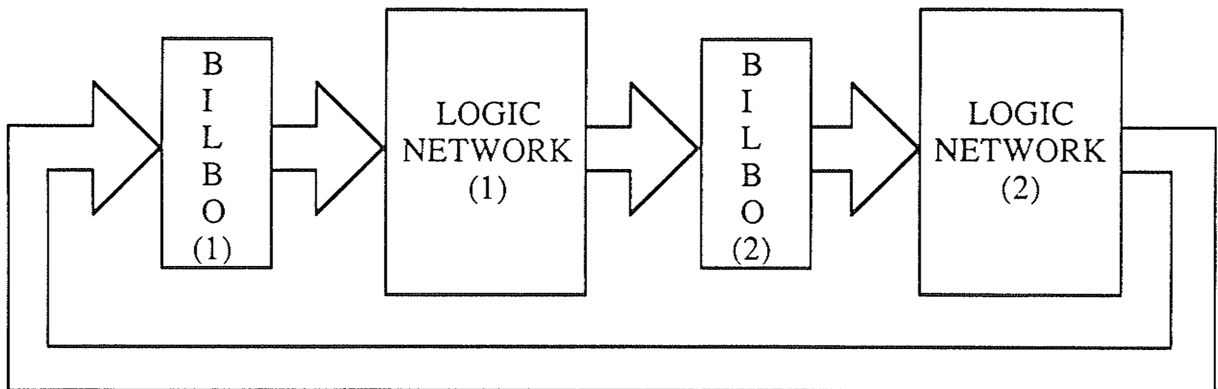

Figure 3.3: *Concatenated Network Testing strategy using Built In Logic Block Observation.*

In a different test setting, a BILBO may be introduced to modular and bus-oriented systems in which functional modules, such as ALUs, RAMs, and ROMs, and other I/0 devices, are interconnected by bus routing (see Figure 3.4) [Fujiwara85]. In such an environment, the BILBOs are used to take the place of clocked latches which

are located to interface each module. Every module requires two BILBOs, one to issue pseudorandom test patterns, and the other to perform parallel signature analysis. The BILBOs of one module are linked together with adjacent BILBOs in another module, via the scan input and output lines, in a parallel fashion. With this configuration, every BILBO in the chain may be loaded or unloaded with initial conditions or signatures, respectively, in one operation.

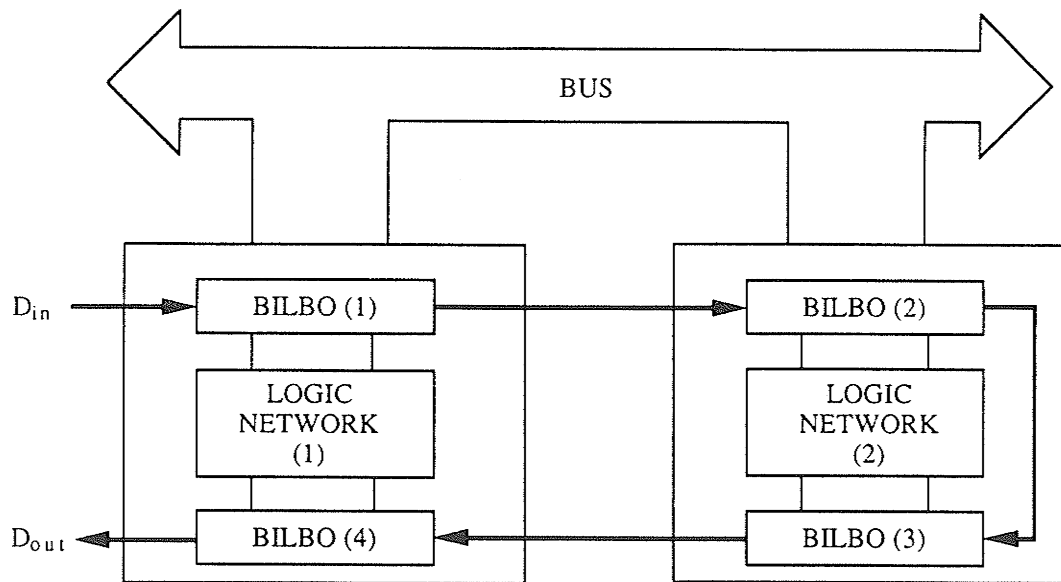

Figure 3.4: *Bus-Oriented Testing strategy using Built In Logic Block Observation.*

Because, in general, combinational logic is highly susceptible to pseudorandom test patterns, the BILBO technique proves very favorable when applied to sequential or combination network testing. If it is known that a network-under-test is pseudorandom testable, then using a BILBO to exercise the test will rid the test engineer of some of

the burden of test generation and fault simulation. When compared to structured DFT techniques, significant advantages are offered by employing the BILBO approach to DFT. Of all the advantages, the most predominant one is that of testing speed. The speed with which testing may be accomplished is many orders of magnitude faster (see [Williams84]). The problem with the BILBO is that many more patterns need be applied to substantiate a certain level of confidence in testability. Even though the Scan Path techniques may not involve as many test patterns, to attain the same level of confidence, the over all time required to conduct the test is much longer. It can be said, with controversy, no doubt, that the test application of the BILBO approach to DFT is significantly less expensive, when one considers the immense cost associated with large testing times and undetected faults at the chip level.

Chapter 4
Circuit Testability using Random Test Pattern Generation

In situations where exhaustive testing is prohibitively time consuming, an alternative means of providing for testability involves the application of only a subset of all possible input test patterns. This is particularly attractive for circuits which contain greater than twenty inputs (i.e., $n > 20$) [Weste85]. Furthermore, because test pattern generation and fault simulation is both expensive and time-consuming, the subset of test patterns may be generated randomly by some kind of inexpensive built-in hardware. The appropriateness of random[4.1] testing developed from the concept that an arbitrary test set may exercise a circuit-under-test, with a high probability of providing reasonable assurance that the circuit is in fact fault free.

For a random test to be practical, the test length must be significantly less than that of an exhaustive test. However, it is expected to be larger in length, for the same fault coverage, than that determined by some test generation algorithm. It is up to the test engineer to balance the savings of smaller development time and cost with the increase in test length [Wagner87].

Combinational logic networks have shown to be quite testable by randomly derived test sets. Random combinational logic networks, with relatively low fan-in, are amongst the most susceptible, of all combinational networks, to random patterns.

[4.1] The word "random," as used in this section, refers to both purely random, or strictly random, and pseudorandom, and is considered "unbiased." The distinction between the two will be described later on in this chapter.

Other forms of combinational networks are not necessarily suitable for random testing, since they may contain sections which are only testable by a meager set of input vectors. Moreover, networks with large fan-in, such as programmable logic arrays (PLAs), are not tested very effectively by random patterns. A simple NAND gate supports these findings. For an n-input NAND gate, with a stuck-at-0 fault at one of its inputs, there is only one input pattern, $111 \cdots 1$, which will exercise the fault. If an "unbiased" random test set is chosen to test this gate, the detection probability of the fault is simply $1/2^n$. It is obvious that the detection probability will be rather small for large n, that is, for a large fan-in, thereby making conventional unbiased random testing unsuitable for this case. Conversely, in general, for smaller n, or lower fan-in, the detection probability becomes significantly higher, making the gate a possible candidate for random testing.

The BIST environment has adopted random testing largely due to two main reasons: i) it offers generous fault detection capability, for some circuits, at low cost; and ii), circuits used to generate random test patterns conform quite well to VLSI. Random testing has also shown to be somewhat effective in detecting typically hard-to-detect intermittent faults (see [Savir80]).

4.1. Purely Random versus Pseudorandom Testing

Purely random generated test sets are the result of a "sampling with replacement" procedure. This ensures that every test vector may be a candidate of the test set with an equal probability of being selected at any given time. In a similar fashion, *pseudorandom* generated test sets are formed by a "sampling without replacement" procedure. The latter procedure, at best, is capable of generating non-repeatable patterns within deterministic cycles of maximal-length. Hence, pattern repetition is only

possible for periods extending beyond maximal-length, since a complete enumeration of test vectors are produced only for every maximal-length cycle. Circuits which generate pseudorandom maximal-length cycles, such as LFSRs, are most often used in BIST structures because they are entirely deterministic, and thus easy to implement.

Past methods of testability analysis relied on the purely random test model of test pattern generation for determining measures of test quality. More recently, Chin and McCluskey [Chin87] have found that the pseudorandom test model is always superior to the random test model. It is responsible for producing more accurate results, shorter test length estimates for a selected test quality, and better test quality results for a given test length.

Using a single stuck-at-fault model, some past methods, based on the purely random test model, predicted test lengths much larger than those associated with the exhaustive application of test patterns. This encouraged the development of a method, based on the pseudorandom test model, which was earlier dismissed because it was thought that the analysis would be too cumbersome, or intractable, to incorporate the aspect of a detectability profile to achieve superior test length estimations.

Wagner *et al.* [Wagner87] have made analytical comparisons between the purely random and pseudorandom test models. Their findings are supportive to the conclusions drawn by Chin and McCluskey [Chin87]. Some of the test quality measures considered for this comparison are as follows: i) expected fault coverage; ii) test confidence; iii) expected test length; and iv), average test length.[4.2] All three of these test qualities were derived using a single stuck-at-fault model assumption, and required the knowledge of a *detectability profile*.

[4.2] A subscript R indicates an expression derived from a random testing model; similarly, a subscript P indicates an expression derived from that of a pseudorandom testing model.

4.1.1. Expected Fault Coverage

The *expected fault coverage,* $E[C_L]$, can be defined as the number of faults which can be detected in a test of length L divided by the total number of possible faults, M. Upon knowing the detectability profile, the expected fault coverage may be estimated by probabilistic analysis. The *exact fault coverage,* however, may be acquired only by direct fault simulation.

4.1.2. Test Confidence

A test quality measure, which is most often used to calculate test lengths when the detectability profile of a circuit-under-test is unknown, is that of *test confidence,* TC (see [Chin87]). It is, basically, the probability that a particular fault, with detectability k, is detectable within a test set of length L. Chin and McCluskey subsequently obtained the *random test length,* L_R, as a function of test confidence, as

$$L_R(TC_R) = \log_e(1 - TC_R) / \log_e(1 - k/N). \tag{4.1}$$

(N is the total number of possible input patterns; that is, $N = 2^n$, where n is the number of primary inputs.) If it is assumed that $N - L \gg k$, then a reliable approximation of the *pseudorandom test length,* L_P, as a function of test confidence can be made by [Chin87]

$$L_P(TC_P) \simeq N \cdot [1 - (1 - TC_P)^{1/k}]^2. \tag{4.2}$$

The idea of determining a *weighted test confidence measure* has been proposed by Shedletsky [Shedletsky77]. It depicts a more realistic test confidence measure, since it

takes into account all individual test confidence's associated with the detection of each fault in a circuit's fault set. In the case where each fault has an equal probability of occurrence, the weighted test confidence is equivalent to the expected fault coverage. This holds true for both purely random and pseudorandom test models. Usually, the weighted test confidence is used to calculate a *fault confidence loss,* which happens to be representatively equivalent to a *fault coverage loss measure.* The only difference between the two, is that the fault confidence loss is much more computationally expensive to calculate. Thus, the practicality of a weighted test confidence measure is somewhat restrictive.

4.1.3. Expected Test Length

The *expected test length,* $E[L_i]$, is a test quality measure which determines the length of a test set, either purely random or pseudorandom, required to detect a particular fault, F_i, of detectability k. For a purely random test model, the expected test length is

$$E_R[L_i] = N/k \ , \tag{4.3}$$

whereas for a pseudorandom test model, the expected test length is written as

$$E_P[L_i] = \frac{N+1}{k+1} \ . \tag{4.4}$$

4.1.4. Average Test Length

If all the faults of a circuit-under-test are equiprobable, and each fault, F_i, has a detectability of k_i, then an *average test length* test quality measure can be calculated for both purely random and pseudorandom test sets. (It should be noted that the average test length measure is simply the average of all individual expected test lengths.)

From equation 4.3, the average test length for a random test set is

$$E_R[L] = \frac{N}{M} \sum_{k=1}^{N} \frac{h_k}{k}, \tag{4.5}$$

where h_k represents the number of faults in the circuit-under-test with detectability k. The average test length for a pseudorandom test set can be deduced, in a similar fashion, by using equation 4.4 as depicted by

$$E_P[L] = \frac{N+1}{M} \sum_{k=1}^{N} \frac{h_k}{k}. \tag{4.6}$$

A more representative prediction of test length can be made by using a *weighted test length.* This approach does not assume that all faults are equiprobable, but, rather, includes in its calculation the estimated probability of each individual fault occurrence. The test engineer must determine the cost effectiveness of this test quality measure as to how it pertains to the circuit-under-test.

4.1.5. Comparison of Purely Random and Pseudorandom Test Quality Measures

For combinational networks, it is not necessary to apply more than a maximal-length test set (except for perhaps the additional application of the all zero test pattern), when a single stuck-at-fault model is used, to provide full testability. A purely random test pattern generator (RTPG) will more than likely have to generate much more than the nominal $2^n - 1$ test patterns before an exhaustive test set is finally applied to the circuit-under-test. A pseudorandom test pattern generator (PTPG), however, is bound to exactly $2^n - 1$ patterns to exhaust all possible input combinations. (as mentioned above, the null test vector is added when necessary.)

In an attempt to show that the pseudorandom test model is superior to the purely random test model for any given test confidence level, a formal development is required (see [Wagner87]). For low detectability, say $k = 1$, the ratio of the pseudorandom to purely random test length, for the same test confidence, TC, is

$$\frac{L_P}{L_R} \approx TC \,/\, \log_e(1 - TC)\,. \tag{4.7}$$

For a large detectability, say k approaching N, equation 4.7 becomes

$$\frac{L_P}{L_R} \approx 1\,. \tag{4.8}$$

Consequently, $L_P / L_R \leq 1$ for the same test confidence, irrespective of the fault detectability. For easily detectable faults, both pseudorandom and purely random test lengths are relatively the same. But, when there exist hard to detect faults, the pseudorandom model offers a more appropriate prediction of test length, because there is

no redundancy in the application test vectors. Thus, the test quality measures provided by the pseudorandom test model are, in general, more accurate than those determined from a purely random test model.

From the analysis in [Chin87] and [Wagner87], it can be stated that the purely random test model is a poor predictor of pseudorandom test behaviour. Because the analysis of the purely random model is no easier to calculate, and is always less accurate than the pseudorandom model, it should be discarded. This, of course, is true only for those cases where unbiased pseudorandom testing is applicable. When "biased," or "weighted," random testing is required, however, a random model is absolutely necessary, since all practical methods which are in use today are all random as opposed to pseudorandom in nature. (Pseudorandom weighted test patterns can be generated with conventional and/or hybrid cellular automata, but only at the expense of a reduced useful test set (see Chapter 6, Sec. 6.3.1).)

Chapter 5
Cellular Automata-An Overview

5.1. Introduction to Cellular Automata

Cellular automata were originally introduced in the late nineteen-forties by John von Neumann and Stan Ulam as possible idealizations of biological (living) systems. At the time, von Neumann's main interest in cellular automata (under the name of cellular spaces) was whether or not machines could reproduce. More appropriately, he wished to devise a simple system capable of self-reproduction. If he could, it would reveal profound similarities between mechanical objects and living organisms.

Von Neumann first attempted to design a machine, or automaton, that could make a physical copy of itself within itself. When complications arose he turned to Ulam for advice. Ulam suggested keeping track of the recursively defined objects von Neumann encountered by using a computer pattern that would automatically generate a second, identical to itself. Needless to say, von Neumann adopted the suggestion and while he never actually ran his program on a computer, his argument was accepted as the first demonstration that a collection of simple elements could automatically reproduce. Although referred to by a variety of names, including "tessellation automata," "homogeneous structures," "cellular structures," "tessellation structures," and "iterative arrays," it was decided that "cellular automata" would be the name best used to describe the era of self-reproducing automata [Wolfram83]. Von Neumann's

work on self-reproducing automata was completed by Arthur Burks [Burks70] and provides information on cellular automata in the formative years of computer science.

In nature, it is common to find systems which behave in extremely complex manners, but contain component parts which are quite simple. *Cellular automata* are mathematical idealizations of such systems. The realization of a cellular automaton consists of a structured lattice (array) composed of identical cells (sites). The values of the discrete variables occupying the sites describe the present state of the completely specified cellular automaton. Evolution occurs in discrete time steps according to some predetermined rule so that each value of the variable at one site is influenced by the values of the variables in its local neighbourhood. In most cases the neighbourhood of a site is taken to be the site itself, the site making the calculation, and its immediately adjacent sites. The successor states of the cellular automaton are updated synchronously, based on the values of its predecessor states, the neighbourhood, and on the defined set of local rules.

In 1970, John Conway introduced what was to become the best-known cellular automaton. The cellular automaton, named the ''game of life,'' had a biological aspect whereby cells are born, live or die depending on the neighbouring population density [Hayes84]. The game of life takes place on a two-dimensional rectilinear lattice where the cells are represented as a 1 or a 0 for living or dead respectively. Under some initial configuration, each cell is contained within a specified neighbourhood consisting of its eight nearest neighbours (i.e., Moore neighbourhood). At every discrete step of the cells evolution, each cell responds to the state of its immediate environment. In other words, each cell checks the state of the eight surrounding cells as well as its own state for each discrete time step. The algorithm for the game of life is as follows: If the center cell is living, it will continue to live in the next generation if either two or three cells in the Moore neighbourhood are living; if there are exactly three living cells in

the Moore neighbourhood, the center cell will live in the next generation irrespective of its present state; any other circumstance not already stated will require that the center cell of the Moore neighbourhood either die or remain dead [Hayes84].

Interesting phenomena occur when one views the game of life evolving with the aid of a display monitor. Some patterns oscillate while others pulsate like Amoebas; many more lapse into a stable or cyclical configuration whereas some die out entirely. The resemblance to living matter was so pronounced that Conway called his game LIFE [Hayes84]. In general, the cells can thrive only if they are neither starved (surrounded by fewer than two neighbours) nor overcrowded (surrounded by more than three neighbours).

5.2. Cellular Automata Classification

There are four properties which characterize a cellular automaton as described by Hayes [Hayes84]. The first property is the geometry of the array of cells. Identical cells interconnected in a line form what is referred to as a one-dimensional or line automaton. A two-dimensional automaton is constructed in a rectilinear lattice formation, while a three-dimensional automaton is readily realizable but not readily visualized.

The second property specifies the neighbourhood of each cell, $S(m)$, calculating its own next state. In two-dimensional automata there are two neighbourhoods which have been given much attention. The von Neumann neighbourhood of a particular cell consists of its four adjacent neighbours, those to the north, south, east, and west (see Figure 5.1a)). If the neighbourhood includes the four diagonally adjacent ones, a Moore neighbourhood is formed (see Figure 5.1b)). A one-dimensional cellular automaton has an even simpler neighbourhood: the neighbourhood is identified as the cells

to the left and right of a particular cell (see Figure 5.1c)). There may be many cells included
in the neighbourhood, all of which are contained within the linear structure.

A third factor considered in describing a cellular automaton is the number of states per cell. The number of states is directly related to the complexity of the cellular automaton design. For example, binary automata, those with only two states per cell (0 or 1), may be used to represent true or false, or living or dead. More complicated automata, such as von Neumann's self-replicating automata, may contain as many as twenty-nine possible states.

Finally, the fourth characteristic is the rule which determines the future state of a cell or the entire automata based on the present configuration of its neighbourhood. There exist a vast number of rules which may be applied to cellular automata, each creating its own unique temporal state. If k represents the number of states per cell and n is the number of cells contained in each local neighbourhood, then there are k^{k^n} possible logical rules. For example, with even modest values of k and n selected, there are more one-dimensional cellular automata than there are atoms in the known Universe [Dewdney84].

In one-dimensional cellular automata: $a_i^{(t)}$ denotes the value of site i at discrete time step t; the site variable occupies an integer between 0 and $k - 1$; the parameter r determines the range of the neighbourhood; and, the rule ϕ describes the iterative evolution according to [Wolfram84a]

$$a_i^{(t+1)} = \phi\left[a_{i-r}^{(t)},\ a_{i-r+1}^{(t)}, \cdots\cdot,\ a_{i+r}^{(t)}\right]. \tag{5.1}$$

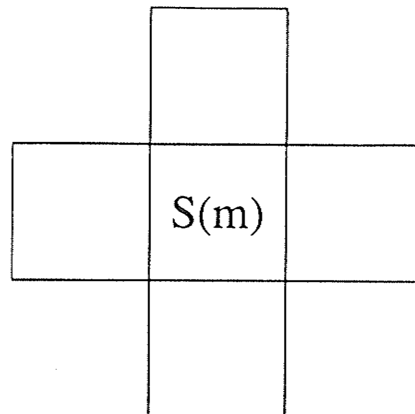

Figure 5.1a): *Von Neumann Neighbourhood (Two-dimensional)*

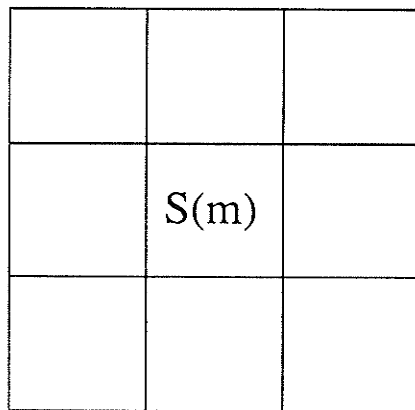

Figure 5.1b): *Moore Neighbourhood (Two-dimensional)*

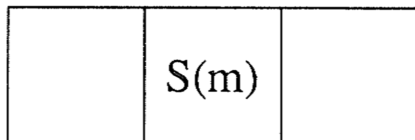

Figure 5.1c): *Local Neighbourhood (One-dimensional)*

For nearest neighbour interaction only, that is, $r = 1$, we may rewrite this equation as

$$a_i^{(t+1)} = \phi\left[a_{i-1}^{(t)},\ a_i^{(t)},\ a_{i+1}^{(t)}\right]. \tag{5.2}$$

A two-dimensional analogy of nearest neighbour interaction can be described by

$$a_{i,j}^{(t+1)} = \phi\left[a_{i,j}^{(t)},\ a_{i,j+1}^{(t)},\ a_{i+1,j}^{(t)},\ a_{i,j-1}^{(t)},\ a_{i-1,j}^{(t)}\right]. \tag{5.3}$$

This represents a five-site von Neumann neighbourhood generalization in an [i,j]-Cartesian coordinate system.

Packard [Packard85] reports evidence that global properties of two-dimensional cellular automata are similar to those of one-dimensional cellular automata. Local phenomena found in two-dimensions also have analogues in one-dimension. Because of the simple geometric construction, there is a greater likelyhood of gaining an analytic understanding of the one-dimensional automaton's evolution. Consequently, the one-dimensional cellular automaton is discussed to reveal generalistic properties of cellular automata. Any further references to two-dimensional cellular automata will be stated explicitly.

5.3. One-dimensional Cellular Automata

The evolution of a one-dimensional cellular automaton may be viewed by observing the successive states of the finite automaton on a video display monitor. A two-dimensional pattern is formed which has one spatial and one temporal axis. Boundary

conditions may be satisfied by using one of the following two methods: i) the first and last sites of the array are joined so that they become neighbours (topologically, a circle is formed creating "cyclic," or "periodic," boundary conditions) (see Figure 5.2a)); and ii), the left and right neighbours, of the first and last sites of the array respectively, contain fixed values (equal to zero for "null" boundary conditions, see Figure 5.2b)). In one-dimensional cellular automata, there are at most $2r+1$ sites in a given neighbourhood. This implies that propagating features generated in time evolution will travel at most r sites per time step.

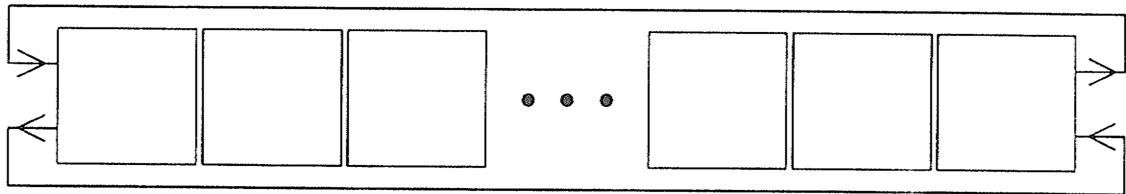

Figure 5.2a): *One-dimensional CA with Cyclic boundary conditions.*

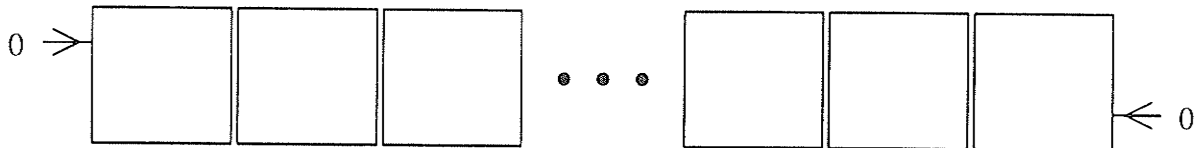

Figure 5.2b): *One-dimensional CA with Null boundary conditions.*

Binary automata are those which have two possible values for the variables at each site, or $k = 2$; but, in addition to this, if $r=1$ (i.e., nearest neighbour selection) "elementary" cellular automata are produced [Wolfram84a]. There are a total of $k^{k^{2r+1}}$ or $2^8=256$ possible distinct elementary cellular automata. (Local rules for elementary cellular automata are described by an eight-bit number.)

A cellular automaton rule is considered "illegal" unless two restrictions are satisfied. First, a *null* or *quiescent* initial configuration, consisting entirely of zeros,

remains invariant under time evolution. This implies that (using equation 5.1)

$$\phi\left[0,\ 0,\ 0,\cdots,\ 0\right] = 0. \tag{5.4}$$

Secondly, the rules must exhibit *reflection symmetry* where symmetric states only are thereby generated. Thus, $a_{n+i}^{(t)} = a_{n-i}^{(t)}$ for some n and all i or

$$\phi\left[a_{i-r}^{(t)},\cdots,\ a_{i+r}^{(t)}\right] = \phi\left[a_{i+r}^{(t)},\cdots,\ a_{i-r}^{(t)}\right]. \tag{5.5}$$

These two restrictions produce what are known as "legal" cellular automata. Out of a total $k^{k^{2r+1}}$ possible one-dimensional cellular automata, $k^{k^{r+1}\cdot\frac{(k+1)}{2}-1}$ are legal [Wolfram84a]. For instance, there are 32 *legal elementary* one-dimensional cellular automata of the form $\beta_1\beta_2\beta_3\beta_4\beta_2\beta_5\beta_4 0$. The 32 legal rules are 0, 4, 10, 22, 32, 36, 50, 54, 72, 76, 90, 94, 104, 108, 122, 126, 128, 132, 146, 150, 160, 164, 178, 182, 200, 204, 218, 222, 232, 236, 250, and 254 (see [Wolfram83]). Consider rule 90; if a truth table is constructed revealing the successor state, $a_i^{(t+1)}$, of a three-site neighbourhood under rule 90, we find a corresponding eight-bit number (01011010) (see Table 5.1). (This number can in turn be represented by its decimal equivalent 90.) One can readily observe that rule 90 satisfies both the quiescent configuration and reflection symmetry restrictions of legality. More appropriately, rule 90 is known as a "modulo-two" rule where the value of a particular site is simply the sum, modulo-two, of the values of its previous neighbouring sites.

The Boolean equivalent of this rule is therefore

$a_{i-1}^{(t)}$	$a_i^{(t)}$	$a_{i+1}^{(t)}$	$a_i^{(t+1)}$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Table 5.1: *Rule 90 Cellular Automaton Truth Table.*

$$S_{n+1}(m) = S_n(m-1) \oplus S_n(m+1), \tag{5.6}$$

where $S_n(m)$ is the value of site m at time step n and $\oplus$ denotes modulo-two addition (or "exclusive disjunction").

5.4. Class and Attractor Behaviour

Patterns which have been generated by one-dimensional cellular automata evolve from initial configurations, or "seeds," consisting of a number of non-zero sites. Wolfram [Wolfram84b], has suggested that some local rules produce simple behaviour while others produce complex behaviour. Empirical results show that these patterns

qualitatively exhibit one of the following features: i) disappearance with time; ii) evolution to a fixed finite size; iii) indefinite growth at a fixed speed; and iv), growth with irregular contractions. It has also been shown that different initial configurations change the patterns generated by a particular cellular automaton; but, the cellular automaton's statistical properties remain relatively unchanged. Therefore, all of the one-dimensional transition rules can be placed into one of four qualitative classes of behaviour: i) spatially homogeneous states; ii) sequences of simple stable or periodic structures; iii) chaotic aperiodic behaviour; and iv), complicated localized structures (some propagating) [Wolfram84b]. The cellular automata found within each class, irrespective of their seeds and rules, reveal common qualitative properties.

It has been conjectured that one-dimensional cellular automata may be the simplest well defined systems capable of complicated self-organizing behaviour. Many continuous dynamical systems evolve from some initial random seed to highly specified structures. This is the direct result of the effect of *attractors* which conform a system towards a subset of all the possible configurations. There is a strong relationship between the classes of cellular automata and the type of attractors found in natural physical systems. For class 1 cellular automata, evolution leads to a stable uniform state, such as all zeros or ones in elementary automata. Hence, the temporal evolution to some final configuration is analogous to a limit point in a continuous dynamical system. Class 2 cellular automata generate patterns which repeat indefinitely (limit sets), analogous to what are known as limit cycles. The chaotic aperiodic limit sets generated by class 3 cellular automata are associated with more interesting entities known as strange or chaotic attractors. This is characteristic of phenomena such as turbulent flow. The evolutions of these systems proceed toward a subset of all the possible configurations. Class 4 cellular automata represent rules which are both rare and intriguing. In predicting the successor state of a class 4 cellular automaton, there is probably no more efficient procedure than allowing the automaton itself to compute the

state. A related conjecture suggests that class 4 automata may be capable of universal computation, so that their evolution may implement any finite algorithm [Hayes84].

Slight perturbations in the initial configurations of the various cellular automata classes are observed to have characteristic effects on stability and predictability. The results of changing the value of a single initial site has the following consequences: class 1 cellular automata experience no change in the final state; class 2 cellular automata differ only in a finite region; class 3 cellular automata change over an ever increasing region; and, class 4 cellular automata experience irregular changes [Wolfram84c].

5.5. Summary

Cellular automata are simply constructed systems which are capable of complex behaviour. They are constructed from a structured lattice composed of identical cells representing site variables. Each site variable is occupied by an integer value describing the present state of the specified cellular automaton. The cellular automaton evolution proceeds according to some predetermined rule, where the successor states are updated synchronously. Four general properties which characterize a cellular automaton are: i) the geometry; ii) the neighbourhood; iii) the number of states per cell; and iv), the rule specifying the evolution.

The structure of a one-dimensional cellular automaton consists of an array of cells interconnected in a line configuration. A one-dimensional cellular automaton is further characterized by the kind of behaviour which results from its evolution. This behaviour can be explained by the presence or absence of attractors in its state transition diagram. There are, in general, four distinct classes which all one-dimensional cellular automata may be categorized in.

Chapter 6
Weighted Test Pattern Generation for Built-In Self-Test using Cellular Automata

The concept of weighted test pattern generation was first initiated to aid in the detectability of hard-to-detect or pseudorandom-resistant faults (see [Schnurmann75, Savir84, Chin84]). By weighting the input probability distribution, an attempt is made to expose the hard-to-detect faults, thereby making them random pattern testable. In other words, an optimal input probability distribution is desired in order to maximize fault coverage and minimize test length. (This is basically an extension of earlier work concerning equiprobable, unbiased, pseudorandom test patterns, with the single stuck-at-fault model assumed in the derivation of test quality measures.) Overall, there exists a class of combinational networks whose testability may be significantly improved by utilizing a weighted probability distribution. Although it may be easy, in practise, to weight the input test patterns for such circuits, important statistical attributes must be maintained to ensure testability.

There are two different techniques introduced for the purpose of providing an alternative to previous Weighted Test Pattern Generator (WTPG) designs. Both of these approaches employ one-dimensional Cellular Automata (CA) structures in an attempt to achieve better statistical properties (see [Podaima88]). Ideally, a system possessing statistical independence is sought. However, because the time evolution of a one-dimensional cellular automaton is deterministically generated, only a certain

degree of "apparent" statistical independence is, at best, attainable. For Built-In Self-Test (BIST), this level of behaviour is sufficient, since it is expected that empirically generated statistical quantities will conform, within reasonable accuracy, to those determined analytically. In this regard, those one-dimensional cellular automata which are capable of generating "chaotic" behaviour are of particular interest. In certain cases, they have also been shown to exhibit "autoplectic" behaviour. Autoplectic behaviour implies that from some simple initial state, the cellular automaton yields complicated, or perhaps unpredictable, configurations in its time evolution. Systems which are autoplectic in nature generate sequences considered effectively random [Wolfram85].

The following analysis examines and assesses the proposed class of WTPGs with respect to those devised previously, based on the maximal-length linear feedback shift register (LFSR). Several time evolution characteristics are investigated. They include: state-time visualization; density and average density; probability mass function (histogram); magnitude spectrum; both auto and cross-correlation coefficient functions; and, bit sequence tuple lengths. These observations form the basis whereof comparisons can be made directly, leading to advantageous amendments in BIST circuitry.

6.1. Weighted Test Pattern Generation for Fault Detectability

The versatility at which pseudorandom patterns may be distributed in a system, and the strikingly high fault detection capability of these test patterns, makes pseudorandom pattern testing particularly suited for use in a BIST strategy. Today, there is little argument that this ideology is applicable to a large class of combinational circuits. Often, however, there are instances where a circuit's testability is hindered by a relatively few number of hard to detect faults. When ordinary unbiased pseudorandom

testing is performed, these faults are categorically known as pseudorandom-resistant faults, and contain remarkably low fault detection probabilities. Circuits known to contain such faults are called pseudorandom-resistant, and must be dealt with by an advanced form of testability.

A successful candidate, which maintains many of the attributes first introduced by Unbiased Pseudorandom Test Pattern Generation (UPTPG), is Weighted Test Pattern Generation (WTPG). This approach involves the application of weighted probability distributions in order to enhance the probability of generating suitable patterns to expose the pseudorandom-resistant faults (see [Chin84]). In doing so, the overall test set will decrease while the fault coverage increases, thereby, making this particular sub-class of circuits weighted random pattern testable. Work by Wunderlich [Wunderlich88], and Waicukauski and Lindbloom [Waicukauski88], have successfully demonstrated that this is, in fact, the case for the 1O ISCAS [Brglez85] benchmark designs.

6.2. Weighted Test Pattern Generation Methodology

By using an adaptive weighted test pattern generation method, or probabilistic fault grading technique, it is possible to find the relationship between the fault detection and rate of excitation of a circuit. With this knowledge, an appropriate weighted probability distribution(s) may then be assigned. Although more recent advances in WTPG have resulted in some efficient heuristics for computing multiple weighted test sets, earlier work concerning Chin and McCluskey [Chin84] suffices to demonstrate, in a simplified manner, the attributes of WTPG.

According to Chin and McCluskey [Chin84], Savir, in [Savir83], explains how a fault detection probability profile for a particular design may be calculated, which would give the probability of detecting any fault given an arbitrary input probability

distribution. Then, in [Savir84], he shows how this fault detection probability profile can be used to estimate the random test pattern length required to perform a test given a certain fault escape probability.

Chin and McCluskey use the fault detection probability profiles, corresponding to each chosen weighting, to isolate the pseudorandom-resistant faults, and to identify the weightings for which the pseudorandom-resistant faults are more readily detectable. Once this is established, a weighted random test set, with either single or multiple distributions of specified length, may be applied to the circuit-under-test, in order to maximize the fault detection probability of all faults concerned. The resulting output data, from the application of the entire test set, may be compacted by conventional CRC signature analysis and subsequently compared for circuit validity.

To illustrate the amendments of WTPG, first consider a fan-out free combinational logic AND-tree consisting of thirteen three-input AND gates to be random tested (see Figure 6.1). Notice that this structure consists of 27 primary inputs for the first 9 gates forming the first of three levels. Additionally, located at the end of the third and final level, is a single primary output. Since there is no fan-out, only those faults at the primary locations need be considered for analysis with a single stuck-at-fault model. By examining three chosen probability distributions (i.e., 25%, 50%, and 75%), a complete fault detection profile corresponding to all possible faults may be constructed, as given in Table 6.1. The detection probabilities for an input stuck-at-1 and stuck-at-0, may be computed by $P_d = P_w^{26}(1-P_w)$ and $P_d = P_w^{27}$, respectively; likewise, the detection probabilities for the output stuck-at-1 and stuck-at-0, may be computed by $P_d = 1 - P_w^{27}$ and $P_d = P_w^{27}$, respectively, where P_w denotes the selected weighted probability distribution, and P_d the corresponding detection probability.

In order to simplify the procedure of computing the required test length for each weighted distribution, Chin and McCluskey capitalized on Savir's notion that "....faults

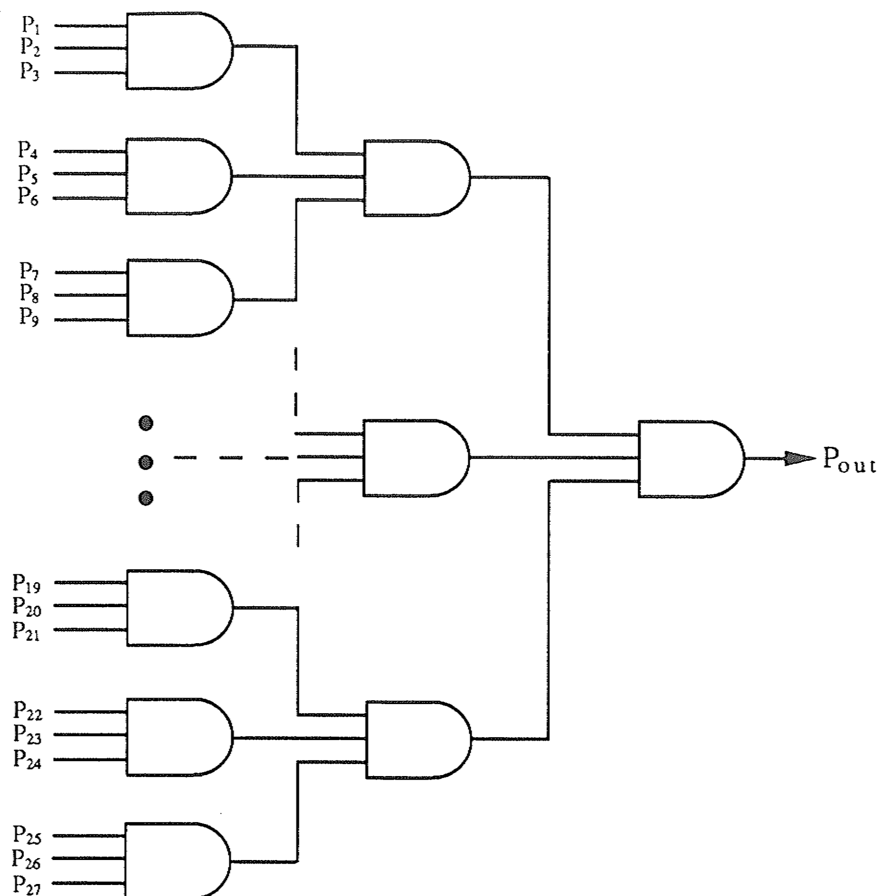

Figure 6.1: *Nested AND-gate Tree with Three-input Logic.*

with a detection probability greater than twice the detection probability of the worst fault (lowest detection probability) do not contribute significantly to the required test length [Savir84].'' Furthermore, his calculations yielded a required test length [Chin87]

$$L = \frac{\left[ln(n_w) - ln(e) \right]}{P_{dw}}, \tag{6.1}$$

Detection Probability Formula	Fault Class	Number	Detection Probability 25%	50%	75%
$P_d=(P_w)^{26}(1-P_w)$	input SA/1	27	$1.67\text{x}10^{-16}$	$7.45\text{x}10^{-9}$	$1.41\text{x}10^{-4}$
$P_d=(P_w)^{27}$	input SA/0	27	$5.55\text{x}10^{-17}$	$7.45\text{x}10^{-9}$	$4.23\text{x}10^{-4}$
$P_d=1-(P_w)^{27}$	output SA/1	1	1.00	1.00	0.9996
$P_d=(P_w)^{27}$	output SA/0	1	$5.55\text{x}10^{-17}$	$7.45\text{x}10^{-9}$	$4.23\text{x}10^{-4}$

Table 6.1: *Fault Detection Probability Profiles (25%, 50%, 75%). Taken from Ref. [Chin84]*

P_w	P_{dw}	n_w	L
25%	$5.55\text{x}10^{-17}$	28	$1.84\text{x}10^{17}$
50%	$7.45\text{x}10^{-9}$	55	$1.46\text{x}10^{9}$
75%	$1.41\text{x}10^{-4}$	27	72310

Table 6.2: *Worst Case Test Set Length for Test Confidence of 99.9% (25%, 50%, 75%). Taken from Ref. [Chin84]*

where P_{dw} is the fault detection probability of the "worst case" fault; n_w is the number of those worst-case faults; and, e is the escape probability, indicative of the desired test confidence level. So, when a test confidence of 99.9% (i.e., $e=0.001$) is selected, the AND-tree example circuit requires the test lengths given by Table 6.2. (Because Savir's model assumes truly random testing, the test length computed for the 50% distribution is larger than an actual exhaustive test set. However, strictly speaking, this model is evidently more than adequate for the purpose of exemplifying the potential benefits warranted by weighted random pattern testing.)

From this information, it is obvious that a single distribution of 75% would serve this structure a substantial improvement in testability over a conventional unbiased (50%) random/pseudorandom test. Considering that this example is trivial in nature, it successfully demonstrates the effectiveness of using weighted random pattern test sets on classified pseudorandom-resistant circuits. The same methodology can be applied to much more complicated combinational networks, so long as they fall into the required classification. For these networks, there may be no single weighted distribution necessary to facilitate testing, but, rather, a compromise between a multitude of different weightings, each with its own individual test length.

6.3. Cellular Automata based WTPG Considerations:

There are two possible techniques for realizing cellular automata based WTPGs: i) using conventional and hybrid one-dimensional cellular automata; and ii), using driving engines with external weighting logic.

6.3.1. Using Conventional and Hybrid One-dimensional Cellular Automata

As suggested by Wolfram [Wolfram83], the simplest statistical quantity characterizing a conventional one-dimensional cellular automaton's configuration (state) is the average fraction (density) of sites with an occupance of one. This simplistic quantification can also be used as an informal way of characterizing a cellular automaton's rule number. In total, there are

$$k^{k^{2r+1}} \tag{6.2}$$

and

$$k^{k^{2r+1}} \cdot \left[k^{(W-1)k^{2r+1}} - 1 \right] \tag{6.3}$$

conventional CA and hybrid CA (HCA) rules, respectively: k represents the number of states of each accompanying cell, r the size of the neighbourhood, and W the total number of cells making up the structure.

Based on nearest neighbouring cells (i.e., $r = 1$), there are 256 different conventional elementary one-dimensional CA rules, each represented by a unique eight-bit binary number. It has been substantiated that the fraction of ones contained in a conventional one-dimensional cellular automaton's binary rule number is, somewhat, indicative its steady-state average density. Suppose, for instance, that the fraction of ones found in some particular conventional CA rule number happens to be equal to one-half (e.g., Rule 30, Rule 45 and Rule 75). If the density evolution is observed for that rule, it is likely, unless the evolution is contained in an arbitrarily short cycle length, that it will lead to a steady-state value of approximately one-half. This implies a supposition that a range of weighting functions may be selected as a consequence of rule number, for as long as cycle lengths do not interfere with the pseudorandom properties of the autoplectic, or chaotic rules. Although these simply constructed rules give justification to the concept of using conventional one-dimensional chaotic CA as WTPGs, there are relatively few which may qualify for this purpose. Consequently, conventional CA with larger neighbourhoods, say $r = 2$, will have to be investigated before any inferences can be made. Because there are 2^{32} of these rules, it is conjectured that several exist; but, due to the large number, they may be difficult to find.

The problem is even more pronounced when HCA are investigated. This is because an HCA rule number can be any combination of those conventional rules with the same size neighborhood. Furthermore, it is not clear what the implications are, in terms of predicting the steady-state average density, of having a number of different

conventional rule numbers contributing to an overall HCA's rule number. If, however, a useful assortment of conventional CA and/or HCA weighted rules can be found, they would prove invaluable as WTPGs by virtue of the fact that exactly W useful outputs can be attained from a structure W cells wide. (This is opposed to having $W/2$ useful outputs for each stage of a WTPG built on the principal of using an original driving engine W cells wide.) But, due to the difficult nature of this promising concept, the task of finding them is, at present, left for future considerations.

6.3.2. Using Driving Engines with External Weighting Logic

Past methods of WTPG design incorporate the maximal-length LFSR as a primary "driving engine," supplying pseudorandom patterns to an array of logic gates performing a weighting operation on incoming patterns. Naturally, it is expected that if a more capable pseudorandom, or even random, number generator is used, the result will be an improvement in the WTPG function. For the purpose of achieving different output probabilities, standard Boolean logic gates, each with their own unique probability profile, maybe configured to deliver the desired weighting. In Appendix A, Figures A.2 and A.3 represent the probability profiles, for the occurrence of a "1," of the various logic gates. The probability profiles of Figure A.2, with, of course, the exception of the Inverter, simply constitutes the situation of equiprobable input probabilities. Conversely, Figure A.3 shows what the output probabilities are for a variety of ranging input probabilities, *Pin A* and *Pin B*.

By connecting the various logic gates, forming single or multi-level logic arrays, a number of incremental output probabilities are attainable. For example, if a 50% equiprobable driving engine is used, only probability increments of 1/4 are possible by incorporating a single-level array of logic. However, if a two-level array of logic is utilized instead, then probability increments of 1/16 are permitted. Even though finer

increments are still further possible, it should be kept in mind that the number of usable outputs substantially decrease, as the number of logic levels increase. For this reason, the resolution of the probability increment is limited by the restrictions imposed upon by VLSI; in particular, the area constraint.

In an attempt to find a better WTPG, based on the application of an external weighting logic, four different driving engines are investigated. They include: i) a non-linear random number generator[6.1]; ii) a pseudorandom maximal-length LFSR; iii) a reversible pseudorandom maximal-length Rule 90/150 HCA; and iv), an irreversible pseudorandom non-maximal-length Rule 30 CA. With the exception of the pictorial state-time evolutions, where the original driving engines are fifty-three cells wide (to allow for better visual presentation for comparison), all statistical analysis is performed on those machines based on engines thirty cells wide (as they are large enough to exhibit global characteristics, but small enough to allow extensive analysis). For the Rule 90/150 HCA of width thirty, the combination

$$Rule\ 90/150\ HCA : 000001100010000110000100111110$$

is used, where a "0" represents a Rule 90 cell and a "1," a Rule 150 cell [Pries88].

[6.1] The non-linear random number generator is based on a uniform distribution generated by a Non-Linear Additive Feedback Shift Register (NLFSR), using the *random* and *srandom* library functions contained in Sun Microsystems' Inc. "C" Library-release 3.5. It is used as an expected, or ideal, independent random number generator for which comparisons can be made.

6.3.2.1. CA based Pseudorandom Driving Engines

We first consider the Cellular Automaton and its suitability to the VLSI environment, together with the richness of its pseudorandom number generation capabilities. A one-dimensional cellular automaton is characterized by a longitudinal lattice of identical cells. Each individual cell is represented by an internal state. At each time step, the internal state is updated according to a specific local function or rule. The internal state of any given cell m at time n is donated by $S_n(m)$. (The n refers to a given discrete time step, or a given clock pulse from a global synchronous clock.) Next-state evaluation of a cell depends, in general, upon its own state and those of its two nearest neighbours on the preceding time step. The rule of operation ϕ computes the new state $S_{n+1}(m)$ based on the values $S_n(m-1)$ (left neighbour), $S_n(m+1)$ (right neighbour), and $S_n(m)$. The value of each of the cells is updated in synchronism with a global clock. At the ends of the array various boundary conditions may be applied, the most common being null and cyclic boundary conditions. From the design perspective, the most attractive of these is the null boundary condition.

In general, four classes of global behaviour have been empirically observed during the time evolution of CA systems operating under various local rules. Class 1 automata evolve to homogeneous final global states, class 2 to periodic structures, class 3 exhibit chaotic behaviour, and class 4 yield complicated localized and propagating structures. Class 3 CA have also been shown in certain cases to exhibit autoplectic behaviour. It is this behaviour which is of particular interest to pseudorandom number generation, since it exemplifies positive spatial and temporal measure entropies.

An example of a class 3 autoplectic elementary cellular automation with periodic boundary conditions is a Rule 30 CA, or

$$S_{n+1}(m) = S_n(m-1) \oplus [S_n(m) \cup S_n(m+1)]. \tag{6.4}$$

For a Rule 30 CA, all possible site configurations occur with equal probabilities, given an equal probability initial ensemble, so that the spatial measure entropy is maximal [Wolfram84d]. It should be further noted that a Rule 30 CA also exhibits computational irreversibility, which implies that no program can predict the sequences of the evolution more rapidly than the direct simulation itself. For this reason, the vertical sequences produced have demonstrated to represent very effective pseudorandom numbers.

Unlike the non-maximal-length Rule 30 CA, the Rule 90/150 hybrid cellular automaton (HCA) is maximal-length in nature, and has the added benefit of null boundary conditions (see [Pries86, Hortensius87]). From a performance point of view, the Rule 90/150 HCA is superior, since it offers class 3 autoplectic behaviour with a simplified wiring scheme. Circuit operation can, therefore, succeed that which is attainable from a Rule 30 CA by virtue of the fact that absolutely no cross-chip communication is required. The architecture of a Rule 90/150 HCA is comprised of combinations of Rule 90 and Rule 150 cells, which happen to be individually linear in nature. A Rule 90 CA is represented by

$$S_{n+1}(m) = S_n(m-1) \oplus S_n(m+1), \tag{6.5}$$

and a Rule 150 CA by

$$S_{n+1}(m) = S_n(m-1) \oplus S_n(m) \oplus S_n(m+1). \tag{6.6}$$

When the superposition principal for CA is applied, the resulting Rule 90/150 HCA will also be linear in nature, but in a convincingly more elaborate fashion.

The only difficulty encountered with the Rule 90/150 HCA structure is that maximal-length logical combinations must be analytically determined to match the size of the structure. However, once a table of combinations is established for all practical sizes, which corresponds to the selection of taps for maximal-length LFSRs, the cost of implementation will be reconciled.

When it comes to test verification, if CA based pseudorandom number generation is used, the expected fault coverage of a circuit is conjectured to conform to the actual fault coverage much more accurately than if LFSR based test pattern generation is assumed. This is especially true when stuck-open-faults, AC transition faults, and shorting faults are taken into account (see [Waicukauski88]). Furthermore, from a practical point of view, because of importance of exhaustive testing, in addition to pseudorandom, or weighted random pattern testing, and information compaction (signature analysis), the Rule 90/150 HCA comes across as being a very important contribution to DFT, particularly suited for use in BIST.

6.3.2.2. WTPG Logical Configurations

Since an improvement in overall wiring complexity would mean an increase in the speed with which a WTPG operates, several interconnection schemes are considered. A basic configuration is characterized by the amount of cell spacing between each consecutive pairing of gates, for a particular logic level. The basic configurations considered in this work include: i) a sharing configuration [S]; ii) a zero spacing configuration [J=0]; and iii), an N/2 configuration [J=(N/2)-1] (see Figure 6.2). Each level may contain a different basic configuration, but it is the overall logic array itself,

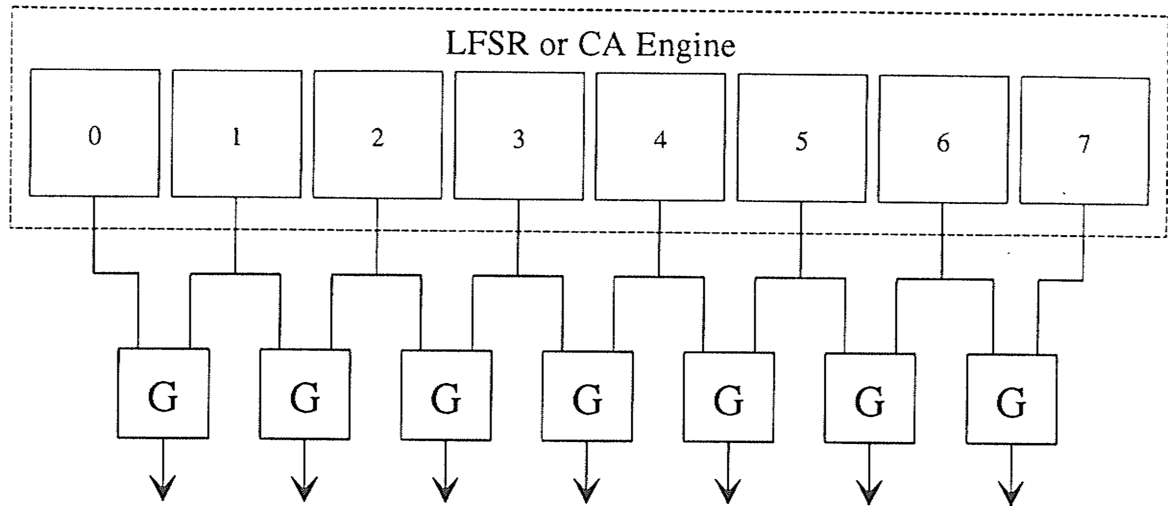

Figure 6.2a): *Parallel WTPG Sharing Configuration: [S].*

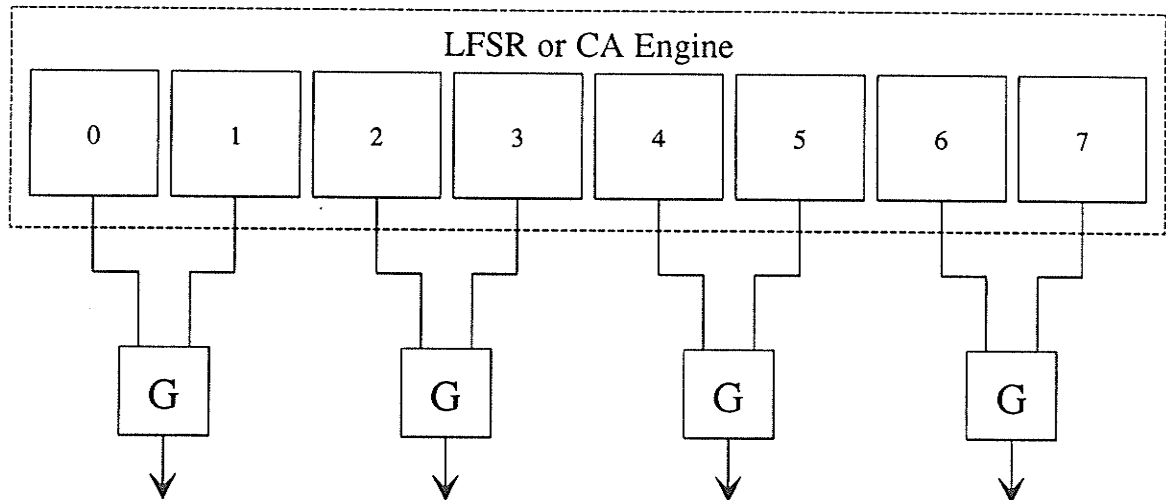

Figure 6.2b): *Parallel WTPG Zero Spacing Configuration: [J=0].*

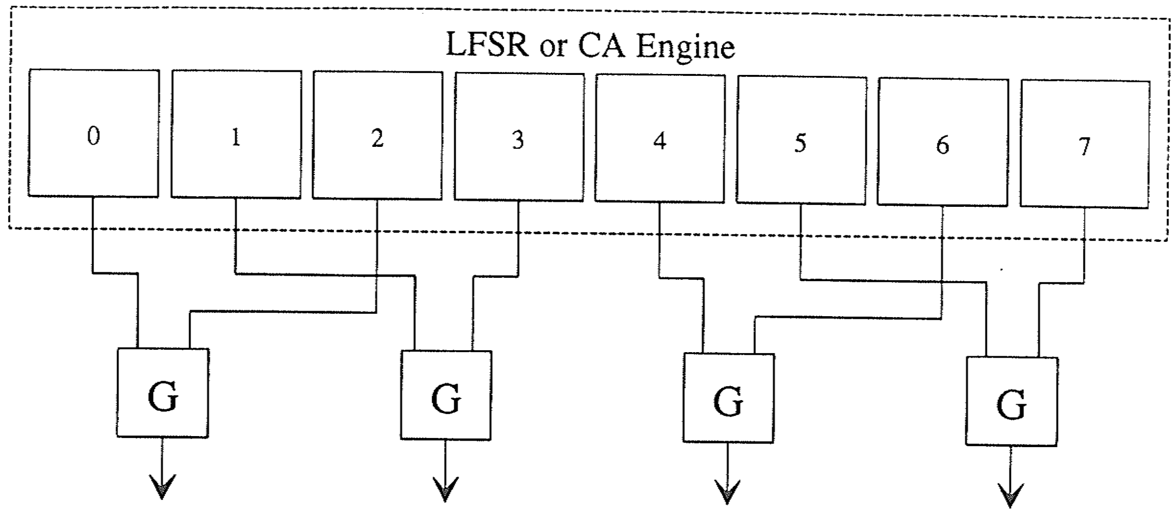

Figure 6.2c): *Parallel WTPG One Spacing Configuration: [J=1].*

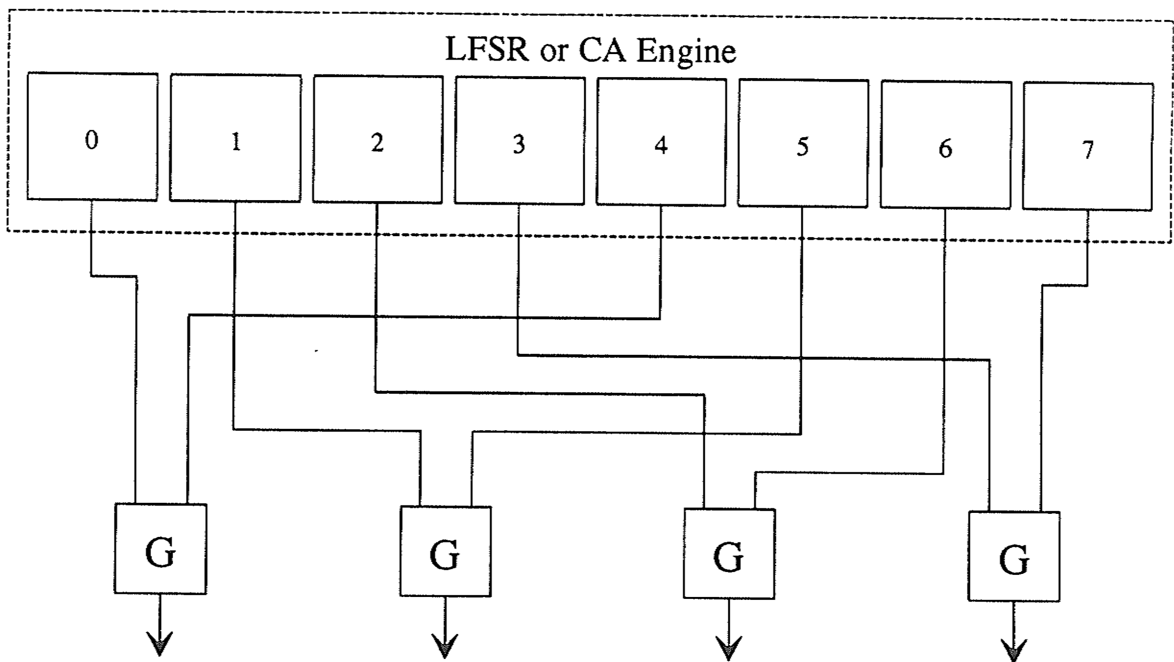

Figure 6.2d): *Parallel WTPG N/2 Configuration: [J=(N/2)-1].*

made up of one or more logic levels, which determines the final output probability assignments. The selection of an appropriate interconnection scheme (logic array) for a particular driving engine is dependent on the statistical qualities demanded, and the conformity of its implementation to VLSI. These factors will help establish a criterion which will make the proper selection amongst the different WTPG candidates possible.

6.4. Statistical Evaluation of WTPGs

An effective way of estimating the behaviour of some particular system is to evaluate a large sample of the time evolution of its simulating model. Thus, it is possible that a sufficiently large sample will contain enough information to offer a good approximation to some particular system's actual traits. By including some common statistical estimators for mean (unbiased), variance, and correlation, the following is intended to help establish some insight into the fundamental predicting mechanisms for some interesting finite state machines (see [Podaima89]).

6.4.1. Statistical Estimators

For any sequence of random variables, statistical estimations can be acquired which attempt to classify, with reasonable assurance, properties of the underlying random process. That is, it is desired that a random sample taken from an ensemble, or population, will contain the necessary information for which methods of statistical inference can be performed. With this basis, several quantities, or statistics, are obtainable; and, naturally, it is expected that some will be more meaningful than others. However, it is possible that even some of the more meaningful statistics will not always constitute unbiased estimates. Moreover, for those circumstances wherein a statistic loses validity, it is of great importance to identify any potential fallacy, so that

its actual benefit can manifest itself. (See Appendix B for the analytical comparisons of confidence intervals using some established unbiased and biased estimators. The derivations are from [Law82] and [Trivedi82].)

6.4.2. State-time Visualization

A simple apparatus which is often used, albeit with caution, because of its "unequivocal" ability, to immediately assess the randomness of an emanating process, is the human eye. As is well appreciated, sometimes the eye fails to interpret what actually resides. Keeping this in mind, a strictly qualitative visual test, with questionable validity, may be performed on a variety of finite state machine state-time evolutions. In Appendix C, Figures C.1 through C.4 represent the state-time evolutions of the original driving engines; Figures C.5 through C.8 represent the state-time evolutions of the sharing configuration at 25% and 75%; Figures C.9 through C.12 represent the state-time evolutions of the zero spacing configuration at 25% and 75%; and, finally, Figures C.13 and C.14 represent the state-time evolutions for the N/2 configuration at 25% and 75%.

By searching for some obvious patterns within an evolution, important pictorial comparisons can be made. Generally speaking, if large global, as opposed to local, patterns appear, this would most likely be damaging to a system's subjective randomness. The test itself can thus be used as a weak pre-test for random behaviour. From the evidence presented in Appendix C, there is noticeable global non-randomness for the LFSR based configurations; for those based on cellular automata, only local self-similar structures prevail. For a more quantitative look into random performance, much more sophisticated objective analysis need, and will, be considered.

6.4.3. Density and Average Density

The density, d_T, of a binary word (test pattern) is defined here as the actual number of ones per word length, W, or

$$d_T = \frac{\#_1(T)}{W} ; \tag{6.7}$$

whereas, the average density,

$$D_T = \frac{1}{T+1} \sum_{i=0}^{T} d_i , \tag{6.8}$$

is merely the average of the densities for some time evolution $T \in \{0, \ldots, L-1\}$, where L represents the total number of words in an evolution. The densities emanating over time evolution reveal information about the sophistication of the underlying process. For instance, if a system possesses total statistical independence, the L random variables making up the time evolution, are mutually independent and identically distributed (IID). Thus, graphically, the density evolution will appear similar to band limited white noise. Also, because the statistical characteristics of a discrete independent temporal process does not change with time, that is, it is said to be a discrete-time stationary process, the average density evolution converges to some mean, μ, rapidly, with infinitesimal steady-state error.

In Appendix D, Figures D.1 through D.4 represent the density and average density evolutions of the original driving engines. These finite state machines each consist of thirty sites, and, as indicated previously, are used as a primary source of pseudorandom test patterns, which feed weighting logic arrays. (An AND array performs a 25%

weighting, and an OR array performs a 75% weighting). Most undoubtedly, for the various gate configurations, the ability of the original generator to emanate high quality pseudorandom numbers is critical to the generation of high quality weighted random test patterns. Figures D.5 through D.8 represent the evolutions of the sharing configuration at 25%, and Figures D.9 through D.12 for those at 75%; Figures D.13 through D.16 represent the evolutions of the zero spacing configuration at 25%, and Figures D.17 through D.20 for those at 75%; and, finally, Figures D.21 and D.22 represent the evolutions of the N/2 configuration at 25%, and Figures D.23 and D.24 for those at 75%. In general, there are only subtle differences between the 25% and 25% operations, as pertaining to the properties of density and average density evolutions, found as a result of using the different respective logic gates.

By observing both density and average density evolutions of the various structures in their different configurations, it is apparent that the configurations of the cellular automata based structures are more attractive than those based on the linear feedback shift register: they resemble, more closely, the evolution based on a statistically independent model.

6.4.4. Probability Mass Function

Any *discrete-time stochastic process* can, in part, be described by its discrete probability density function, or Probability Mass Function (PMF). This emphasizes the importance of deriving a histogram which, at least approximately, represent the PMF of some random process. For any finite state machine, each site position can take on only one of two values, "0" or "1." If it is further stipulated that the process governing the evolution of a finite state machine be statistically independent, each bit position can be completely specified by a *Bernoulli* random variable. By Theorem E.1 (Appendix E), when the entire word (width) is examined, a function of W Bernoulli

random variables can be formulated contributing to the entire finite state machine's specification. For such a system, the densities, d_T, emanating over time evolution conform to a discrete *Binomial or Delta* PMF. In light of Theorem E.2 (Appendix E), it is then possible to compare a histogram generated empirically, by some deterministic finite state machine, to a PMF obtained assuming total statistical independence. Although this comparison does not provide information as to how independent, or dependent, a particular deterministic process is, it does reveal some important aspects of distribution convergence. Naturally, it is expected that a machine based on a statistically independent model will converge rather rapidly to a Binomial PMF.

In Appendix F, Figures F.1 through F.3 represent the Combination (Binomial Coefficient) Profiles, and Binomial PMFs, of thirty, twenty-nine, and fifteen contiguous Bernoulli random variables, respectively. From the combination profiles, one can determine, exactly, the expected number of possible combinations of a particular one's count, or density d_T. This information is, in turn, used in the computation of the corresponding Binomial PMFs, which provide the expected probability of generating a particular density for any given number of random variables. These Binomial distributions are used as a reference of comparison for histograms generated by deterministic methods, with test sets of length L equal to 500, 5,000, and 20,000.

Figures F.4 through F.7 represent the density histogram evolutions of the original driving engines; Figures F.8 through F.11 represent the density histogram evolutions of the sharing configuration at 75%; Figures F.12 through F.15 represent the density histogram evolutions of the zero spacing configuration at 75%; and, finally, Figures F.16 and F.17 represent the density histogram evolutions of the N/2 configuration at 75%. It is evident from these plots that both Rule 90/150 hybrid and Rule 30 cellular automata based structures, in their various configurations, converge more quickly, and are thus more favourable. However, as indicated by Figures F.8 through F.11, the density

histogram evolutions of the sharing configuration (including those based on a statistically independent driving engine) never do converge to the ideal Binomial PMF. Nonetheless, this level of behaviour may be appreciably acceptable for some of the less restrictive applications.

6.4.4.1. Goodness-of-fit Test

The probability mass function, $p(X)$, or probability distribution function, $F(X)$, completely characterizes the behaviour of the random variable X [Trivedi82]. Because of this property, it is of great importance to establish the "goodness-of-fit" between a distribution (sampling distribution), or histogram, determined empirically, with that which is proposed, or postulated. The best approach used to substantiate a measure of similarity is to test a hypothesis regarding some previously known characteristic. With the benefit of some statistical measure, a null hypothesis, Ho, may, at best, be rejected or reputed. It has been determined that a χ^2 (Chi-square) statistic is particularly suited for this type of analysis. Explicitly stated, the χ^2 test is made with the null hypothesis Ho: The data $X_1, X_2, \ldots, X_L$ are IID discrete random variables, with a Binomial PMF.

Definition 6.1:

According to Knuth [Knuth81], the χ^2 test is perhaps the best known of all standard statistical tests. Its potential is best emphasized by virtue of the fact that it may be used in conjunction with many other statistical tests.

The χ^2 statistic is used mainly for the purpose of providing the means by which arbitrary, empirically determined, data can be compared to some ideal or "expected" value. This is done by weighting the squares of the differences between the observed data and the expected data (determined by some discrete

or continuous probability function) in the form of a summation. The statistic is expressed in the form

$$\chi^2 = \sum_{i=0}^{k-1} \frac{\left[N_i - LP(i)\right]^2}{LP(i)}, \tag{6.9}$$

where L denotes the size of the independently selected data set; $LP(i)$ the expected number of outcomes; and N, the observed number for some particular category i, ranging from 0 to $k-1$. (For large L, equation 6.9 is approximately χ^2 distributed.)

By Definition 6.1, it is obvious that the quantity χ^2 should be reasonably small for Ho to be reputed; otherwise, if χ^2 is considered too large, Ho will be outright rejected. By observing tabulated quantities of a χ^2 distribution, a corresponding percentage point, or probability, is obtainable for some degree of freedom ($d.f.$). (As a ''rule-of-thumb'' $LP(i) \geq 5$, so the $d.f$ is available as a by-product of the computational process.) This percentile forms the basis of the determining factor to which the null, or alternative hypothesis, Ha, is rejected.

A more direct use of the χ^2 test, as applied to random number generator testing, involves probabilistically judging the actual data emanating from a random number generator. Although this procedure is no more different than that required in the testing of a null hypothesis for distribution fitting, its implications are more profound. In view of Knuth [Knuth81], no definitive statement can be made as to whether a sequence is random or not; however, what can be said is how probable, or improbable, certain sequences are of being randomly generated. With respect to this outlook, he

has suggested the following interpretation as a means of rating the randomness of large sequences of "apparently" random data: If χ^2 is less than 1% or greater than the 99% entry, the sequence is "rejected" as not being sufficiently random; if χ^2 resides between the 1% and 5% entries, or the 95% and 99% entries, then the sequence is "suspect;" if χ^2 resides between the 5% and 10% entries, or the 90% and 95% entries, the sequence is "almost suspect;" but, if the χ^2 lies somewhere between 10% and 90%, then the χ^2 is thought to be a value which could be produced by a random sequence.

Typically, a χ^2 test is performed at least three times on different sequences of adequate length. Also, it should be noted that for instances where χ^2 testing is applicable, it is only valid asymptotically for independently observed data. Therefore, tests which check for independence (such as, a serial correlation, tuple, temporal measure entropy, spectral, lattice, or run test), depending on the kind of data comprising the sequence, should be performed first, so that the authenticity of the χ^2 can be acknowledged.

Configuration	d.f.	NLFSR	LFSR	HCA R90/150	CA R30
Engine	18	Pass	Fail	Pass	Pass
[S]	15	Fail	Fail	Fail	Fail
[J=0]	10	Pass	Fail	Pass	Pass
[J=N/2-1]	10	Pass	Fail	Pass	Pass

Table 6.3: *Chi-square PMF Test for the various WTPG Configurations.*

In order to make quantitative comparisons of density histogram convergence among the various configurations, a χ^2 test can be adequately performed on the

histograms generated from a large test set. Because it is known that an IID statistical process dictates ideal behaviour, it should, therefore, represent the postulated, or reference, function in the form of a Binomial PMF. With respect to this and Definition 6.1, the χ^2 metric can then be computed for an analytical establishment. Using a test set of 20,000 vectors, an approximation level between 10% and 90%, and the criterion that $LP(i) \geq 5$, Table 6.3 clearly illustrates that the LFSR based configurations generate density sequences which fail the χ^2 test; whereas, most of the cellular automata based configurations generate density sequences which can be considered randomly generated. (From earlier discussions concerning the sharing configuration, it should not be surprising that both LFSR and cellular automata based sharing configurations generate discrete density sequences which fail the χ^2 test.) This, overwhelmingly, exemplifies the undesirability of LFSR based WTPGs, especially considering the smaller test lengths, which is more commonly associated with weighted random pattern testing.

6.4.5. Magnitude Spectrum

A measure used to indicate the rate at which densities change on a global level, within a sample function, can be made by applying a Fourier Transform. Since the density sample function d_T (i.e., the density evolution of some discrete-time stochastic process) represents what can be considered a large set of points, "sampled" with fixed increments of time Δ, it is typical of what the density evolution looks like for a number of other start up "seeds." Thus, if d_T contains L density values, its Discrete Fourier Transform can be determined by [Press86]

$$F(f_n) = \Delta \sum_{k=0}^{L-1} d_k e^{2\pi kni/L}, \tag{6.10}$$

where

$$d_k = d(t_k), \ t_k = k\Delta, \ k \in \{0, 1, \ldots, L-1\}, \tag{6.11}$$

and

$$f_n = \frac{n}{L\Delta}, \ n \in \{-\frac{L}{2}, \ldots, \frac{L}{2}\}. \tag{6.12}$$

By choosing $n \in \{0, 1, \ldots, L/2\}$, the resulting positive frequencies of a two-sided power spectral density are calculated to provide insight into the frequency content, or *magnitude spectrum* $|F_n|$ of a density time evolution. For a statistically independent process, the magnitude spectrum is expected to have an appreciably flat frequency response. In Appendix G, Figures G.1 through G.4 represent the magnitude spectrums of the original driving engines; Figures G.5 through G.8 represent the magnitude spectrums of the sharing configuration at 75%; Figures G.9 through G.12 represent the magnitude spectrums of the zero spacing configuration at 75%; and, finally, Figures G.13 and G.14 represent the magnitude spectrums of the N/2 configuration at 75%.

As expected, the magnitude spectrums for the configurations based on the statistically independent model do, in fact, look flat, or "white." On the other hand, when the spectrums of the LFSR based configurations are observed, the majority of the power is found to be contained in the low frequencies. This trend is similar to what is expected if the originating density evolution waveforms are fundamentally triangular in nature. Looking back to the originating density evolution of the LFSR, it is apparent that the contained waveform is somewhat triangular (see Figure D.20a), Appendix D). This behaviour is direct result of the dependencies amongst consecutive density values

d_T. Upon closer inspection, any succeeding density can only be the same or differ by, at the most, one density increment (i.e., $d_{T+1} = \{d_T + 1/W, d_T, d_T - 1/W\}$).

A serial test, which is basically a generalization of the χ^2 test to higher dimensions, is an empirical test which provides an indirect check on the assumption that individual d_T's are independent [Law82]. If the d_T's are correlated, the pairs (d_T, d_{T+1}) will tend to cluster around the diagonals of a unit square. The application of a χ^2 test will definitively detect this, and, thereby, indicate non-randomness in the generation process. Such is the case for the density evolution of any LFSR based configuration, where the predicting mechanism, as indicated, is rather simplistic. Evidently, it is the low frequency content of the magnitude spectrum which is most damaging to randomness.

Not surprisingly, both Rule 90/150 hybrid and Rule 30 cellular automata based configurations have far better suitable magnitude spectrums. The Rule 90/150 HCA configurations all appear flat; whereas, the Rule 30 CA configurations have slightly more high than low frequency content. By virtue of the fraction of ones contained in the eight-bit binary representation of the rule number 30, most test patterns generated by a Rule 30 CA driving engine are found to be composed of 50% ones. This property is responsible for the high frequency ''activity,'' which may very well include the effects of aliasing due to a limited fixed sampling frequency. In spite of the fact that this, and other non-maximal length chaotic class 3 CA (e.g., Rule 45, and Rule 75), have magnitude spectrum(s) which contain disproportionately more high frequency content, they are known to possess good random behaviour, at least up until they commence cyclic repetition. Consequently, the effects of disproportionately more high frequency content in the magnitude spectrum of any finite state machine is nowhere near as detrimental to randomness as those of low frequencies. Furthermore, it should also be noted that if any original driving engine is abundant in high frequency, then so too

will be the case under its various weighting configurations.

6.4.6. Auto and Cross-Correlation Coefficient Functions

The correlation coefficient function, $R(T)$, as shown in Lemma 6.1, is often used as a first-order estimate of dependence between bit sequences.

Lemma 6.1:

Let $<X_i>$ be a sequence of L integers, whereupon each individual integer is represented as a binary word consisting of W bits. Thus, for every $<X_i>$ there are W-bit sequences, $<x_i>$, of the same length.

The correlation coefficient function between any two binary streams, including itself, may be calculated by

$$R(T) = \frac{1}{L}\sum_{k=0}^{L-1} \frac{\left\{ \left(x_i[k] - \mu_i \right) \cdot \left(x_{i+j}[(k+T) \bmod L] - \mu_{i+j} \right) \right\}}{\sqrt{\sigma^2{}_i \cdot \sigma^2{}_{i+j}}}, \qquad (6.13)$$

where:

T = phase offset between sequences;

$x_i[t]$ = t-th bit in sequence $<x_i>$;

$x_{i+j}[t]$ = t-th bit in sequence $<x_{i+j}>$;

μ_i = mean of sequence $<x_i>$;

μ_{i+j} = mean of sequence $<x_{i+j}>$;

$\sigma^2{}_i$ = variance of sequence $<x_i>$;

$\sigma^2{}_{i+j}$ = variance of sequence $<x_{i+j}>$.

If any two bit sequences are statistically independent, then the cross-correlation coefficient function between them will always be equal to zero, $R(T)=0$. However, the converse is not true; that is, $R(T)=0$ does not immediately imply that any two bit sequences are statistically independent. Hence, a correlation coefficient function equal to zero is a necessary but not sufficient condition for statistical independence. If the evolution of a truly statistically independent finite state machine is examined, all parallel cross-correlation coefficient functions for the W bit positions will be equal to zero. But, if there are statistical dependencies inherent in the process, which is almost always the case, the cross-correlation coefficient functions will be other than zero (i.e., $1 \geq R(T) > 0$ and $0 > R(T) > -1$, where $T \in \{0, 1, \ldots, L-1\}$). In Appendix H, Figures H.1 through H.4 represent the space-phase correlation plots of the original driving engines; Figures H.5 through H.8 represent the space-phase correlation plots of the sharing configuration at 75%; Figures H.9 through H.12 represent the space-phase correlation plots of the zero spacing configuration at 75%; and, finally, Figures H.13 and H.14 represent the space-phase correlation plots of the N/2 configuration at 75%.

Each of the aforementioned plots consist of two independent axes: the bit stream (n), and the phase (T). In each figure header, a reference stream is specified, which is denoted as sequence $<x_{i+j}>$ in equation 6.13. This stream participates in the computation of all auto and cross-correlation coefficient functions. In all, there are W of these functions, each corresponding to a particular bit sequence $<x_i>$. At every intersection point in the grid, there is a corresponding absolute value of the correlation coefficient (i.e., between 0 and 1). From these values, a great deal of information can be attained as to the amount of dependencies inherent in the process. The lower the level of dependency, the better the generator.

6.4.6.1. Correlation of Original Driving Engines

For an original driving engine based on a statistically independent model, the values of the coefficients are, as expected, relatively low everywhere. The only exception is when the coefficient is computed by auto-correlating the reference stream with itself for a phase of $T = 0$. If this particular condition is satisfied, any independent, or dependent, machine will obviously have a coefficient "peak" of value 1 (see Figure H.1).

For an original driving engine based on a maximal-length LFSR, there is an extremely large space-phase correlation ridge (see Figure H.2). Furthermore, by Lemma 6.2, if an entire maximal-length test set is used in the construction of the space-phase correlation plot, there can be only two different correlation coefficient values possible.

Lemma 6.2:

Every maximal-length LFSR of size W has a two-level auto-correlation coefficient function given by (see [Golomb82])

$$R^{A}{}_{LFSR}(T) = \begin{cases} 1 & ; \text{for } T = 0,\ n = N_R \\ -1/(2^{W} - 1) & ; \textit{elsewhere}, \end{cases} \tag{6.14}$$

and a two-level cross-correlation coefficient function given by

$$R^{C}{}_{LFSR}(T) = \begin{cases} 1 & ; \text{for } T = n - N_R \\ -1/(2^{W} - 1) & ; \textit{elsewhere}, \end{cases} \tag{6.15}$$

where N_R is the reference stream number.

In this case, the coefficient satisfying $T = n - N_R$ is equal to one and the coefficients elsewhere, simply equal to $1/(2^W - 1)$. This is true independent of the chosen reference stream. However, because Figure H.2 is made up of a considerably smaller sample test set, there are more than just two coefficient values. Residual ridges, occurring as a result of the linear recursiveness property, or "taps," of the LFSR structure give rise to innumerable different correlation values. These smaller subordinate ridges are completely suppressed if the entire maximal-length test set is used in the determination of the plot. In either case, a large dominating space-phase correlation ridge remains, and is most undesirable. For example, when testing memory induced faults, such as stuck-open or intermittent faults it is necessary to apply a "set-up" pattern followed immediately by a specific "testing" pattern. Because an LFSR cannot generate certain two-bit test patterns back-to-back, there is a prevalent possibility that certain faults will go undetected.

For an original driving engine based on a maximal-length Rule 90/150 HCA, there is no large space-phase correlation ridge. In fact, as indicated by the different referenced plots, the worst a Rule 90/150 HCA can be occurs at the very ends of the structure, and it is still as good as an LFSR (see Figure H.3b)). In other words, a Rule 90/150 HCA behaves exactly as an LFSR only for the last two sites on either end of the structure. However, if the very last sites of a Rule 90/150 HCA are occupied by Rule 150 cells, instead of Rule 90 cells, this problem is eliminated. Independent of the other site occupancies, as long as the very last sites are Rule 150 cells, there will be a substantial reduction in the coefficient value at $(T = 1, n = 0)$. If an entire maximal-length test set is used in the construction of the space-phase correlation plot, the coefficient value at $(T = 1, n = 0)$ will be reduced from 1 to $1/(2^W - 1)$.

Similar to the maximal-length LFSR, the maximal-length Rule 90/150 HCA also has a unique number of possible correlation coefficients. It is conceivable that Lemma 6.3 may be extended to a postulated generality for all maximal-length linear hybrid cellular automata.

Lemma 6.3:

Every maximal-length Rule 90/150 HCA of size W has a two-level auto-correlation coefficient function given by

$$R^{A}{}_{HCA}(T) = \begin{cases} 1 & ;\ \text{for } T = 0,\ n = N_R \\ -1/(2^W - 1) & ;\ elsewhere, \end{cases} \tag{6.16}$$

and a two-level cross-correlation coefficient function given by

$$R^{C}{}_{HCA}(T) = \begin{cases} 1 & ;\ \text{for } T = k_n \\ -1/(2^W - 1) & ;\ elsewhere, \end{cases} \tag{6.17}$$

where N_R is the reference stream number, and k_n is some specified amount.

Although the Rule 90/150 HCA has a two-level cross-correlation coefficient function, the value k_n is generally not known. What is known, however, is that, like the maximal-length LFSR, the W bit sequences are the same, except for some phase offset, and are linearly recurring. Because they are linearly recurring, a characteristic polynomial may be written which, for a maximal-length sequence (also called pseudo-noise (PN) sequence), is irreducible. Hence, it is then possible to establish, without

the necessity of performing exhaustive simulation, if a given Rule 90/150 HCA combination leads to maximal-length PN sequences. Recently, Serra *et al.* [Serra89] have found that it is, in fact, possible to determine an appropriate maximal-length Rule 90/150 HCA combination, corresponding to the taps of an associated maximal-length LFSR, based on LFSR/Linear CA isomorphic properties. This method will have the benefit of significantly reducing the computational time required to find maximal-length linear HCA machines.

When compared to the maximal-length LFSR driving engine, the maximal-length Rule 90/150 HCA offers a much better space-phase correlation plot due to the large, and, as of yet, unsubstantiated values of k_n. Even when the output streams of the LFSR are ''scrambled,'' there are still n peaks contained within the space-phase correlation plot. Overall, the Rule 90/150 HCA, with its regularity and modularity, has far better correlation characteristics, which makes it much more practical in detecting a wide assortment of faults. (It should be further noted that ''better'' randomness can be attained by using an arbitrarily ''mixed'' combination of Rule 90 and 150 HCA cells forming the overall maximal-length Rule 90/150 HCA structure. If the number of consecutive Rule 90 or 150 cells are limited in length, if possible, there will be a reduction in the size of the self-similar structures which evolve in the state-time evolution. Because these pictorial structures indicate dependencies inherent in the evolution, if their respective sizes are reduced, there will be an associated reduction in the amount of dependencies plaguing the time evolution mechanism.)

For an original driving engine based on a non-maximal-length Rule 30 CA, there is an exponentially decreasing space-phase correlation ridge (see Figure H.4). Due to the existence of strange attractors in a Rule 30 CA, maximal-length test sets are not capable of being generated. Therefore, depending on the initial seeding, the maximum non-replicating test set will vary, and so too will the characteristics of its space-phase

correlation plot. Generally speaking, however, as long as a sufficiently large sample function of its time evolution is non-repeating, the correlation ridge will not vary to any large extent. So, for a typical Rule 30 CA, it is expected that the coefficient values of the space-phase correlation ridge will be of significance until $T \geq 16$. This leads to a definite amendment over the maximal-length LFSR, but at the expense of a considerably smaller usable test set.

6.4.6.2. Correlation of Weighted Configurations

As evident from the space-phase correlation plots, the structures formed by adding different configurations of weighting logic appear to have correlations directly related to the original driving engines. The most noticeable effect is a ''broadening'' in some of the original correlation peaks and/or ridges.

For the sharing configuration of any driving engine, whenever there is a peak of approximate value 1 to be found, there are also two adjacent coefficients of values approximately equal to one-third. Wherever there is a large space-phase correlation ridge, its peaks evolve along $T = n + c$, where c is a constant defining the beginning location of the ridge. For the Rule 90/150 HCA configurations, it should be pointed out that the unsettled coefficients are the result of having Rule 90 cell occupancies at the end sites of the original generator. If Rule 150 cells are used instead, the problem peaks will be suppressed. From the space-phase correlation plots of the zero spacing configurations, there are, once again, some trends which are generally shared. For instance, wherever there is a space-phase correlation ridge, it happens to evolve along $T = \frac{1}{2}n + c$; and, the peaks contained therein are also broadened. Finally, in the N/2 configuration, the LFSR based configuration has a broadened space-phase correlation ridge that evolves along $T = n + c$. In the very same configuration, the Rule 90/150 HCA shows no appreciable correlation peaks, when the end effects are suppressed.

From the above correlation analysis, it can be clearly stated that the CA based configurations have much more attractive space-phase correlation plots than those based on the LFSR. In particular, the Rule 90/150 HCA based configurations have the most acceptable properties of all the deterministic machines examined, and certainly present themselves as excellent parallel WTPG candidates. Of these configurations, the zero spacing configuration offers a distinctive advantage in the detection of stuck-open-faults, beyond those based previously on the LFSR, with the added benefit of a nearest neighbour connection scheme. This is the configuration which is most recommended for implementation in a weighted test pattern-BIST environment.

6.4.7. Bit Sequence Tuple Lengths

Before a bit sequence can be considered effectively random it must possess apparent statistical independence. That is to say, besides conforming to the appropriate distribution, the emanating bits must appear as if they were generated independently. If this is the case, then the number of k-tuples contained within a given sequence length should, by Lemma 6.4, be relatively consistent.

The number of "k-tuples", as defined here, include the occurrence of overlapping tuples. For example, within a single run of four ones, there are four k-tuples of length one; three k-tuples of length two; two k-tuples of length three; and, one k-tuple of length four. (A so called "run" is a contiguous sequence of ones isolated by one or more zeros.)

Lemma 6.4:

The expected number of k-tuples, $T_L{}^{(k)}$, found in the evolution of a single Bernoulli random variable can be obtained by

$$E[T_L^{(k)}] = (L-k+1)\cdot p^k, \tag{6.18}$$

where p is the probability of generating a 1 ($1-p$ the probability of generating a 0), L is the number of successive Bernoulli trials, and k is the tuple length.

Proof:

Let X be a Bernoulli random variable. Then, there are only two possible outcomes for any trial T. Thus,

$$X_T = \begin{cases} 1 & ;\ \textit{with probability } p \\ 0 & ;\ \textit{with probability } (1-p). \end{cases} \tag{6.19}$$

For an independently generated Bernoulli sequence of length L, the total possible assortment of k-tuple lengths range from 0 to L, or $T_L^{(k)} \in \{0, 1, \ldots, L\}$. So, corresponding to each tuple length, an exact expression can be formulated which determines the number of k-tuples for any Bernoulli sequence. Specifically, the number of k-tuples, for $1 \le k \le L$, can be obtained directly by the following expressions:

$$T_L^{(1)} = \sum_{T=0}^{L-1} X_T, \tag{6.20}$$

$$T_L^{(2)} = \sum_{T=0}^{L-2} X_T X_{T+1}, \tag{6.21}$$

$$T_L^{(3)} = \sum_{T=0}^{L-3} X_T X_{T+1} X_{T+2}, \tag{6.22}$$

•

•

•

•

$$T_L^{(L)} = X_0 X_1 X_2 \cdots X_{L-1} . \tag{6.23}$$

By way of mathematical induction, the general form

$$T_L^{(k)} = \sum_{i=0}^{L-k} \prod_{j=0}^{k-1} X_{i+j} \tag{6.24}$$

can be deduced.

Since it is assumed that the statistical mechanism governing the bit sequence evolution is described by an independent Bernoulli distribution, the expected number of k-tuples can be found for $1 \le k \le L$. Using equation 6.24, and the property

$$E[\sum_i Y_i] = \sum_i E[Y_i] , \tag{6.25}$$

we can obtain the following expectation values:

$$E[T_L^{(1)}] = L \cdot p , \tag{6.26}$$

$$E[T_L^{(2)}] = (L-1) \cdot p^2 , \tag{6.27}$$

$$E[T_L^{(3)}] = (L-2)\cdot p^3, \tag{6.28}$$

•

•

•

•

$$E[T_L^{(L)}] = p^L. \tag{6.29}$$

Applying mathematical induction once more, equations 6.26 through 6.29 can be combined to form a general expression for the number of k-tuples; thus, the expected number of k-tuples ($1 \leq k \leq L$, and $L \geq 1$) can be calculated by

$$E[T_L^{(k)}] = (L-k+1)\cdot p^k, \tag{6.30}$$

where $E[T_L^{(k)}] \in [0, L]$.

In accordance to equation 6.30, an absolute, or expected, tuple profile may be used as a basis for which comparisons, amongst a variety of machines and configurations, can be made.

6.4.7.1. Tuple Profiles-Parallel Weighted Test Pattern Generation

As illustrated in Figure I.1 (Appendix I), the tuple profiles of both zero spacing and N/2 configurations, based on a statistically independent non-linear model are very close to what is expected. This result remains true independent of which bit sequence,

n, is selected. Furthermore, because of the sophisticated nature of this model, these profiles are indicative of what would have to be considered typical statistically independent behaviour.

For a maximal-length LFSR, Figure I.2 shows that the sequences logically combined by the zero spacing (or sharing) configuration does not give the appropriate tuple profile. It is only for the N/2 configuration that the tuple profile coincides with that which is expected. These findings also happen to be independent of the chosen bit sequence, due to the fact that every consecutive bit sequence is identical with the exception of an accompanying phase shift of $T = 1$. The necessity of using a spacing of J=N/2-1, in order to achieve proper LFSR generated tuple profiles, has been addressed by earlier work concerning Chin and McCluskey [Chin84]. In a logical approach, they discuss the rationale behind the selection of the N/2 configuration for the case of a 25% weighted random bit sequence. To support the finding presented here, this can be analogously extended to the case of a 75% weighting.

When the tuple profiles of a maximal-length Rule 90/150 HCA are examined, we find that the results are no longer completely independent of which bit sequence is selected (see Figures I.3 and I.4). For the zero spacing configuration, the general trend is that the bit sequences which are furthest away from the end sites provide the better tuple profiles. In contrast, the tuple profile along $n = 0$ for the Rule 90/150 HCA is no better than those offered by the LFSR zero spacing configuration. However, if Rule 150 cells are used in place of the Rule 90 cells at the end sites of the original generator, there will be a great improvement in the tuple profile of bit sequence $n = 0$. In fact, it will appear as good as any of the other tuple profiles.

If the Rule 90/150 HCA is reconfigured in the N/2 configuration, there are only moderate gains in the tuple profiles to be attained. This is because the N/2 configuration has statistical characteristics reminiscent of those found in the zero

spacing configuration. In light of this, and the potential advantage in overall circuit performance because of a reduced wiring complexity, it is the zero spacing configuration of the Rule 90/150 HCA which lends itself for use in BIST.

The tuple profiles of Figure I.5 are generated using the configurations based on a Rule 30 CA. The findings are much the same as for the Rule 90/150 HCA. Deviations from the expected tuple profile are evident, but are not serious enough to indicate that a Rule 30 CA WTPG suffers from inherent dependencies which would make it a less effective generator. It should be noted, however, that the periodic boundary conditions of the Rule 30 CA, coupled with the fact that the rule itself is non-maximal-length, may be sufficient cause to prevent it from being implemented in a BIST strategy.

6.4.7.2. Tuple Profiles-Concatenated Weighted Test Pattern Generation

As an added means of testing for statistical independence, the tuple profiles of bit streams formed by concatenating individual test patterns are examined. This test is of particular interest if all the bits of a machine are combined to form a useful stream $W \cdot (2^W - 1)$ bits long. Figure I.6 shows that the tuple profiles for the sharing configuration do not coincide at all with what is expected. Even a WTPG based on a statistically independent model would fail, by virtue of the large dependencies introduced spatially by the shared logic. According to Figure I.7, the concatenated bit streams of all the implementable machines, under the zero spacing configuration, do not differ appreciably from one another. They all conform relatively close to the expected profile. Finally, as illustrated by Figure I.8, under the N/2 configuration, only the LFSR based tuple profile is unacceptable. Unfortunately, this also happens to correspond to the most acceptable LFSR based parallel WTPG configuration.

Chapter 7
Cellular Automata Based Testing Apparatus

With the emergence of VLSI circuits and the combinatorial explosion associated with exhaustive testing, a concentrated effort has resulted in a design methodology known as Design For Testability (DFT). DFT includes structured techniques such as Level Sensitive Scan Design (LSSD), Scan Path, Scan/Set Logic, Random Access Scan, as well as Built-In Self-Test (BIST) techniques such as Autonomous Testing, Syndrome Testing, and Built In Logic Block Observation. For any of these methods which rely on on-chip pseudorandom, or weighted random, test pattern generation, an overall improvement in the testing environment will result by introducing Cellular Automata (CA) based structures. Specifically, this is accomplished by replacing Linear Feedback Shift Register (LFSR) based test pattern generators with those based on cellular automata (see [Hortensius87]).

7.1. Cellular Automaton Logic Block Observer

A variation of a Built In Logic Block Observer (BILBO) is presented here so as to demonstrate the practicality of one-dimensional CA based structures to BIST. The basic testing concept of what has become to be known as a Cellular Automaton Logic Block Observer (CALBO) is similar to that of a BILBO, in that its selectable function option modes are identical. Physical measures, such as Input/Output complexity

associated with the number of pins, and requirements of external automated test equipment, are therefore similar in most respects. This implies that wherever a BILBO is incorporated in a BIST strategy, a CALBO may be successfully used in its place, with the added benefit of superior employment of pseudorandom test patterns.

Measures of controllability, observability and predictability, are also similar, in that the register latches utilized during testing are the same for both the BILBO and CALBO. The decision to employ either of these techniques, then, rests upon concerns such as area overhead, and possible parasitic degradation of the original system by the added test circuitry. An additional factor to be considered is the time performance of the test circuit itself, which will be severely degraded for large systems if across-chip communication is necessary.

Four immediate benefits for the CALBO approach are apparent from the discussion which follows: i) the communication is local, being restricted to nearest neighbour cells providing freedom from the communication constraints of an LFSR; ii) the cells are regular and topologically similar to one another, in contrast to the increasing complexity of an LFSR layout as the number of sites increases; iii) routing for the test circuit is no more complicated than the original interconnection of latches (i.e., topological complexity is contained); and iv), the ability to pass any random number test arises naturally from the chaotic class 3 (autoplectic) behaviour of the cellular automaton, and the random numbers are distributed over the entire system.

7.2. Practical Considerations

The topology of a BILBO largely consists of an LFSR, where selected outputs are tapped and fed back through exclusive-OR gates to a 2X1 MUX. One immediate difficulty in the design lies in selecting the appropriate feedback taps; that is, the taps

are not independent of the size of the LFSR for maximal-length polynomial division. Another potential difficulty arises when having to send the higher order tap(s) back to the multiplexer input, as this distance is normally a significant fraction of the chip and grows (linearly) with W, the number of cells. A technological fix may include the use of larger exclusive-OR gates, hierarchical drivers, or wider and thicker metal lines to circumvent problems associated with delay and current density. The cost involved with this approach arises from a reduced global topological uniformity, a major design deterrent for increasing W. In any event, a time penalty is experienced of $\Omega(\log W)$, or even worse, of $\Omega(W)$, if current density limitations are taken into account [Card86]. By adopting cellular architecture, a CALBO circuit does not suffer from this symptom, as the required communication amongst cells is restricted to nearest neighbours. On the other hand, the hardware of the basic cell has been modestly increased to accommodate the required storage of the present state, the local logic to implement a given rule, and the incorporation of transmission-gate multiplexers.

Another area of further investigation is the system size W at which the utility of the CALBO approach is expected to supercede that of the BILBO. The measures for this comparison are at present a moving target as are many decisions in DFT. A justifiable suggestion is the utilization of an established measurement criterion, such as the AT (area-time) metric. The constant factors in A and T are of immediate relevance to non-asymptotic design decisions, whereby these factors are more easily accounted for in a relative manner. In addition, a heuristic factor is introduced based upon wiring difficulty and technological fixes.

Since it is most beneficial to have the services of a total uniform CALBO topology, construction may be served by the Rule 30 CA unit cell of Figure 7.1. In this way, cells may be added, or deleted, without any disturbance to existing cells. But, here are two immediate problems which come to mind that renders the Rule 30

CALBO unadvantageous: i) its pseudorandom number generation is non-maximal length; and ii), because it includes periodic boundary conditions, there is a serious degradation introduced as the result of an across-chip interconnection. As a design alternative, to rid the test engineer of such problems, a Rule 90/150 CALBO may be constructed with far less overhead. Based on the Rule 90 and Rule 150 unit cells of Figure 7.2, a combination of cells can be found for any particular size CALBO, which will yield maximal-length operation.

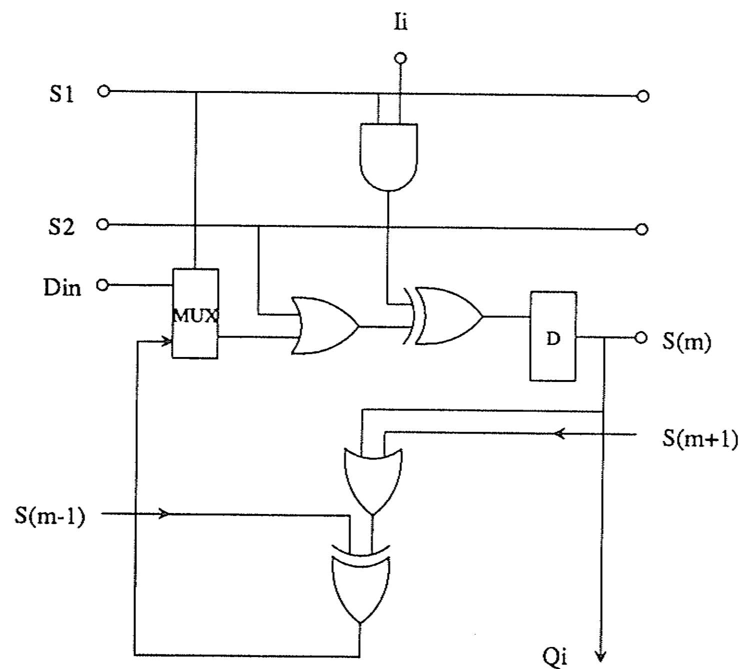

Figure 7.1: *Non-linear Rule 30 CA CALBO Cell.*

Another attractive feature inclusive in this design is that there is no across-chip communication, whatsoever, (only null boundary conditions). Even though a circuit modification to a Rule 90/150 CALBO may be cause for redesign, it may be accomplished with great ease since circuit regularity is well maintained throughout.

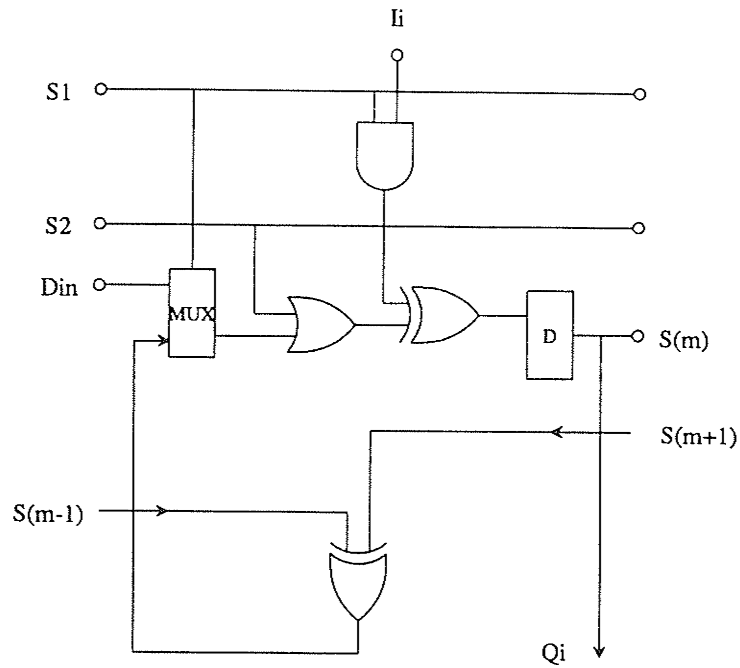

Figure 7.2a): *Linear Rule 90 CA CALBO Cell.*

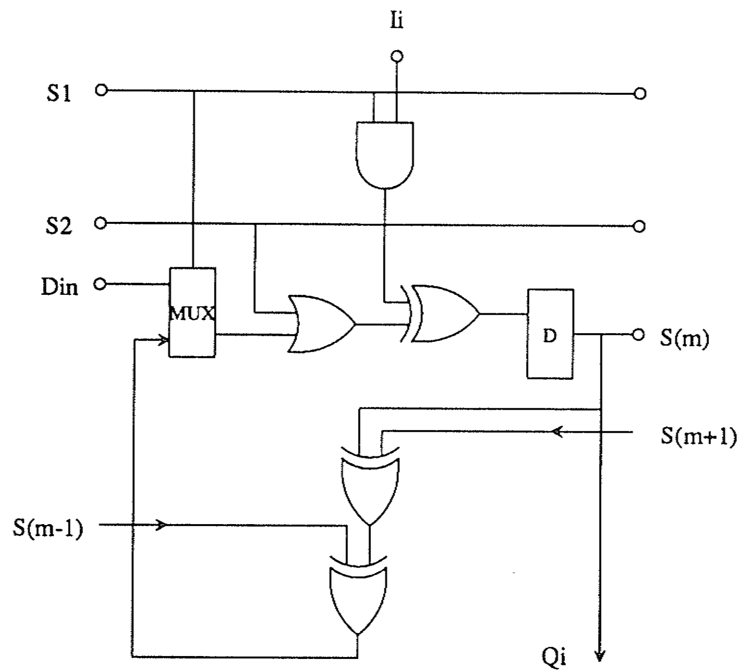

Figure 7.2b): *Linear Rule 150 CA CALBO Cell.*

Upon selection of a Rule 90/150 CALBO, a conservative estimate of at most two times the area per cell, as the unit cell for the BILBO, is required. This estimate comes about as a consequence of considering the area requirements of both Rule 90 and Rule 150 cells. It accounts for the additional logic necessary to perform the cellular automaton operation at each cell along the array, as well as for the 2X1 MUX which allows each cell to feed back onto itself. The D-type flip-flop, analogous to the case of a BILBO, acts as a one-bit memory device necessary for each operational mode. For the present discussion, the increased local wiring complexity associated with the CALBO is traded against the increased global wiring complexity of the BILBO. The area for the BILBO increases as C_1W, with W the number of register cells, and C_1 the unit cell area; whereas, the area for the CALBO increases approximately as $2C_1W$. This ignores the area of the additional wiring in the BILBO, which will unquestionably prove problematic for sufficiently large W. A time complexity comparison favours the CALBO since communication is restricted to nearest neighbours; whereas, communication in the BILBO, in the general case, extends over a considerable fraction of the chip width. In the hierarchical driver scheme, we assume this time to increase as $C_2 \log n$ for the BILBO; it is simply C_2 for the CALBO. The AT metric implies that the CALBO circuit approach is preferred to the BILBO circuit when $W \geq 4$, since the AT metric for the BILBO, $C_1C_2W\log W$, and CALBO, $2C_1C_2W$ are equivalent when $W = 4$. (Employing an AT^2 metric would mean that the CALBO is preferred above $W = 3$.) If, instead, the delay associated with communication in the BILBO was dominated by current density limitations, the time would go as C_2W. Once again, the AT metric would imply that the CALBO circuit approach is preferred to the BILBO circuit, but this time for $W \geq 2$. The AT metric comparisons supports the finding that the CALBO is particularly well suited to BIST for those situations where pseudorandom testing is applicable, as we are attempting to capitalize upon the virtues of the CALBO over the BILBO.

7.3. Signature Analysis Using Cellular Automata

A further issue, which must also be addressed at this time, is in regards to the appropriateness of the CALBO circuit for signature analysis. In the LFSR based BILBO circuit, signature analysis is usually performed via a Multiple-input Signature Register (MISR), in a similar manner to the generation of cyclic redundancy codes. It is also the contention of the CALBO circuit to be capable of compacting data with comparable error detection, as a direct consequent of the Rule 90/150 HCA's rich maximal-length pseudorandom number generation capability. Recent analysis by Hortensius [Hortensius89] and Serra *et al.* [Serra89] have indeed shown that this is, in fact, true. By means of analytical deduction, along with extensive simulation, it has been categorically confirmed that the data compaction capability of both maximal-length LFSR and HCA based MISRs are generally the same. This is a direct consequence of the isomorphism between the two finite state machines.

7.4. Weighted Cellular Automaton Logic Block Observer

In so far as W is sufficiently large, such that probabilistic testing is the only practical test means available, certain one-dimensional cellular automata have demonstrated to be attractive alternatives to the LFSR. Such is the case for the Rule 90/150 HCA, where it can be successfully implemented as a stand alone pseudorandom test pattern generator/signature analyzer, or in more advanced CALBO technology.

Furthermore, a "Weighted" Test Pattern Generator (WTPG), or "Weighted" Cellular Automaton Logic Block Observer (WCALBO), may be constructed from those qualifying cellular automata by utilizing external weighting logic, with some informal ad hoc design techniques. As discussed in a previous section (Sec. 6.4: Statistical Evaluation of WTPGs), there is, once again, great improvements noted over

conventional LFSR based designs. This novel find represents an important contribution to BIST, since many pseudorandom-resistant circuits are found, in an overwhelming number of instances, to be weighted random pattern testable; and, they are dependent on apparent statistical independence for detecting a wide assortment of faults (such as, memory induced faults, and transition faults). Waicukauski and Lindbloom [Waicukauski88] have subjected this class of pseudorandom-resistant circuits to weighted random pattern testing. They have found that test lengths, when compared to unbiased pseudorandom test sets, have been reduced by orders of magnitude, while fault coverage is significantly increased.

The hardware considerations for this new test circuitry must provide accommodation for the delivery of a multitude of pre-established weighted distributions. This must be done in such a manner that the amendments of an overall reduced test set will outweigh the complications incurred by an increased area overhead. For instance, a cost-effective parallel WTPG may incorporate, say, a tri-distribution (e.g., 25%, 50%, and 75%), as it is expected that a wide range of potential pseudorandom-resistant faults can be exposed with minimal additional logic, in reasonable time. The inclusion of an array of gates, consisting of NORs and NANDs, will take care of the generation of the 25% and 75% distributions, respectively; whereas, the generation of the 50% distribution is simply attained by selecting outputs directly from the original driving engine. With the additional placement of 3X1 MUXs, a weighted distribution selectability is permitted for the testing mode of operation.

Chapter 8
Conclusions and Future Work

8.1. Conclusions

The main contribution of this thesis was in improving upon conventional Weighted Test Pattern Generator (WTPG) circuitry for Built-In Self-Test (BIST), thereby increasing the fault detection capability of the apparatus, and the means by which testability is achieved. The methodology is centered around the concept of employing one-dimensional Cellular Automata (CA) as alternatives to Linear Feedback Shift Registers (LFSRs). By using these structures as primary ''driving engines'' driving logic arrays, the result is a WTPG function with improved statistical properties, wiring complexities, and performance.

There were several statistical aspects of WTPG behaviour investigated:

- **State-time Visualization:** Using the human eye in a strictly subjective qualitative test, it was determined that the LFSR based WTPGs had very poor local and global randomness. The CA based configurations, on the other hand, had signs of only local self-similar structures, which were considered no where near as detrimental to randomness.
- **Density and Average Density:** The density evolutions of those LFSR based WTPGs were found to change by, at most, one incremental density. This implied that the emanating densities were considered serially correlated, and were thus

indicative of unacceptable random behaviour. For those CA based WTPGs, the density evolutions appeared statistically independent. When the average density evolutions were investigated, the LFSR based WTPGs could not attain their steady-state average in the duration observed, whereas those based on CA managed to attained their steady-state average rather quickly.

- **Probability Mass Function:** According to the density histogram evolutions, it was apparent that the density evolutions of the LFSR based WTPGs did not converge upon the desired Binomial PMF in an acceptable manner. In fact, when a goodness-of-fit test was performed, it was shown that all LFSR based WTPGs with a test length of 20,000 failed the χ^2-test. In contrast, due to the respectable histogram convergence of those based on CA, they successfully passed the χ^2-test. (This was true for all but the sharing configuration, under which even a WTPG based on a statistically independent driving engine failed rejectedly.)
- **Magnitude Spectrum:** To further investigate the global properties of the emanating density evolutions, their Fourier domains were examined. In doing so, it was found that the LFSR based WTPGs contained disproportionately more low frequency content, which was assessed as being most damaging to randomness on a global level. Further to this, it was also substantiated that the CA based WTPGs contained some additional high frequency content, but due to the limitations in the fixed sampling frequency of the original ensemble, was thought not necessarily detrimental to global randomness.
- **Auto and Cross-Correlation Coefficients:** From the space-phase correlation plots, it was generally shown that the LFSR based WTPG configurations contained large amounts of cross-correlation; however, as was the case for the CA based WTPGs, they produced much less cross-correlation. The Rule 30 CA based configurations contained what appeared to be an exponentially decreasing space-phase correlation

ridge, while for those based on the Rule 90/150 HCA, there was no appreciable quantity of correlation that would prove problematic.

When the original Rule 90/150 HCA driving engine was investigated in more detail, several properties were characterized: i) the finite-state machine possesses two-valued auto and cross-correlation coefficient functions, giving rise to Pseudo-noise (PN) bit sequences at each site; ii) there is a definitive linear recursive relationship dictating the evolution of the PN bit sequences; iii) the extra correlation coefficient values occurring on both end sites (under Rule 90 occupancies), for a time or phase shift of one, can be suppressed by replacing those Rule 90 cells, located at the end sites, with Rule 150 cells. (This occurrence is a direct result of fixed null boundary conditions.); and iv), when the placement of Rule 90 and 150 cells, making up the maximal-length Rule 90/150 HCA combination, is arbitrarily, but uniformly, distributed, self-similar structures in the state-time evolution are reduced. This is, in turn, characteristically better for randomness.

In general, it can be stated that the effect of a functional logic array on the evolution of a WTPG tends to produce a ''broadening'' in the space-phase correlation ridges of the original generator. Furthermore, in the Zero Spacing configuration, there is a relatively flat space-phase correlation plot for the Rule 90/150 HCA based WTPG. This property reflects its excellent ability to detect an assortment of memory induced faults for those instances when weighted random testing is applicable. The degree to which this is possible is unattainable by any parallel LFSR based WTPG.

- **Bit Sequence Tuple Lengths:** In order for a bit sequence to be considered effectively random, it must exhibit a particular weighting drawn in a independent, or ''apparently'' independent, fashion. To test for this, a bit sequence tuple test can be performed. For the LFSR based WTPGs considered, only the

N/2 configuration met the outlined requirements of "independence." Conversely, it was shown that both CA based WTPGs (i.e., Rule 30 CA, and Rule 90/150 HCA), in the much simpler Zero Spacing configuration, produced satisfactory bit sequence tuple profiles. With the added benefit of a concatenated tuple test, the N/2 LFSR WTPG, even with its proper individual bit sequence tuple profiles, indicated unacceptable concatenated tuple profiles. Under the same circumstances, the CA based WTPGs faired remarkably well.

In retrospect, it was determined that the CA based WTPGs exhibited much better local and global random properties, and, with reservations, appeared similar to the statistically independent model. By virtue of the local communication architecture, and regular topology, of the CA based WTPGs, there is a reduced wiring complexity associated with the development of such BIST test circuitry. (In this regard, they are extendible to Scan techniques, such as Boundary Scan, and to observation techniques incorporating a weighted Cellular Automaton Logic Block Observer (WCALBO) with multiple distributions.) In particular, it was learned that the Rule 90/150 HCA, under the Zero Spacing WTPG configuration, demonstrated sufficiently acceptable random properties, with increased circuit performance, as a result of its ability to conform to VLSI. On the other hand, there were difficulties encountered with the LFSR WTPG in its N/2 configuration with respect to both randomness and wiring complexity, which precluded its acceptability.

8.2. Future Work

For future considerations, the effort should be directed toward developing methods of attaining W outputs from W inputs for those CA based WTPG

schemes involving original driving engines. It can also be considered important to study conventional chaotic CA and HCA with larger neighbourhoods, since they may allow a weighting function to be selected as a direct consequence of their rule numbers. In this way, W-bit weighted test patterns, with excellent pseudorandom characteristics, may be derived from W CA cells. Finally, so as to demonstrate the necessity of better WTPGs to DFT, in particular, BIST, actual fault coverage capability of the WTPGs based on CA and LFSRs should be compared.

References

1. **[Abadir85]** Magdy S. Abadir and Melvin A. Breuer, "A Knowledge-Based System For Designing Testable VLSI Chips," *IEEE Design and Test*, pp. 56-68, 1985.

2. **[Agarwal81]** V.K. Agarwal and A.S. Fung, "Multiple Fault Testing of Logic Circuits by Single Fault Test Sets," *IEEE Transactions on Computers*, vol. C-30, No. 11, pp. 854-855, November 1981.

3. **[Aylor86]** James H. Aylor, Barry W. Johnson, and Bruce J. Rector, "Structured Design For Testability in Semicustom VLSI," *IEEE Micro* pp. 51-57, February 1986.

4. **[Bate87]** J.A. Bate and D.M. Miller, "The Exhaustive Testing of Stuck-open Faults in CMOS Combinational Circuits", *Developments in Integrated Circuit Testing*, Academic Press: New York, 1987.

5. **[Bardell86]** P.H. Bardell and W.H. McAnney, "Pseudorandom Arrays For Built-In Tests," *IEEE Transactions on Computers*, vol. C-35, pp. 653-658, July 1986.

6. **[Brglez85]** F. Brglez, P. Pownall, and R. Hum, "Accelerated ATPG and Fault Grading via Testability Analysis," *Proc. IEEE International Symposium on Circuits and Systems*, pp. 695-698, 1985.

7. **[Burks70]** A.W. Burks, *Essays on Cellular Automata*, University of Illinois Press: Urbana, pp. 103-131, 1970.

8. **[Card86]** H.C. Card, W. Pries, and R.D. McLeod, "Contributions to VLSI Computational Complexity Theory from Bounds on Current Density," *Integration*, North-Holland Publishing Company: Amsterdam, pp. 175-183, June 1986.

References

9. **[Chin84]** C.K. Chin and E.J. McCluskey, "Weighted Pattern Generation for Built-In Self-Test," *Centre for Reliable Computing, Tech Rep. No. 84-7*, Stanford University: Stanford, CA, 1984.

10. **[Chin87]** C. K. Chin and Edward J. McCluskey, "Test Length for Pseudorandom Testing," *IEEE Transactions on Computers*, vol. C-36, No. 2, pp. 252-256, February 1987.

11. **[David86]** Rene David, "Signature Analysis for Multiple Output Circuits," *IEEE Transactions on Computers*, vol. C-35, pp. 830-837, September 1986.

12. **[Dewdney84]** A.K. Dewdney, "Computer Recreations," *Scientific American*, pp. 18-30, 1984.

13. **[Feugate88]** Robert J. Feugate, Jr., and Steven M. McIntyre, *Introduction To VLSI Testing*, Prentice Hall: Englewood Cliffs, New Jersey, 1988.

14. **[Fujiwara85]** Hideo Fujiwara, *Logic Testing and Design For Testability*, MIT Press: Cambridge, Massachusetts, 1985.

15. **[Golomb82]** S.W. Golomb, *Shift Register Sequences*, Holden-Day: San Francisco, California, 1982.

16. **[Hayes84]** Brian Hayes, "Computer Recreations," *Scientific American*, pp. 12-21, March 1984.

17. **[Hortensius87]** Peter D. Hortensius, *Parallel Computation of Non-Deterministic Algorithms in VLSI*, Ph.D. Dissertation, University of Manitoba, Winnipeg, Manitoba, Canada, 1987.

18. **[Hortensius89]** Peter D. Hortensius, R.D. McLeod, and H.C. Card, "Cellular Automata-Based Signature Analysis for Built-In Self-Test," *to appear in IEEE Transactions on Computers*, 1989.

19. **[Knuth81]** Donald E. Knuth, *Seminumerical Algorithms-The Art of Computer Programming*, Addison-Wesley Publishing Company: Massachusetts, 1981.

20. **[Law82]** Averill M. Law and W. David Kelton, *Simulation Modelling And Analysis*, McGraw-Hill Book Company: New York, 1982.

21. **[Packard85]** Norman H. Packard and S. Wolfram, ''Two-Dimensionsl Cellular Automata'', *J. Stat. Phys.*, pp. 901-948, 1985.

22. **[Podaima88]** B.W. Podaima, and R.D. McLeod, ''Weighted Test Pattern Generation for Built-In Self-Test using Cellular Automata,'' *Proc. Third Technical Workshop-New Directions for IC Testing*, pp. 195-205, October 1988.

23. **[Podaima89]** B.W. Podaima, and R.D. McLeod, ''Utilities and Statistics for Weighted Test Pattern Generators for Built-In Self-Test using Cellular Automata,'' *TR89-P1, University of Manitoba, Winnipeg, Manitoba, Canada*, 1989.

24. **[Press86]** W.H. Press, B.P. Flannery, and W.T. Vetterling, *Numerical Recipes-The Art of Scientific Computing*, Cambridge University Press: New York, pp. 381-396, 1986.

25. **[Pries86]** W. Pries, A. Thanailakis, and H.C. Card, ''Group Properties of Cellular Automata and VLSI Applications,'' *IEEE Transactions on Computers*, vol. C-35, pp. 1013-1024, December 1986.

26. **[Pries88]** W. Pries, *private communication pertaining to maximal-length Rule 90/150 HCA generators*, 1988.

27. **[Savir80]** J. Savir, ''Detection of Single Intermittent Faults in Sequential Circuits,'' *IEEE Transactions on Computers*, vol. C-28, No. 7, pp. 673-678, July 1980.

28. **[Savir83]** J. Savir, G. Diteow, and P.H. Bardell, ''Random Pattern Testability,'' *Digest of Papers, 13th Annual International Symposium on Fault-Tolerant Computing*, pp. 80-89, June 1983.

29. **[Savir84]** J. Savir and P.H. Bardell, ''On Random Pattern Test Length,'' *IEEE Transactions on Computers*, vol. C-33, No. 6, pp. 467-474, June 1984.

30. **[Schnurmann75]** H.D. Schnurmann, E. Lindbloom, and R.G. Carpenter, ''The Weighted Random Test-Pattern Generator,'' *IEEE Transactions on Computers*, vol. C-24, No. 7, pp. 695-700, July 1975.

31. **[Serra89]** M. Serra, T. Slater, J.C. Muzio, and D.M. Miller, ''The Analysis of Linear Cellular Automata and Their Aliasing Properties,'' *submitted to IEEE Transactions on Computer Aided Design of Integrated Circuits*, 1989.

32. **[Shedletsky77]** J.J. Shedletsky, ''Random testing: Practicality versus effectiveness,'' *Proc. 7th International Conference on Fault Tolerant Computing*, pp. 175-179, June 1977.

33. **[Sudhakar86]** Sudhakar M. Reddy, Madhukar K.Reddy, ''Testable Realizations for FET Stuck-Open Faults in CMOS Combinational Logic Circuits,'' *IEEE Transactions on Computers*, vol. C-35, pp.742-754, August 1986.

34. **[Trivedi82]** Kishor S. Trivedi, *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*, Prentice-Hall, Inc.: Englewood Cliffs, New Jersey, 1982.

35. **[Wagner87]** K.D. Wagner, C.K. Chin and E.J. McCluskey, ''Pseudorandom Testing ,'' *IEEE Transactions on Computers.*, vol. C-36, No. 3, pp 332-343, March 1987.

36. **[Waicukauski88]** J.A. Waicukauski and E. Lindbloom, ''Fault Detection Effectiveness of Weighted Random Patterns,'' *Proc. IEEE International Test Conference*, pp. 245-250, 1988.

37. **[Weste85]** Neil West and Kamran Eshraghian *Principles of CMOS VLSI Design* Addison-Wesley Publishing Company: Massachusett, 1985.

38. **[Williams84]** T.W. Williams, ''VLSI Testing,'' *IEEE Transactions on Computers*, pp. 126-136, October 1984.

39. **[Williams86]** T.W. Williams, *VLSI Testing*, North Holland: New York, 1986.

References

40. **[Wunderlich88]** Hans-Joachim Wunderlich, ''Multiple Distributions for Biased Random Test Patterns,'' *Proc. IEEE International Test Conference*, pp. 236-244, 1988.

41. **[Wolfram83]** S. Wolfram, ''Statistical Mechanics of Cellular Automata,'' *Reviews of Modern Physics*, vol. 55, No. 3, pp. 601-644, July 1983.

42. **[Wolfram84a]** S. Wolfram, ''Universality and Complexity in Cellular Automata,'' *Physica D10*, Elsevier Science Publishers B.V., pp. 1-35, 1984.

43. **[Wolfram84b]** S. Wolfram, ''Cellular Automata as Models of Complexity,'' *Nature*, vol. 311, No. 4, pp. 419-424, October 1984.

44. **[Wolfram84c]** S. Wolfram, ''Cellular Automata: Towards a Paradigm For Complexity,'' *Nature*, 1984.

45. **[Wolfram84d]** S. Wolfram, ''Computational Theory of Cellular Automata,'' *Comm. Math. Phys.*, vol. 96, pp. 15-57, 1984.

46. **[Wolfram85]** S. Wolfram, ''Origins of Randomness in Physical Systems,'' *Reviews of Modern Physics*, vol. 55, No. 5, pp. 449-452, July 1985.

Appendix A

Boolean Logic Probability Profiles

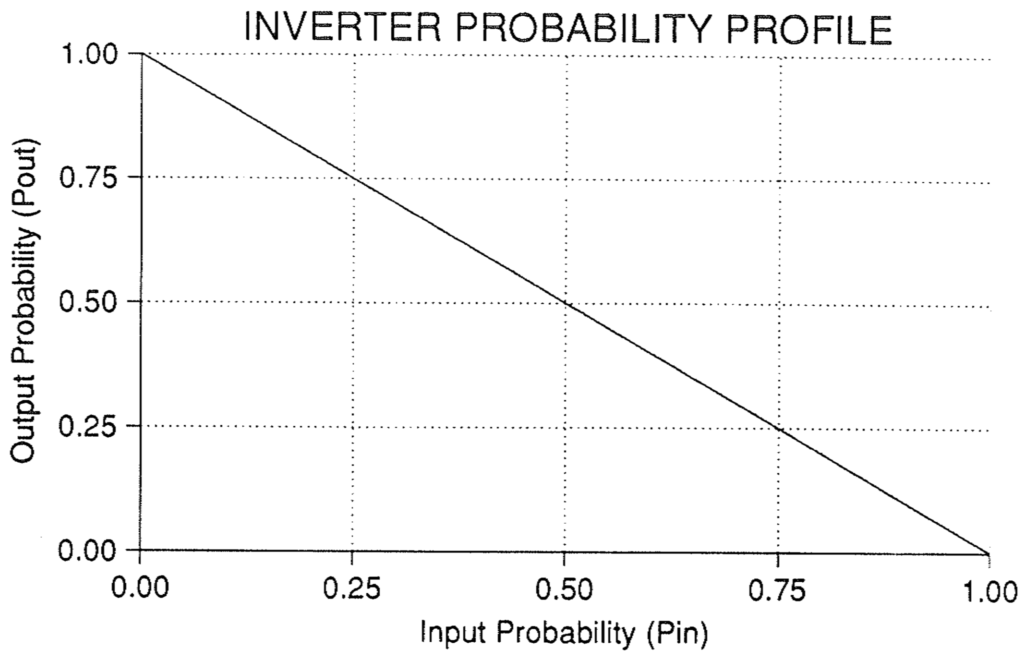

Figure A.1a): *Probability Profile for Inverter.*

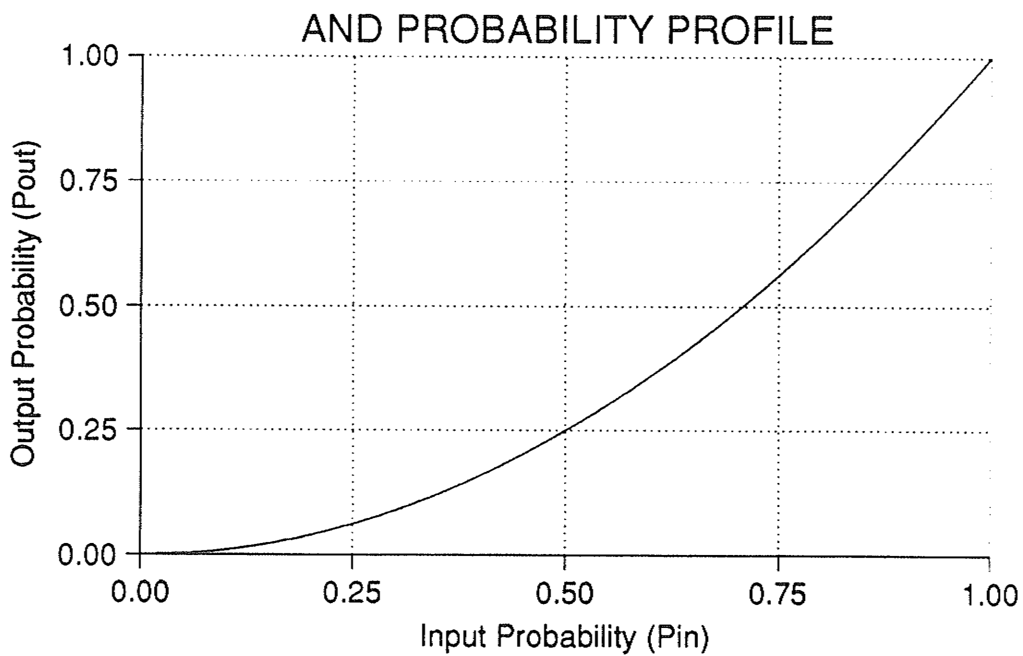

Figure A.1b): *Probability Profile for AND gate: equal input probabilities.*

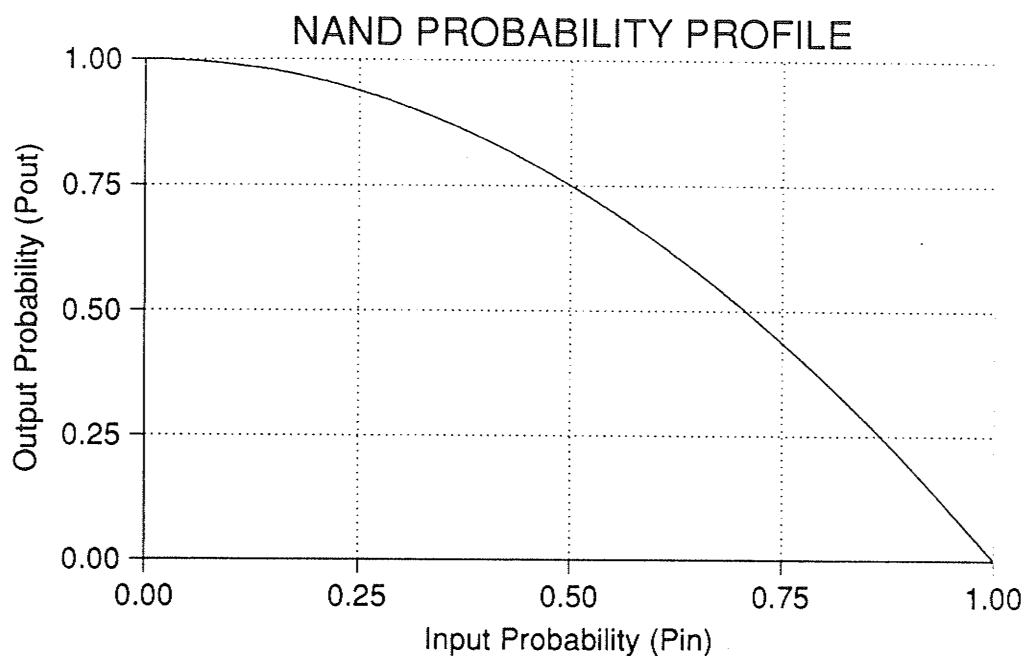

Figure A.1c): *Probability Profile for NAND gate: equal input probabilities.*

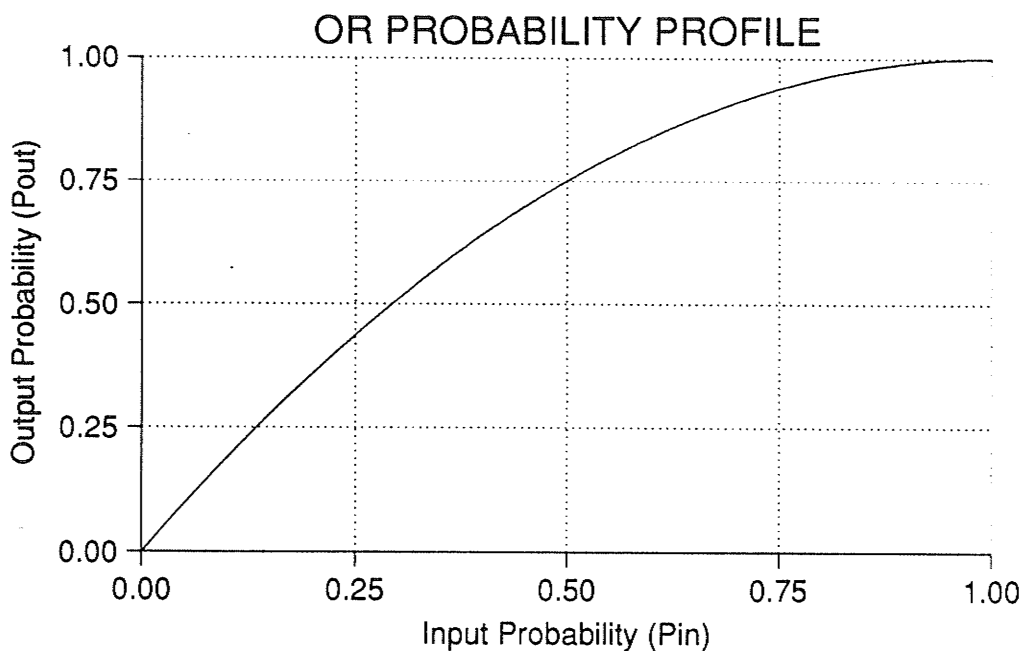

Figure A.1d): *Probability Profile for OR gate: equal input probabilities.*

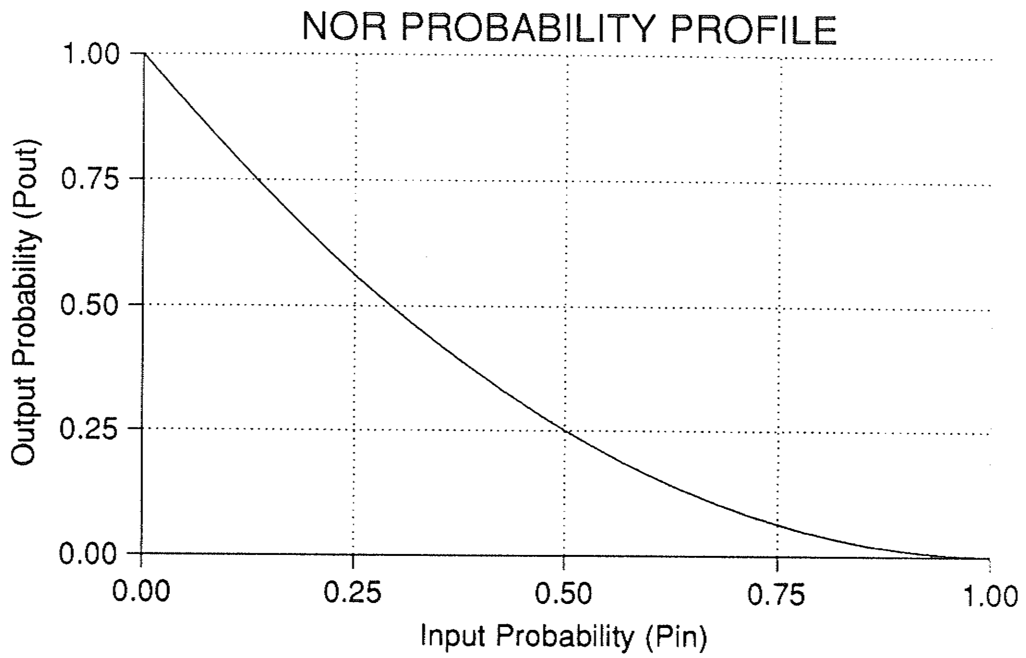

Figure A.1e): *Probability Profile for NOR gate: equal input probabilities.*

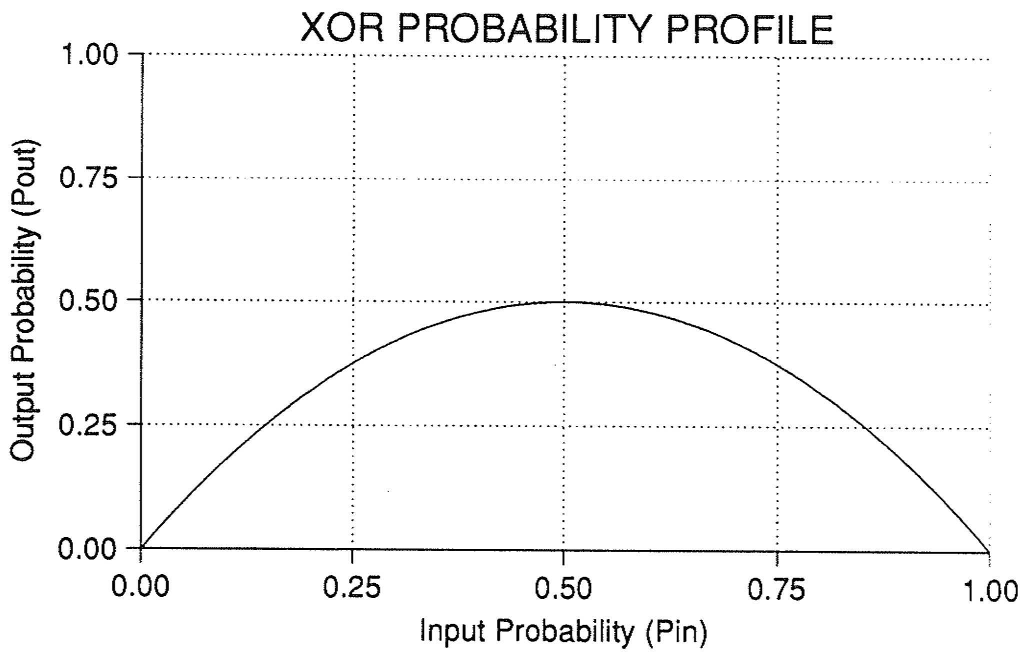

Figure A.1f): *Probability Profile for XOR gate: equal input probabilities.*

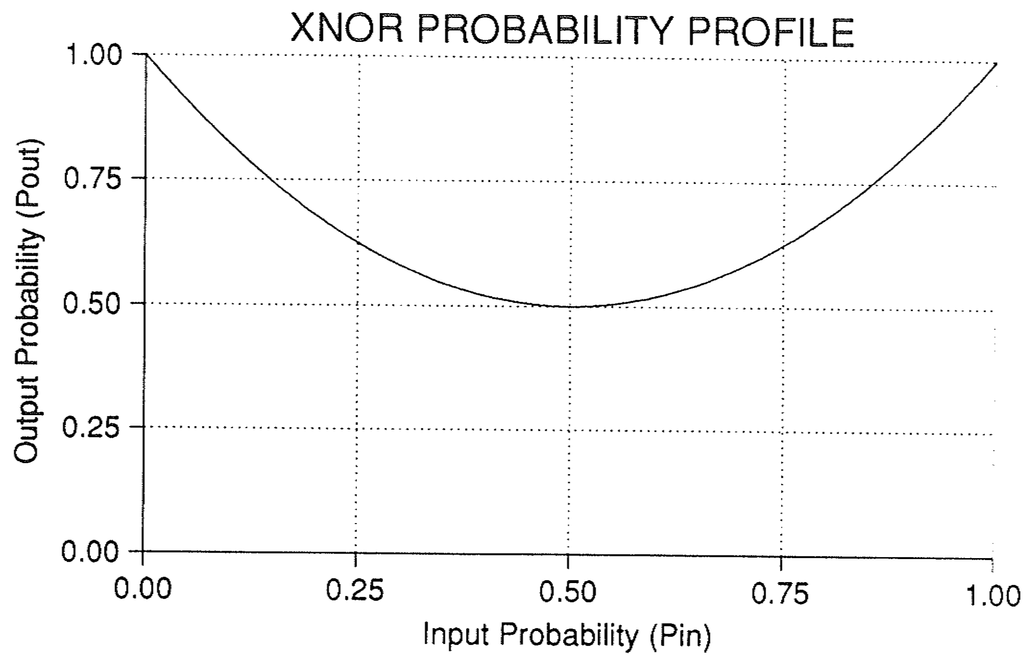

Figure A.1g): *Probability Profile for XNOR gate: equal input probabilities.*

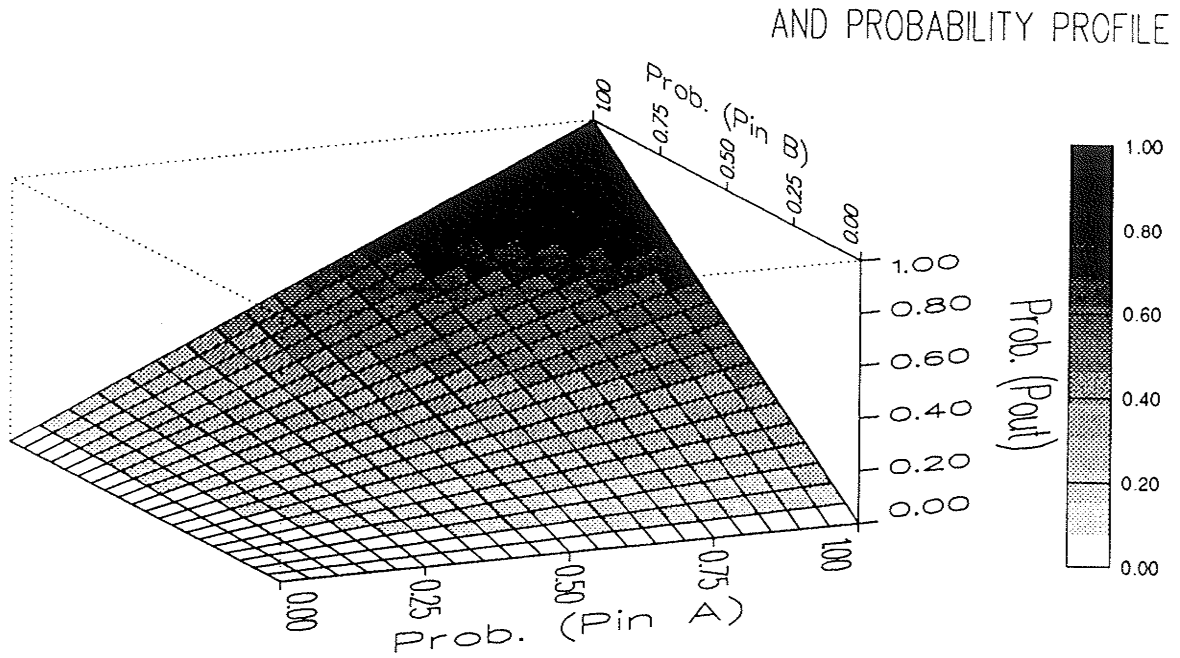

Figure A.2a): *Probability Profile for AND gate: variable input probabilities, Pin A and Pin B.*

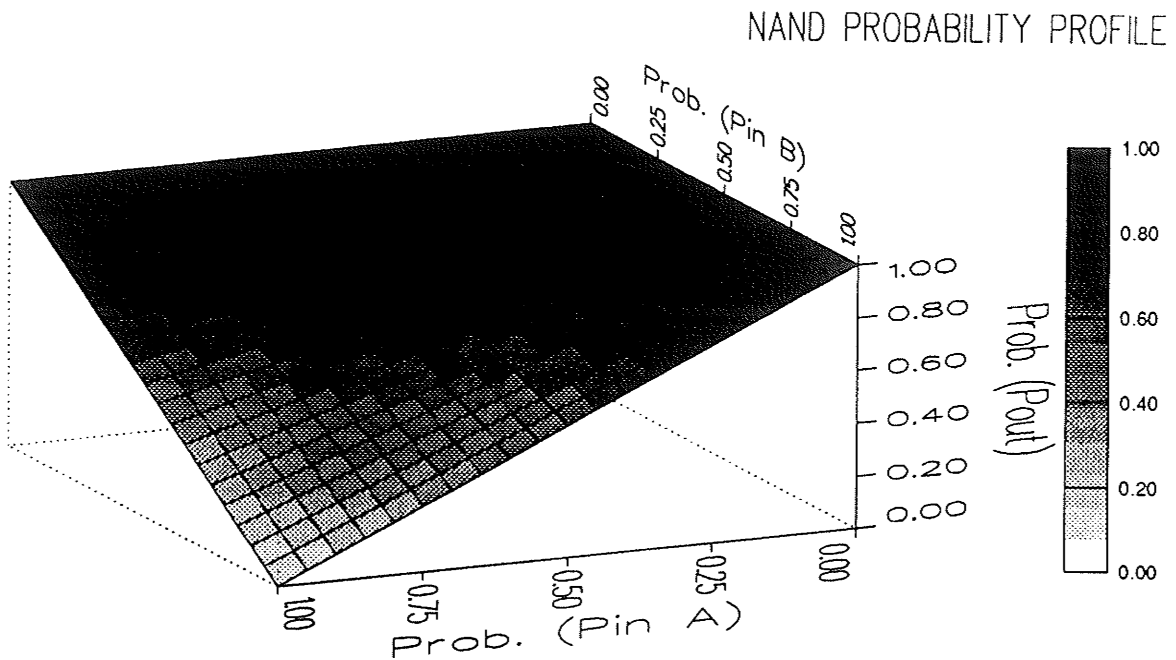

Figure A.2b): *Probability Profile for NAND gate: variable input probabilities, Pin A and Pin B.*

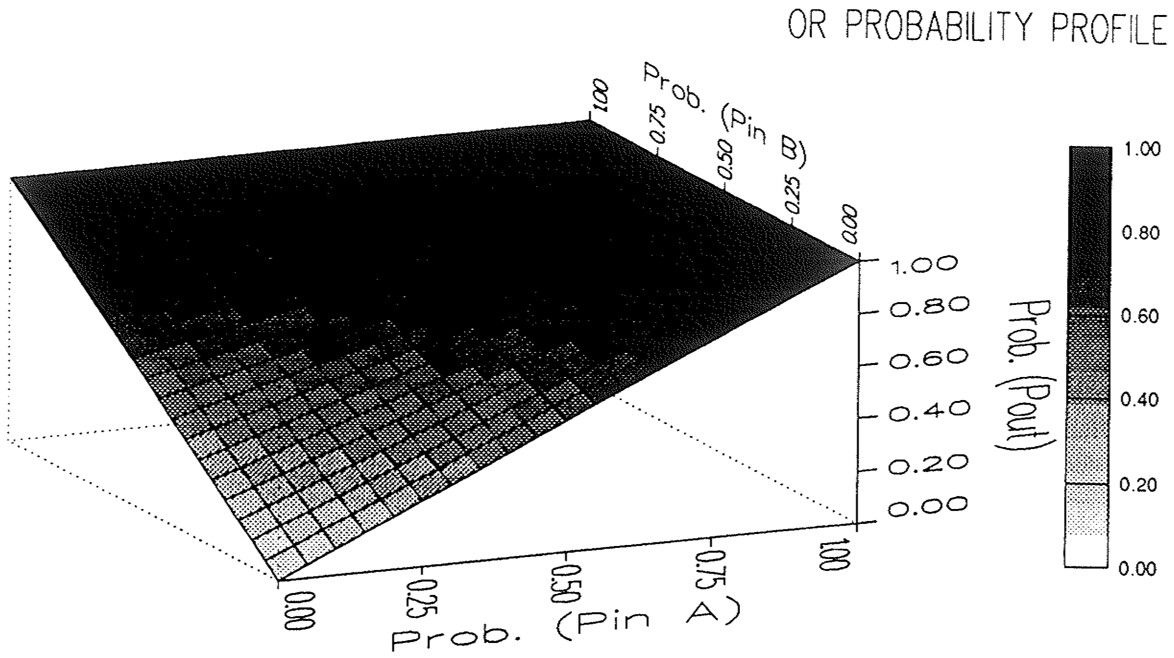

Figure A.2c): *Probability Profile for OR gate: variable input probabilities, Pin A and Pin B.*

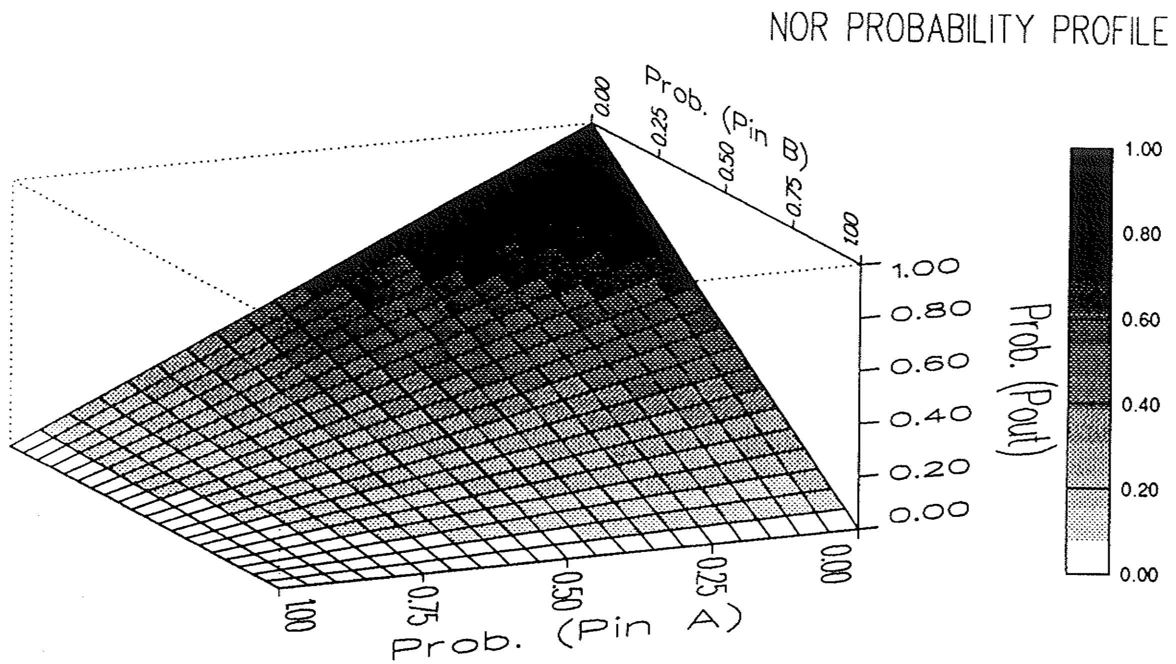

Figure A.2d): *Probability Profile for NOR gate: variable input probabilities, Pin A and Pin B.*

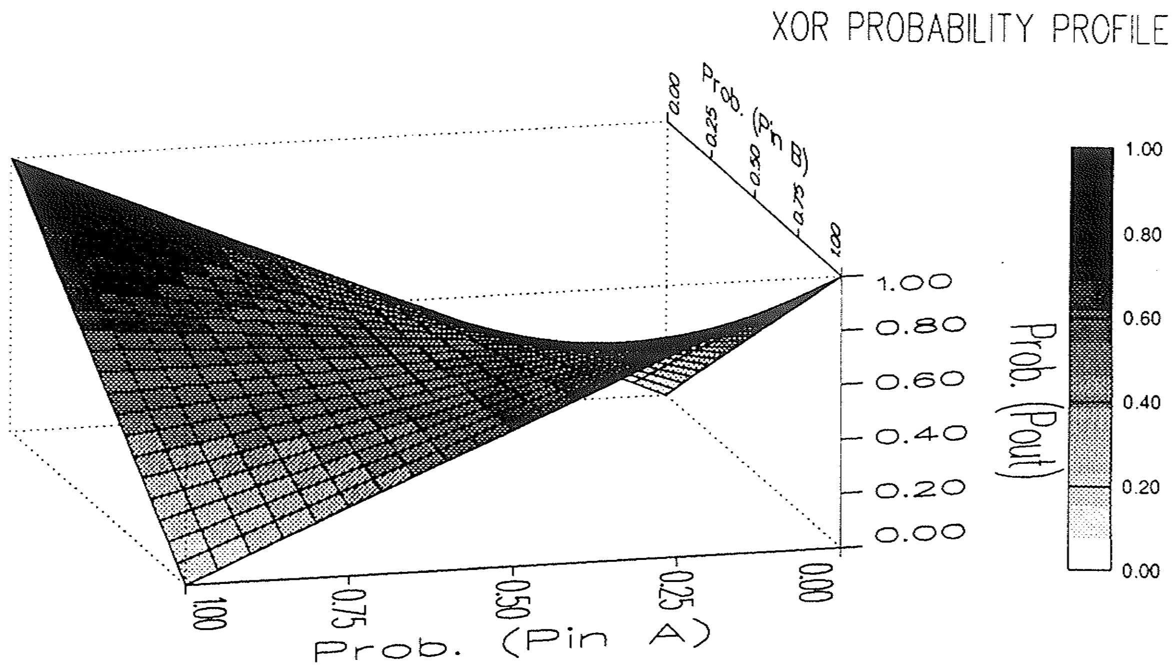

Figure A.2e): *Probability Profile for XOR gate: variable input probabilities, Pin A and Pin B.*

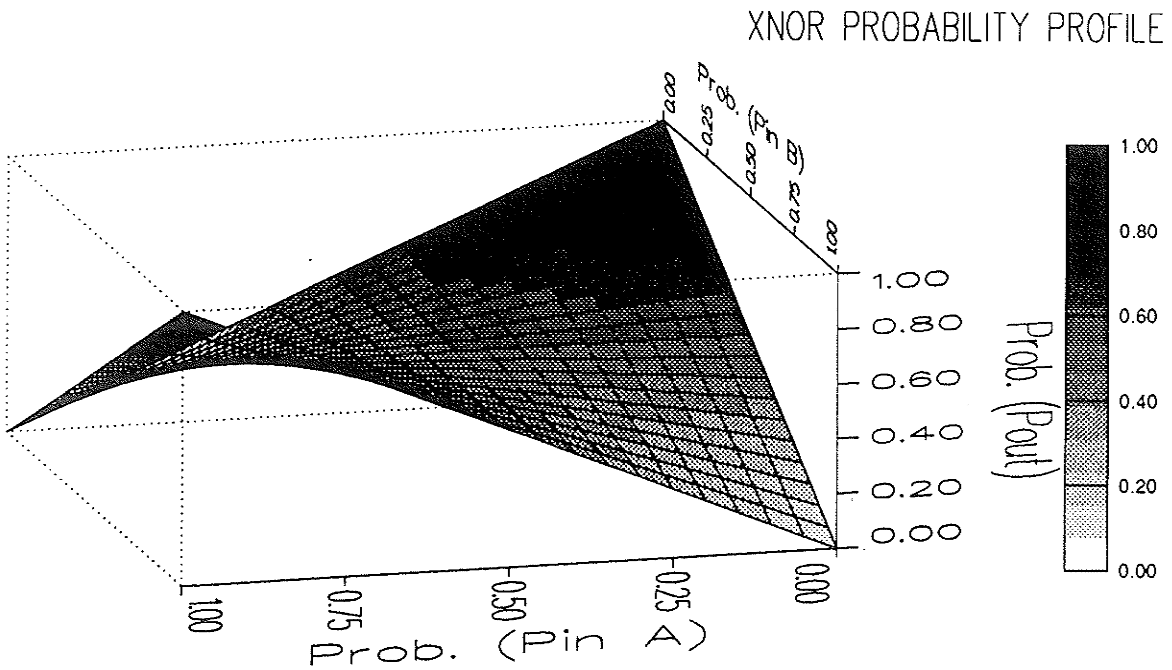

Figure A.2f): *Probability Profile for XNOR gate: variable input probabilities, Pin A and Pin B.*

Appendix B
Statistical Estimators

B.1 Estimation of Independent Random Variables

Consider a sequence of mutually Independent Identically Distributed (IID) random variables, $<x_i>$, with finite mean, μ, and variance, σ^2. For this ensemble, the sample mean

$$\overline{X}_S = \frac{1}{N} \sum_{i=0}^{N-1} x_i \tag{B.1}$$

is an unbiased estimator of μ since $E[\overline{X}_S] = \mu$. For the same ensemble, the sample variance

$$\sigma^2{}_S = \frac{1}{N-1} \sum_{i=0}^{N-1} \left(x_i - \overline{X}_S \right)^2 \tag{B.2}$$

is an unbiased estimator of σ^2 because $E[\sigma^2{}_S] = \sigma^2$. An interesting statistic which is used to measure the accuracy of $\overline{X}_S$ as an estimator of μ, is the variance of the sample mean, $\sigma^2[\overline{X}_S]$ [Law82]. From Theorem B.1, it is obvious that for asymptotically large N, $\overline{X}_S$ should approach μ.

Theorem B.1:

If $\overline{X}_S$ is the sample mean of N mutually independent identically distributed random variables,

$$\sigma^2[\overline{X}_S] = \frac{\sigma^2}{N}. \tag{B.3}$$

Proof:

From the definition of variance [Trivedi82],

$$\sigma^2\left(\sum_{i=0}^{N-1} k_i x_i\right) = \sum_{i=0}^{N-1} k_i{}^2 \cdot \sigma^2[\overline{X}_i] \tag{B.4}$$

if all x_is are mutually independent of one another. Then,

$$\sigma^2[\overline{X}_S] = \sigma^2\left(\sum_{i=0}^{N-1} \frac{x_i}{N}\right)$$

$$= \sum_{i=0}^{N-1}\left(\frac{1}{N}\right)^2 \cdot \sigma^2[x_i]. \tag{B.5}$$

Finally, using the fact that all x_is are mutually independent and identically distributed (IID) of one another,

$$\sigma^2[\overline{X}_S] = \left(\frac{1}{N}\right)^2 \cdot N \cdot \sigma^2[x_i]$$

$$= \frac{\sigma^2}{N} . \tag{B.6}$$

It is also possible to find an unbiased estimator of $\sigma^2[\overline{X}_S]$ by replacing σ^2 in Theorem B.1 with $\sigma^2{}_S$. Then,

$$\hat{\sigma}^2[\overline{X}_S] = \frac{1}{N(N-1)} \sum_{i=0}^{N-1} \left[x_i - \overline{X}_S \right]^2 , \tag{B.7}$$

where $\hat{\sigma}^2[\overline{X}_S]$ denotes that it is an unbiased estimator of $\sigma^2[\overline{X}_S]$; that is, $E[\hat{\sigma}^2[\overline{X}_S]] = \sigma^2[\overline{X}_S]$. By using the above unbiased statistics, a confidence interval for μ can be established, which would further assist in the assessment of the sampling function.

B.2 Estimation of Dependent Random Variables

As mentioned above, it is not always possible to obtain unbiased statistical estimators. For these circumstances, further statistical understanding is necessary so as to establish statistical validity. Because the above statistics were derived based on the assumption of statistical independence, it is not surprising that the sample estimators were unbiased quantities. In practise, however, since simulated output data is almost always correlated, or mutually dependent, it is important to note what affect this will have on the genuineness of the sample estimators of equations B.1 and B.2.

In order to analyze the affect of dependent random variables representing a typical stochastic process, a new assumption must be made. The assumption of a

covariance stationary stochastic process will suffice in providing a realistic dependency model.

Theorem B.2:

A discrete-time stochastic process, with random variables $<x_i>$, is said to be covariance stationary if

$$\mu_i = \mu, \tag{B.8}$$

$$\sigma^2_i = \sigma^2, \tag{B.9}$$

and

$$C_{i,i+j} = Cov(x_i, x_{i+j}) \tag{B.10}$$

is independent of i for $i, j \in \{0, 1, \ldots, N-1\}$, where N represents the number of random variables [Law82].

From Theorem B.2, the sample mean remains an unbiased estimator of μ. But, unfortunately, the same cannot be said about the sample variance. Because $E[\sigma^2_S] \neq \sigma^2$ for the dependent case, it is not an unbiased estimator. Instead,

$$E[\sigma^2_S] = \sigma^2 \cdot \left[1 - 2 \sum_{j=0}^{N-1} \frac{\left(1 - j/N\right) \rho_j}{N-1} \right], \tag{B.11}$$

where

$$\rho_j = \frac{C_{i,i+j}}{\sqrt{\sigma^2{}_i \sigma^2{}_{i+j}}} = \frac{C_j}{\sigma^2} \tag{B.12}$$

is the correlation coefficient of the dependent process [Law82]. If ρ_j is positively correlated, $E[\sigma^2{}_S] < \sigma^2$; thus, in general, if $\sigma^2{}_S$ is used to estimate σ^2, invalidation in the overall statistical analysis could emerge, seriously affecting the reliability of any related statistics. Consequently, this implies that the derivation of a confidence interval for μ is no longer a simple task, since the variance of $\bar{X}_S$ is not equal to σ^2/N. In fact, it can be shown that

$$\sigma^2[\bar{X}_S] = \frac{\sigma^2}{N} + \frac{2}{N}\sum_{j=1}^{N-1}\left[1 - \frac{j}{N}\right]\rho_j \tag{B.13}$$

for a covariance stationary stochastic process. According to Law and Kelton [Law82], if equation B.7 (for the IID process) is used as an estimator for $\sigma^2[\bar{X}_S]$ (for the covariance stationary stochastic process) there will be two factors contributing to errors. The first is the bias of using $\sigma^2{}_S$ as an estimator of σ^2, or

$$-\frac{2}{N-1}\sum_{j=0}^{N-1}\left[1 - j/N\right]\rho_j\,. \tag{B.14}$$

If equations B.11 and B.13 are combined,

$$E[\frac{\sigma^2_S}{N}] = \frac{\left[N/D(N)\right] - 1}{N-1} \cdot \sigma^2[\overline{X}_S], \tag{B.15}$$

where

$$D(N) = \left[\frac{1}{N} + \frac{2}{N}\sum_{j=1}^{N-1}\left[1 - j/N\right]\rho_j\right]. \tag{B.16}$$

Thus, the second error in the estimator will occur as a result of the factor

$$\frac{\left[N/D(N)\right] - 1}{N-1}. \tag{B.17}$$

B.3 Method of Independent Replications

It is statistically beneficial to have a test confidence interval for μ predetermined so that we can test $\overline{X}_S$ for some arbitrary stochastic process. To do this requires some assumptions, approximations, and the utilization of the *central limit theorem* (see [Law82]).

For a sequence of IID random variables $<x_i>$, with mean μ and variance σ^2, it has been shown, by the central limit theorem, that for sufficiently large N the t-distribution function

$$t^{(N)} = \frac{\overline{X}_S - \mu}{\sqrt{\sigma^2_S/N}} \tag{B.18}$$

closely approximates the standard normal distribution function. Therefore, a confidence interval for μ in the $100(1-\alpha)$ percentile is given by

$$\overline{X}_S \pm Z_{\alpha/2}\sqrt{\sigma^2{}_S/N}\ . \tag{B.19}$$

In a similar fashion, for a covariance stationary stochastic process (one which assumes a specific kind of interdependence between its random variables), a t-distribution function can be written, with some general assumptions, as [Trivedi82]

$$t^{(N)} = \frac{\overline{X}_S - \mu}{\sigma \cdot \sqrt{C(\rho_j)/N}}\ , \tag{B.20}$$

where

$$C(\rho_j) = 1 + 2\sum_{j=0}^{\infty} \frac{\rho_j}{\sigma^2}\ . \tag{B.21}$$

Once again, a $100(1-\alpha)$ percentile confidence interval for μ can be approximated by the standard normal distribution function for large N; thus,

$$\overline{X}_S \pm Z_{\alpha/2}\sqrt{C(\rho_j)\sigma^2{}_S/N}\ . \tag{B.22}$$

Trivedi has noted that the confidence interval given by equation B.22 is difficult to estimate because of the problems encountered in deriving estimates for $C(\rho_j)$ and $\sigma^2{}_S$. They suggest a method of determining better estimators for mutually dependent

processes, using the method of *independent replications*.

By replicating an experiment containing N observations M times (each with a different seed), the results of the experiment will be independent even though each individual event is knowingly dependent. The connotation of such a statement is impressive: it means that reliable measures of statistical quantities are possible, even when different kinds of dependencies are involved. This method is undoubtedly more suited for determining a confidence interval for any stochastic process, albeit, at the added expense of an increase in the amount of experimentation conducted.

For some arbitrary stochastic process, let $<x_i>$ denote a sequence of random variables of length N, and $\overline{X}_S(j)$ and $\sigma^2{}_S(j)$ the sample mean and variance, respectively, in the j-th experiment. Then, we may write

$$\overline{X}_S(j) = \frac{1}{N}\sum_{i=0}^{N-1} x_j(j), \tag{B.23}$$

and

$$\sigma^2{}_S(j) = \frac{1}{N-1}\sum_{i=0}^{N-1}\left[x_i(j) - \overline{X}_S(j)\right]. \tag{B.24}$$

With the results of j independent sample means, an unbiased estimator for the ensemble mean can be determined by

$$\overline{X}_S = \frac{1}{M}\sum_{j=0}^{M-1}\overline{X}_S(j)$$

$$= \frac{1}{M \cdot N} \sum_{j=0}^{M-1} \sum_{i=0}^{N-1} x_i(j), \tag{B.25}$$

whereas an estimate of the variance of the sample means can be estimated by

$$\hat{\sigma}^2_S = \frac{1}{M-1} \sum_{j=0}^{M-1} \left[\overline{X}_S(j) - \overline{X}_S \right]^2$$

$$= \frac{1}{M-1} \sum_{j=0}^{M-1} \overline{X}^2_S(j) - \frac{M}{M-1} \overline{X}^2_S . \tag{B.26}$$

Consequently, if both M and N are large, the t-distribution

$$t^{(M,N)} = \frac{\overline{X}_S - \mu}{\sqrt{\hat{\sigma}^2_S / M}} \tag{B.27}$$

can approximate the standard normal distribution function. This implies that a confidence interval for M at the $100(1-\alpha)$ percentile, can be denoted as

$$\overline{X}_S \pm Z_{\alpha/2} \sqrt{\hat{\sigma}^2_S / M} . \tag{B.28}$$

In retrospect, one can appreciate that only approximations can, at best, be made when constructing confidence intervals for μ, whether or not the random variables are mutually independent. Even so, it is purposeful to provide some objective insight into understanding the hazards of using biased sample estimators for statistically evaluating stochastic processes. In the analysis presented, as appreciated, much of the statistics are affected by biased estimates, albeit, ever so slightly. They include: density and

average density; probability mass function; and, auto and cross-correlation coefficients. Even though biased estimates are ever present, they do not deter or alter any of the said conclusions. This is because the sample ensembles are sufficiently large so as to reveal the evolutionary, or global, behaviours of the statistical evolution mechanisms, which is all that is considered as important for investigation. In any event, the subtle differences which do occur are percieved as having no drastic impact on the validity of those residing BIST test quality measures. However, as appreciated, they could inflict recognizable anomalies if used in more demanding capacities.

Appendix C

State-Time Evolutions

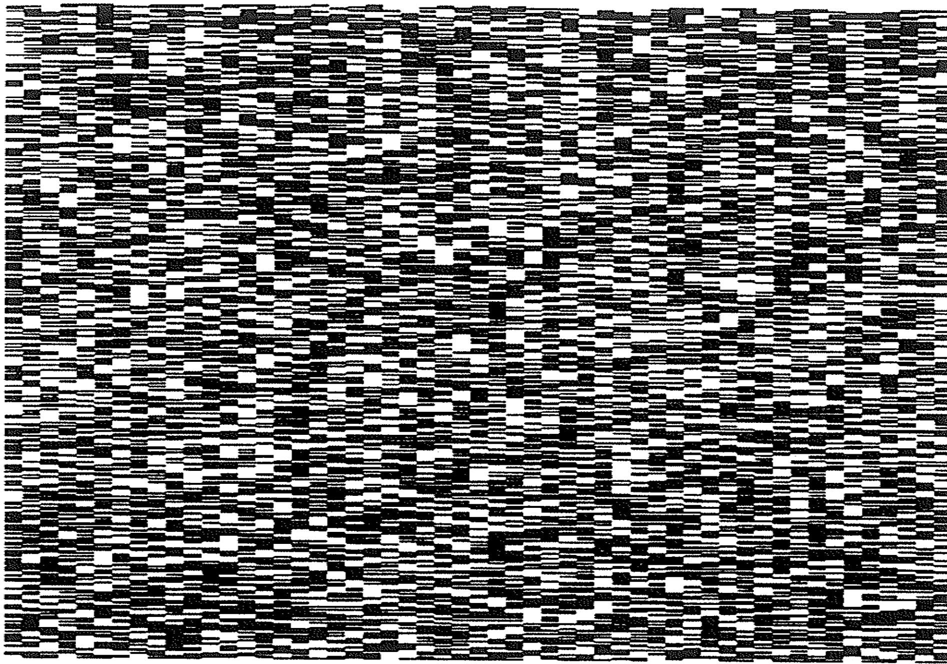

Figure C.1: *Space-Time Evolution of test patterns for NLFSR: W=53; L=400.*

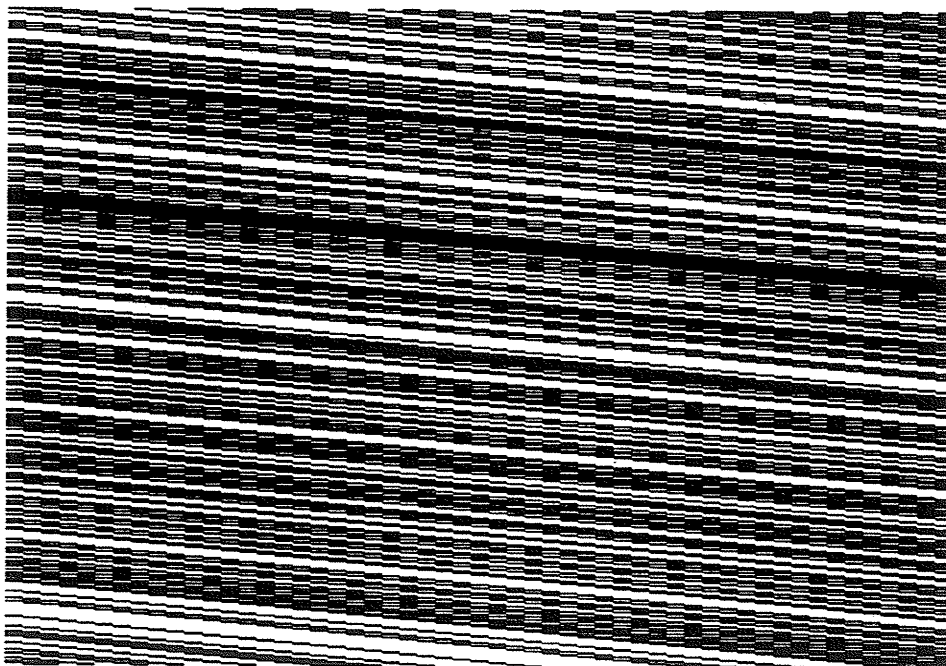

Figure C.2: *Space-Time Evolution of test patterns for LFSR: W=53; L=400.*

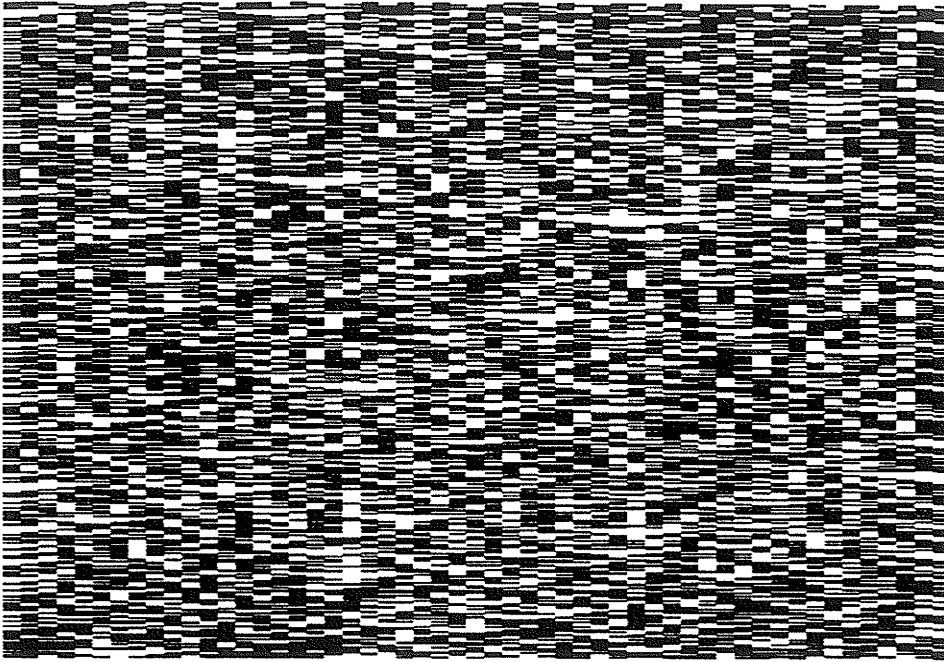

Figure C.3: *Space-Time Evolution of test patterns for Rule 90/150 HCA: W=53; L=400.*

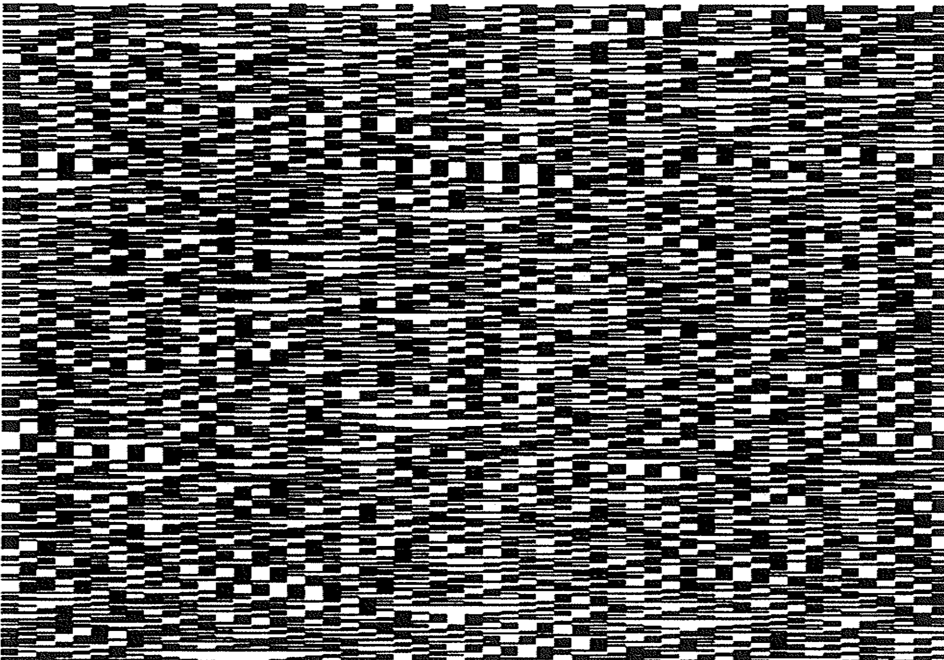

Figure C.4: *Space-Time Evolution of test patterns for Rule 30 CA: W=53; L=400.*

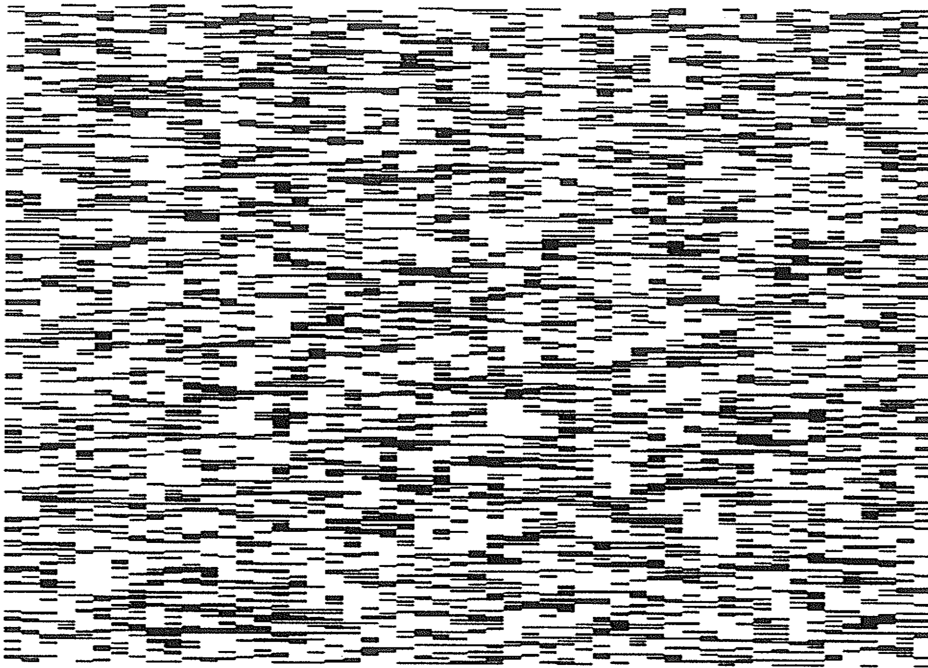

Figure C.5a): *Space-Time Evolution of test patterns for NLFSR weighted to 25% (AND-gate bank): [S]; W=52; L=400.*

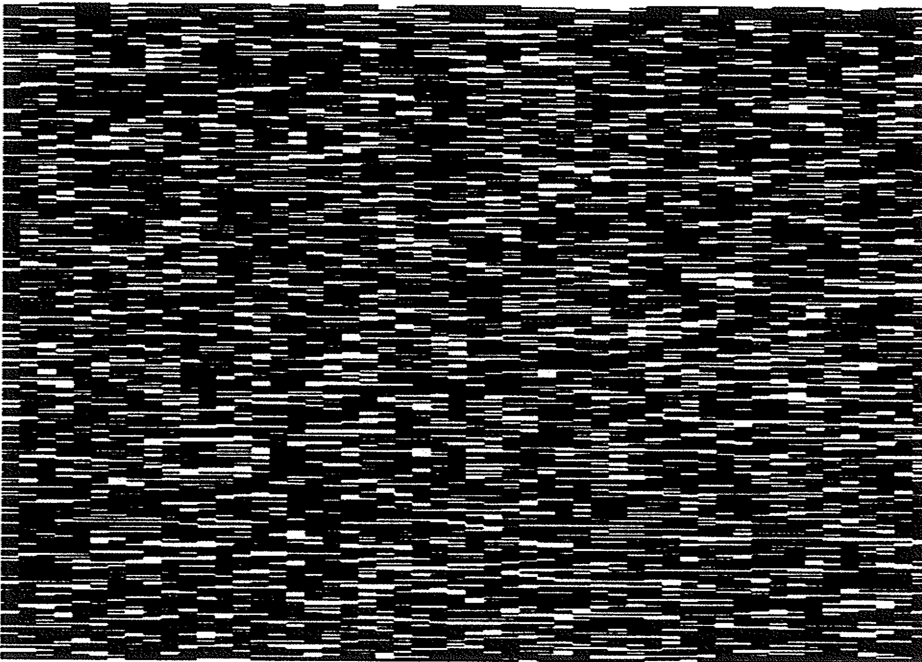

Figure C.5b): *Space-Time Evolution of test patterns for NLFSR weighted to 75% (OR-gate bank): [S]; W=52; L=400.*

Figure C.6a): *Space-Time Evolution of test patterns for LFSR weighted to 25% (AND-gate bank): [S]; W=52; L=400.*

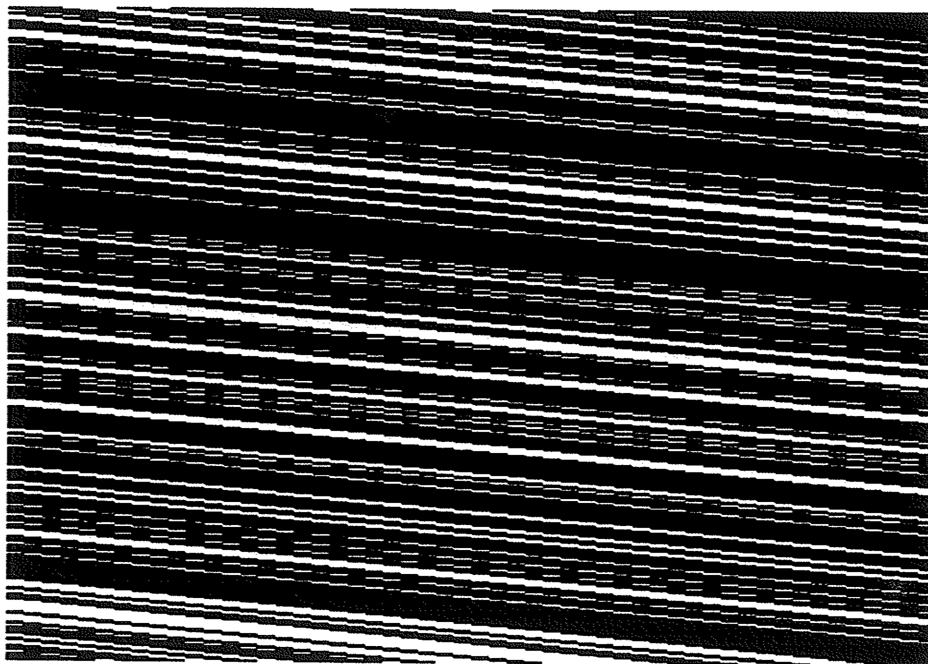

Figure C.6b): *Space-Time Evolution of test patterns for LFSR weighted to 75% (OR-gate bank): [S]; W=52; L=400.*

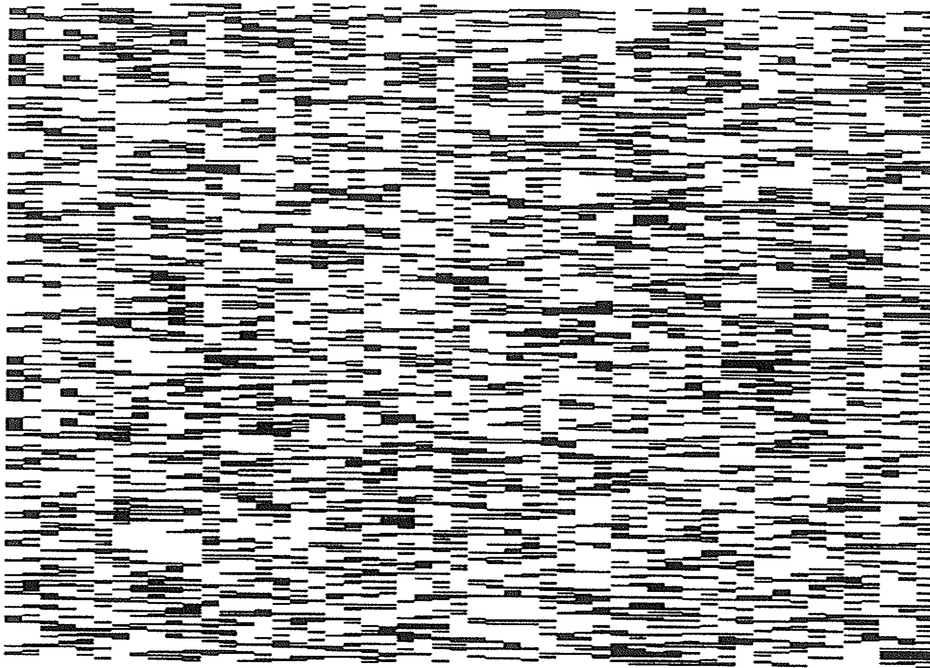

Figure C.7a): *Space-Time Evolution of test patterns for Rule 90/150 HCA weighted to 25% (AND-gate bank): [S]; W=52; L=400.*

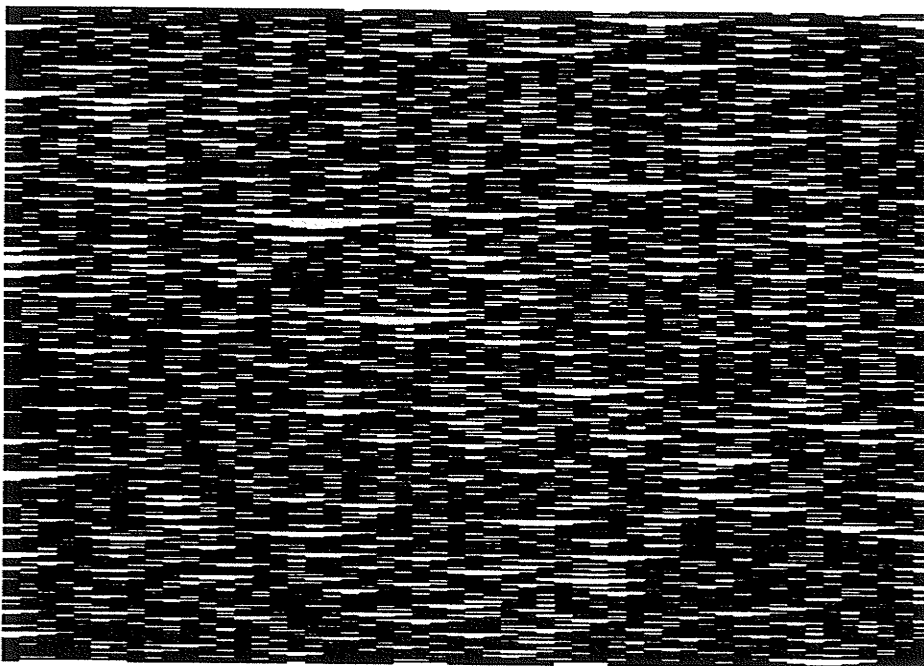

Figure C.7b): *Space-Time Evolution of test patterns for Rule 90/150 HCA weighted to 75% (OR-gate bank): [S]; W=52; L=400.*

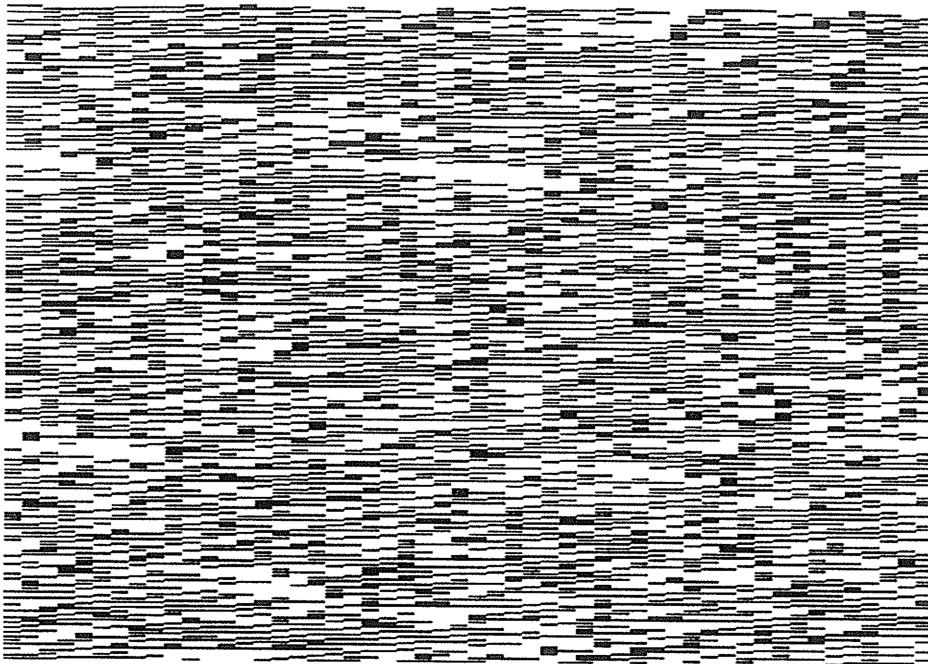

Figure C.8a): *Space-Time Evolution of test patterns for Rule 30 CA weighted to 25% (AND-gate bank): [S]; W=52; L=400.*

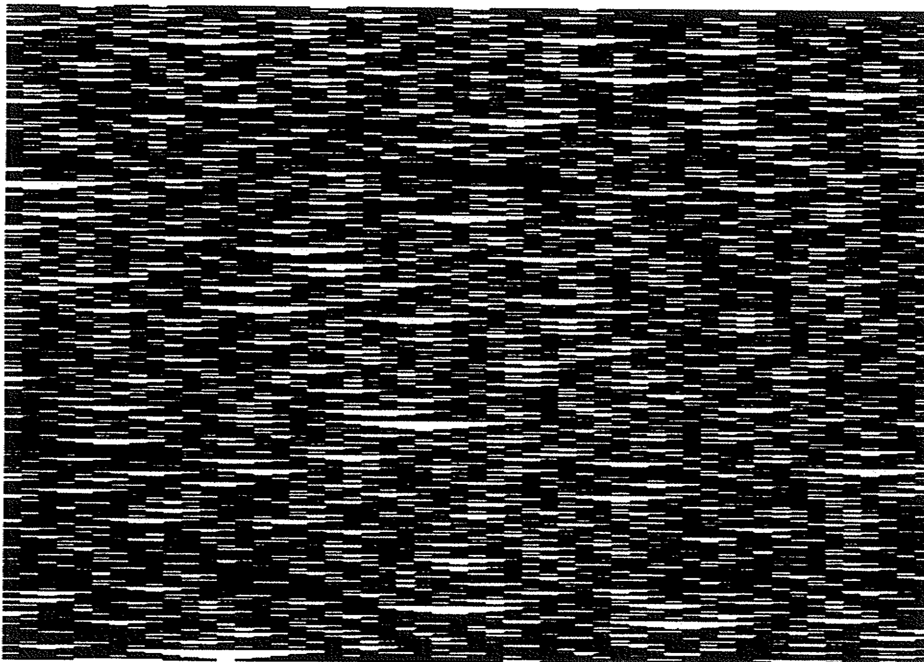

Figure C.8b): *Space-Time Evolution of test patterns for Rule 30 CA weighted to 75% (OR-gate bank): [S]; W=52; L=400.*

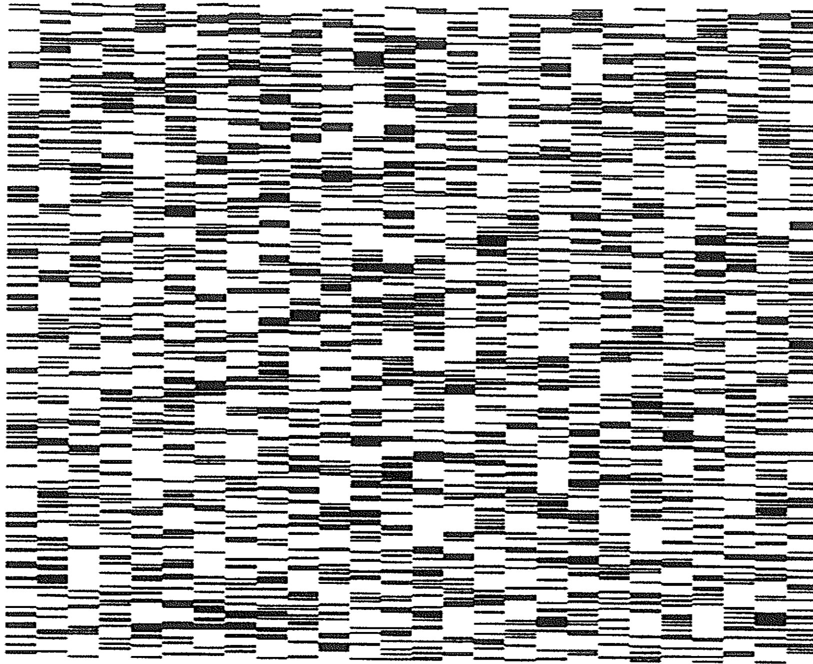

Figure C.9a): *Space-Time Evolution of test patterns for NLFSR weighted to 25% (AND-gate bank): [J=0]; W=26; L=400.*

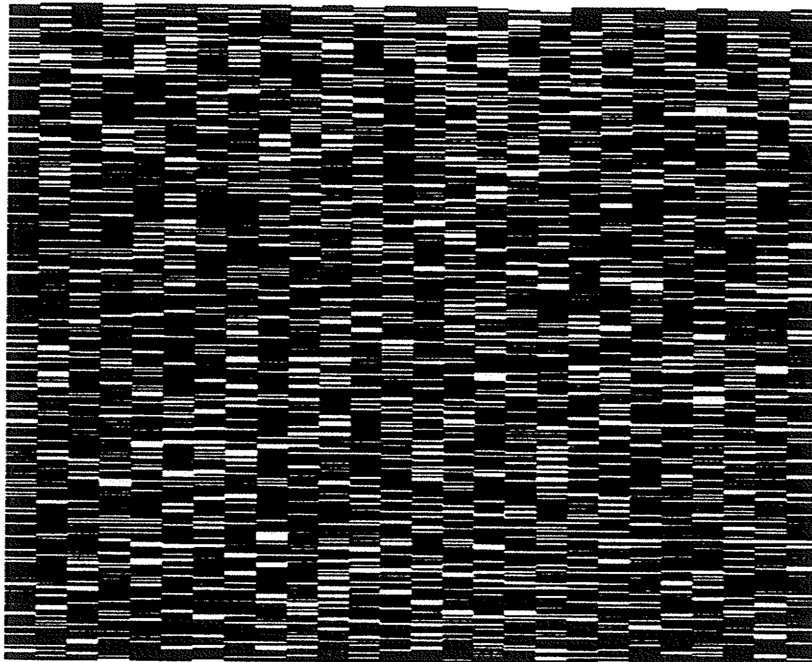

Figure C.9b): *Space-Time Evolution of test patterns for NLFSR weighted to 75% (OR-gate bank): [J=0]; W=26; L=400.*

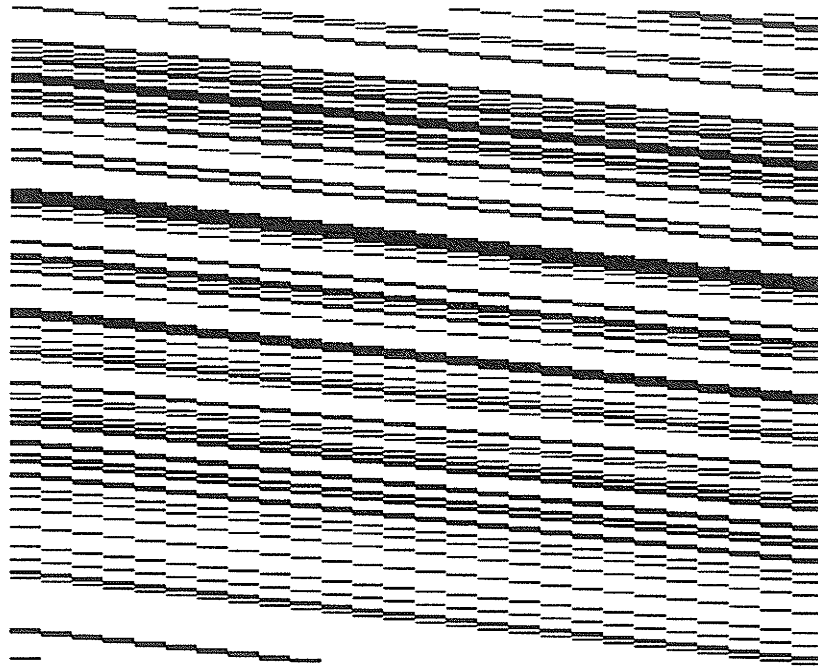

Figure C.10a): *Space-Time Evolution of test patterns for LFSR weighted to 25% (AND-gate bank): [J=0]; W=26; L=400.*

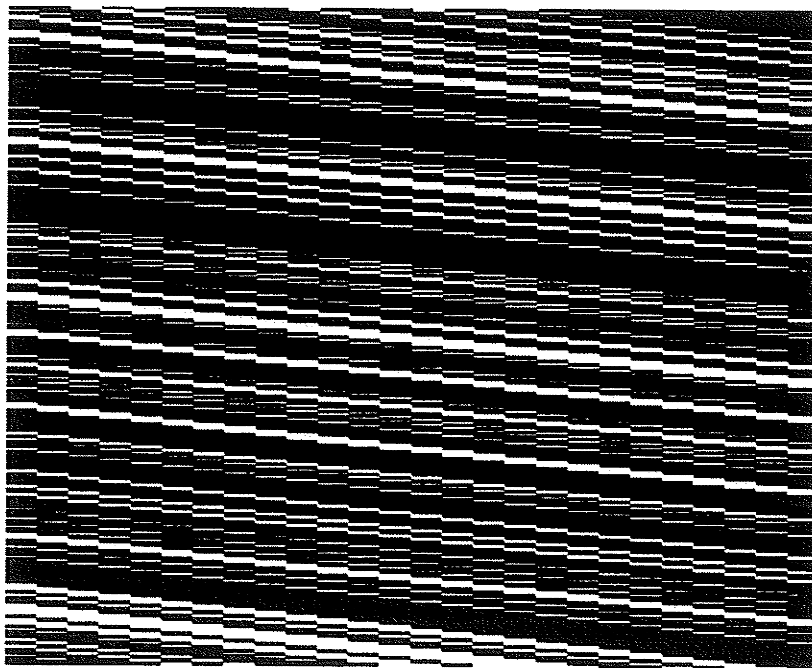

Figure C.10b): *Space-Time Evolution of test patterns for LFSR weighted to 75% (OR-gate bank): [J=0]; W=26; L=400.*

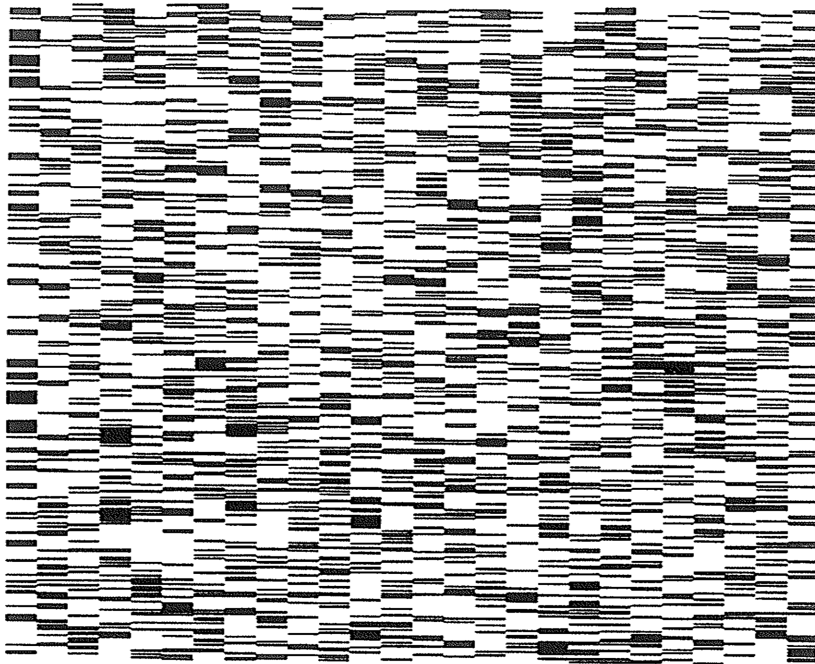

Figure C.11a): *Space-Time Evolution of test patterns for Rule 90/150 HCA weighted to 25% (AND-gate bank): [J=0]; W=26; L=400.*

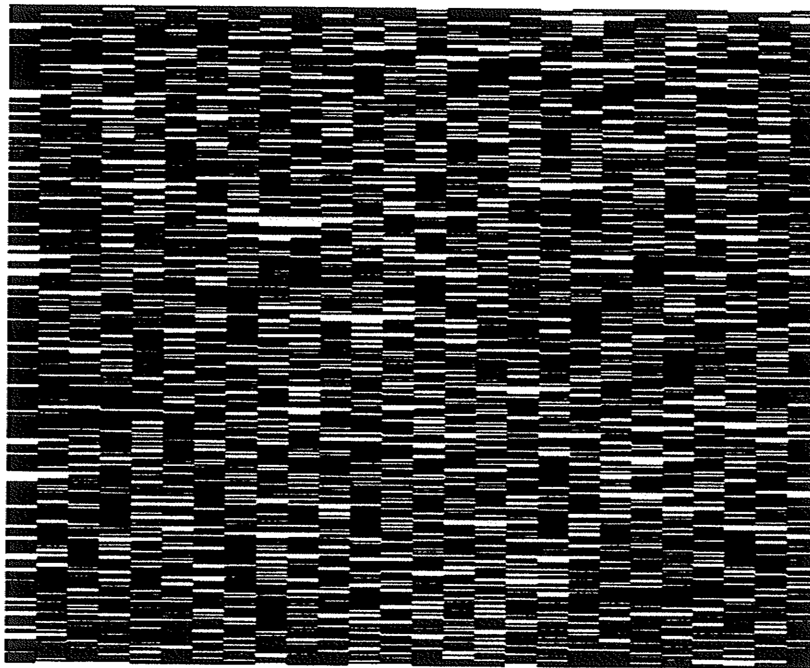

Figure C.11b): *Space-Time Evolution of test patterns for Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=0]; W=26; L=400.*

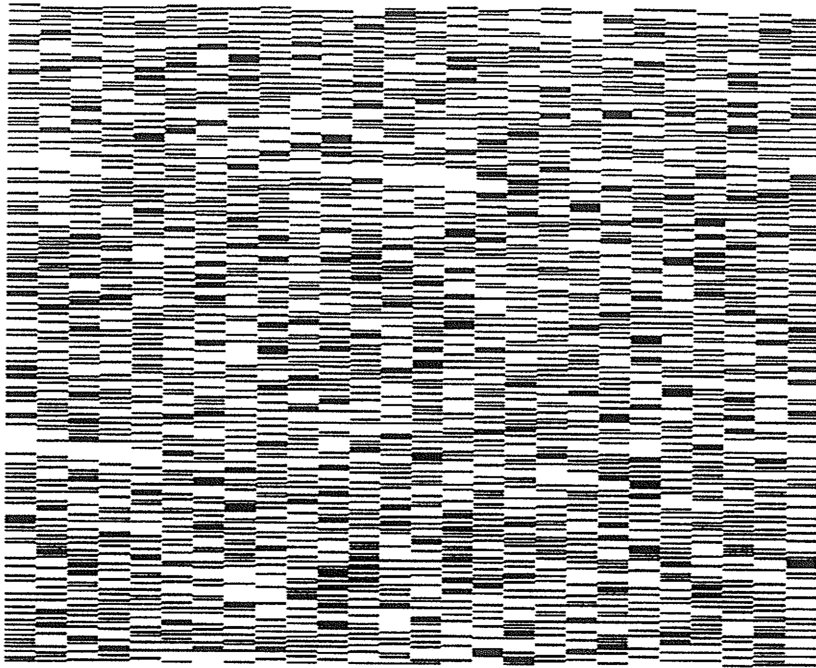

Figure C.12a): *Space-Time Evolution of test patterns for Rule 30 CA weighted to 25% (AND-gate bank): [J=0]; W=26; L=400.*

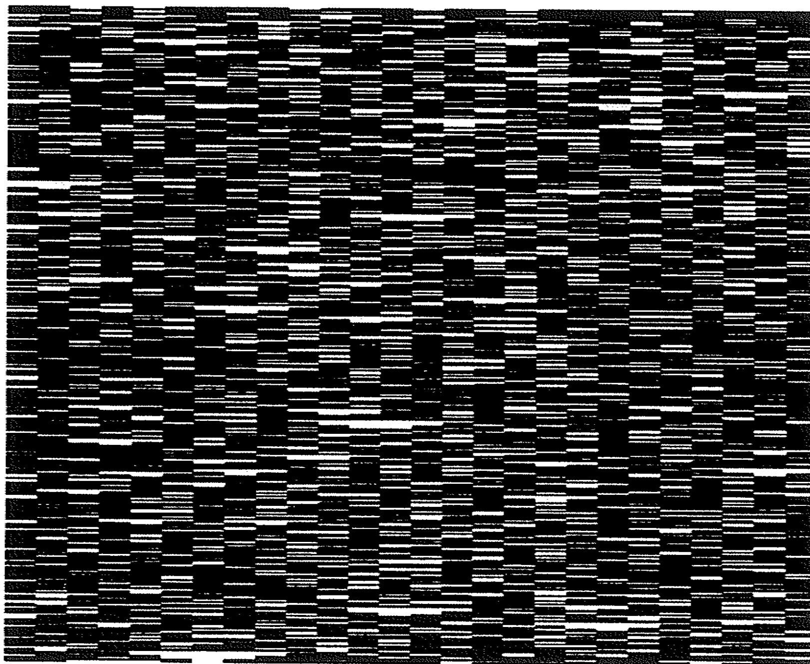

Figure C.12b): *Space-Time Evolution of test patterns for Rule 30 CA weighted to 75% (OR-gate bank): [J=0]; W=26; L=400.*

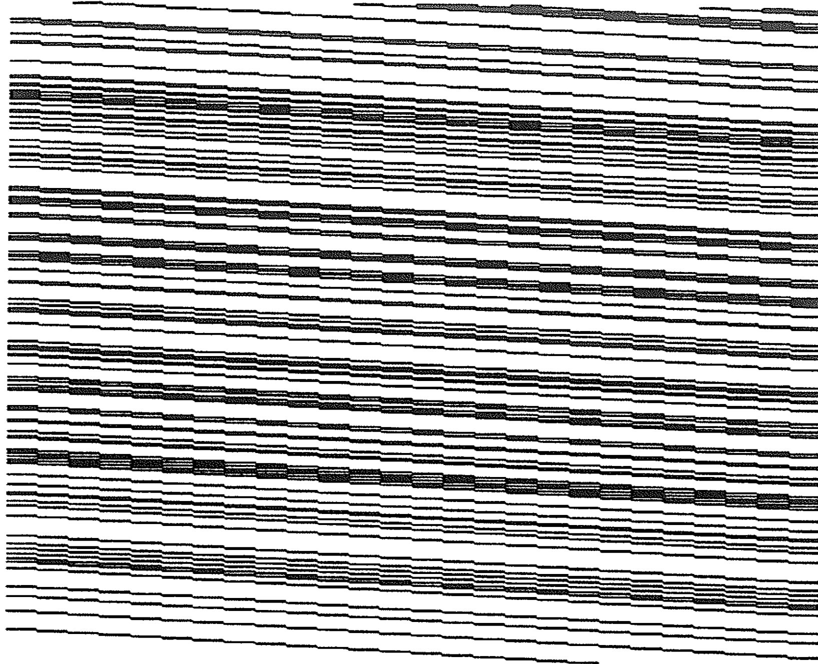

Figure C.13a): *Space-Time Evolution of test patterns for LFSR weighted to 25% (AND-gate bank): [J=25]; W=26; L=400.*

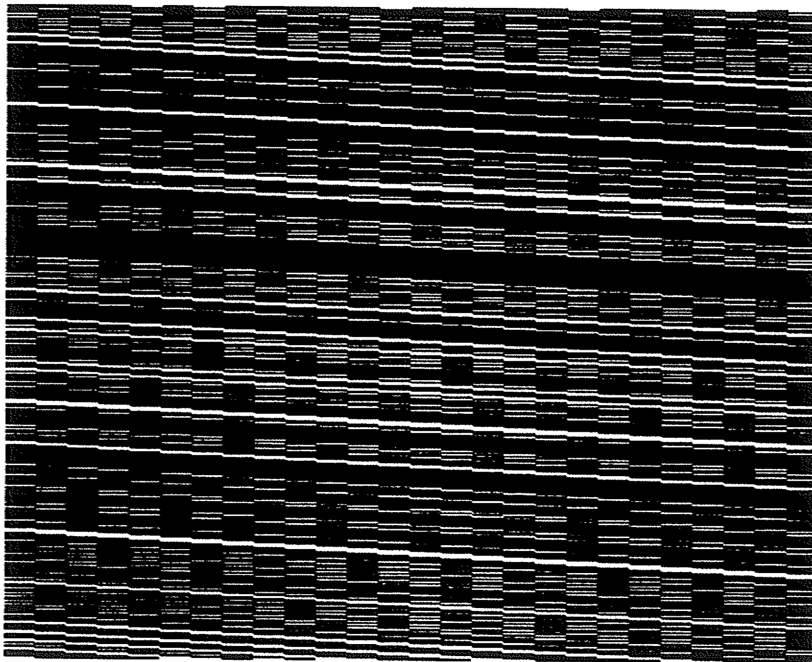

Figure C.13b): *Space-Time Evolution of test patterns for LFSR weighted to 75% (OR-gate bank): [J=25]; W=26; L=400.*

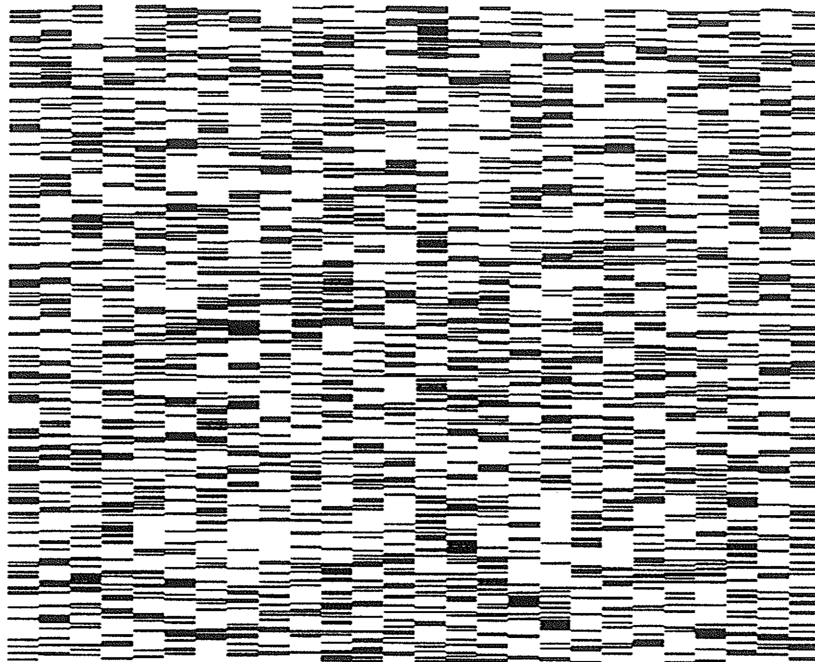

Figure C.14a): *Space-Time Evolution of test patterns for Rule 90/150 HCA weighted to 25% (AND-gate bank): [J=25]; W=26; L=400.*

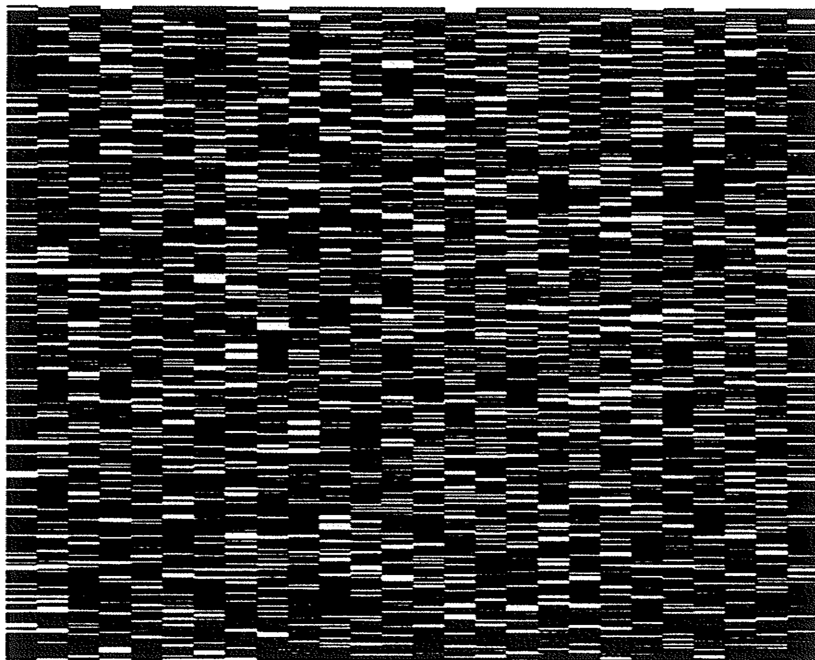

Figure C.14b): *Space-Time Evolution of test patterns for Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=25]; W=26; L=400.*

Appendix D

Density and Average Density Evolutions

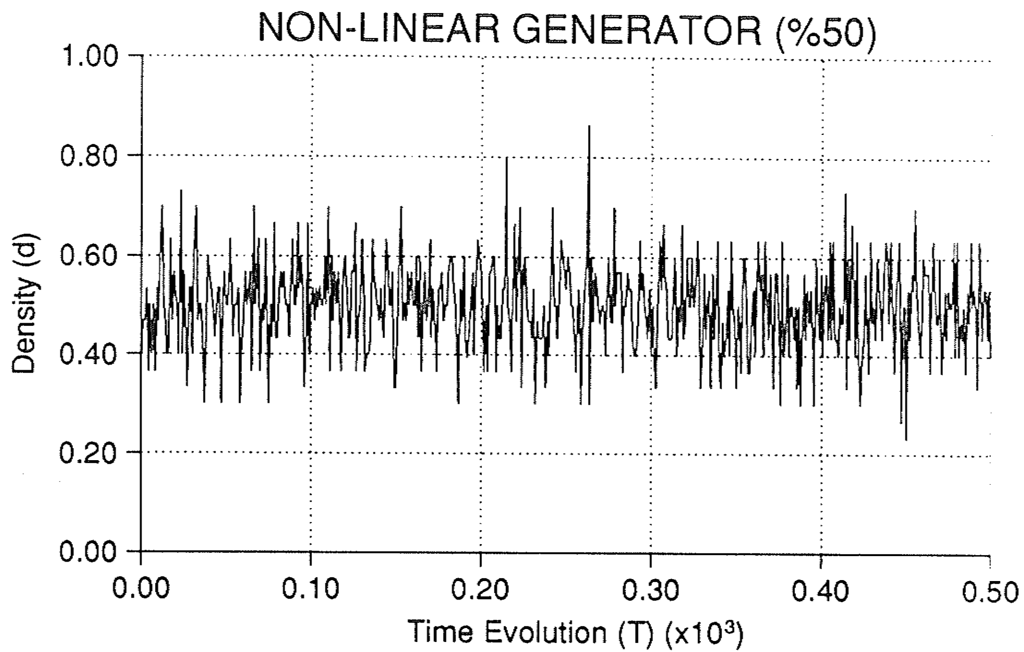

Figure D.1a): *Density Evolution of test patterns for NLFSR: W=30; L=500.*

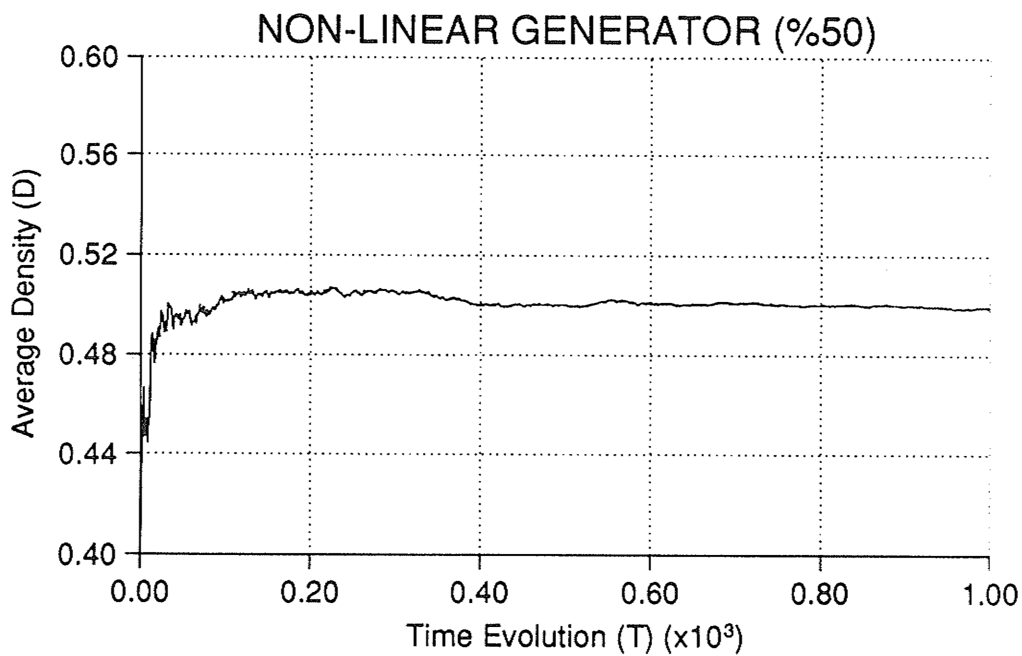

Figure D.1b): *Average Density Evolution of test patterns for NLFSR: W=30; L=1,000.*

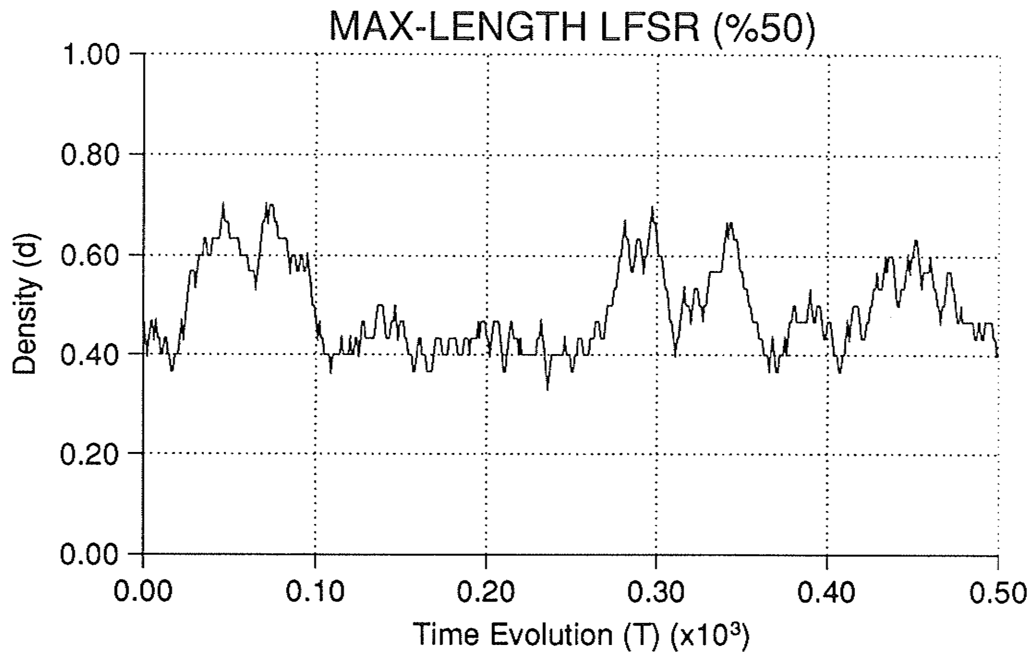

Figure D.2a): *Density Evolution of test patterns for LFSR: W=30; L=500.*

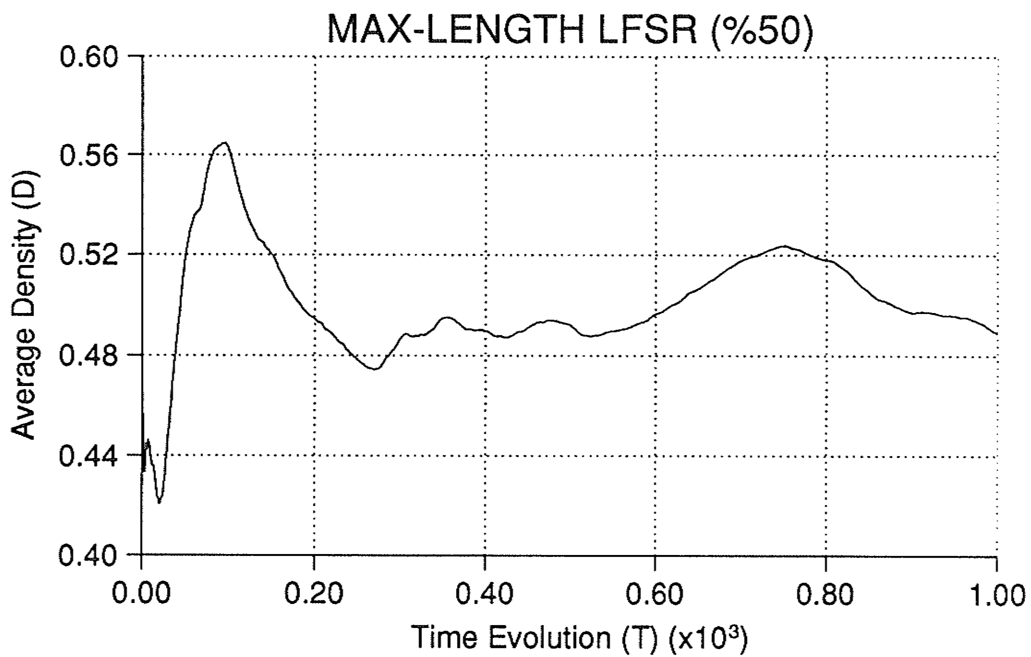

Figure D.2b): *Average Density Evolution of test patterns for LFSR: W=30; L=1,000.*

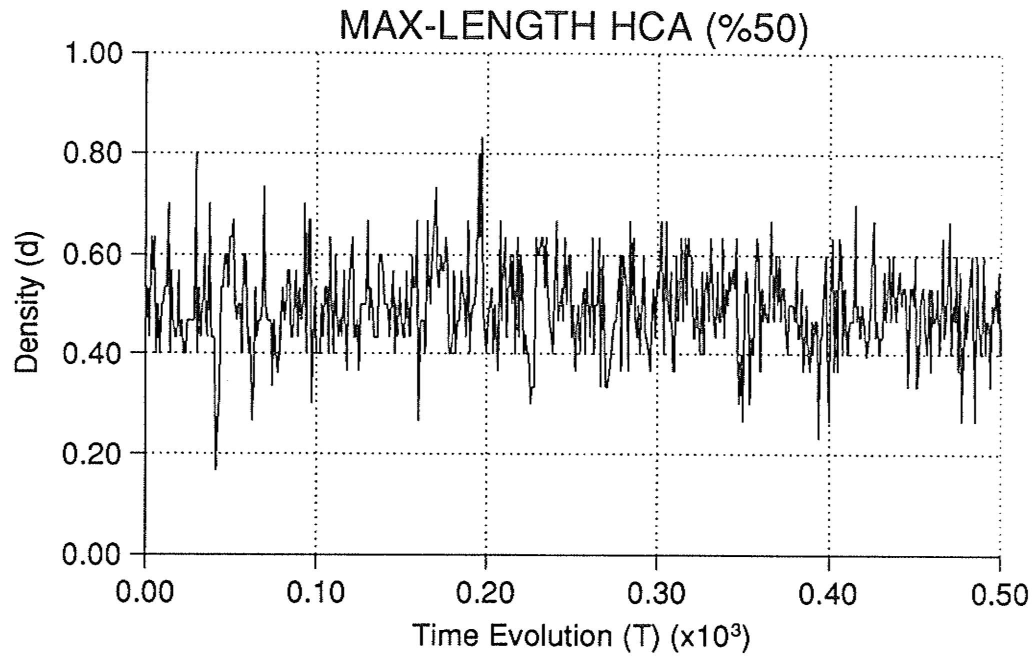

Figure D.3a): *Density Evolution of test patterns for Rule 90/150 HCA: W=30; L=500.*

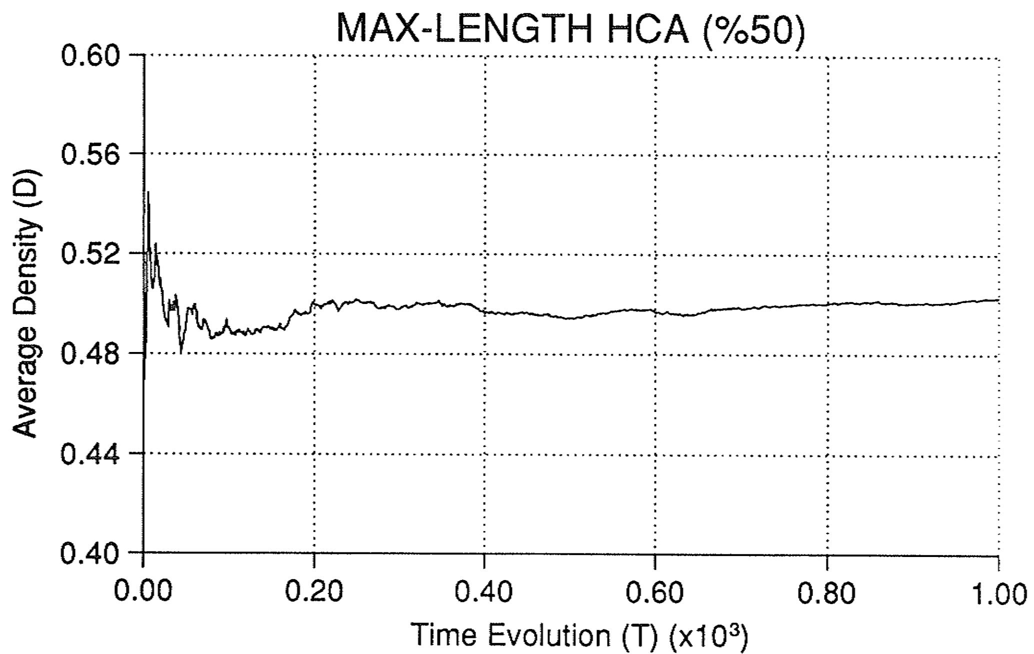

Figure D.3b): *Average Density Evolution of test patterns for Rule 90/150 HCA: W=30; L=1,000.*

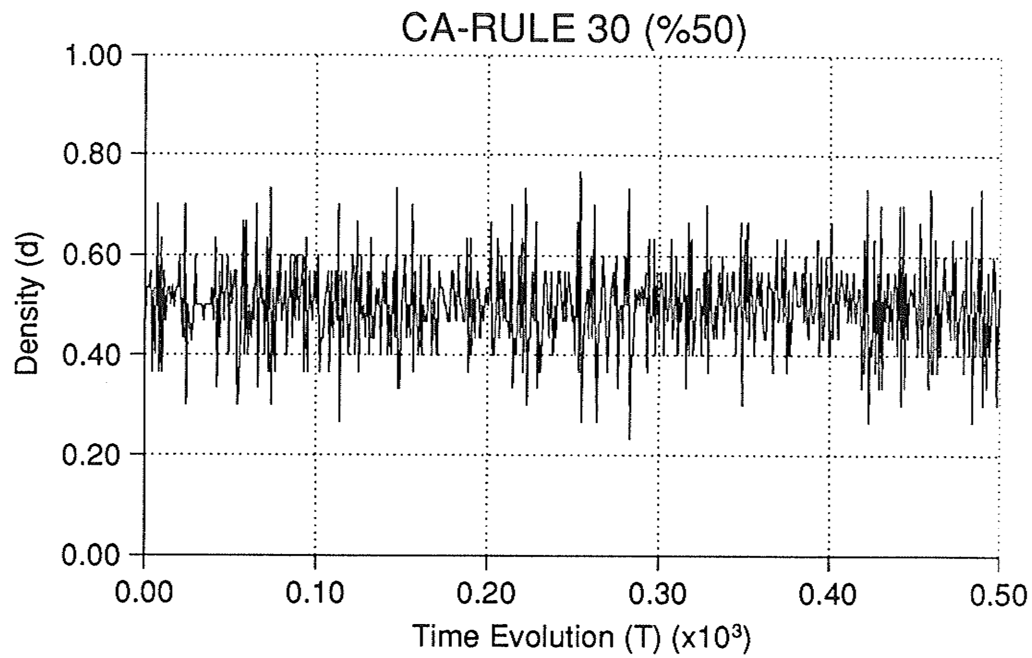

Figure D.4a): *Density Evolution of test patterns for Rule 30 CA: W=30; L=500.*

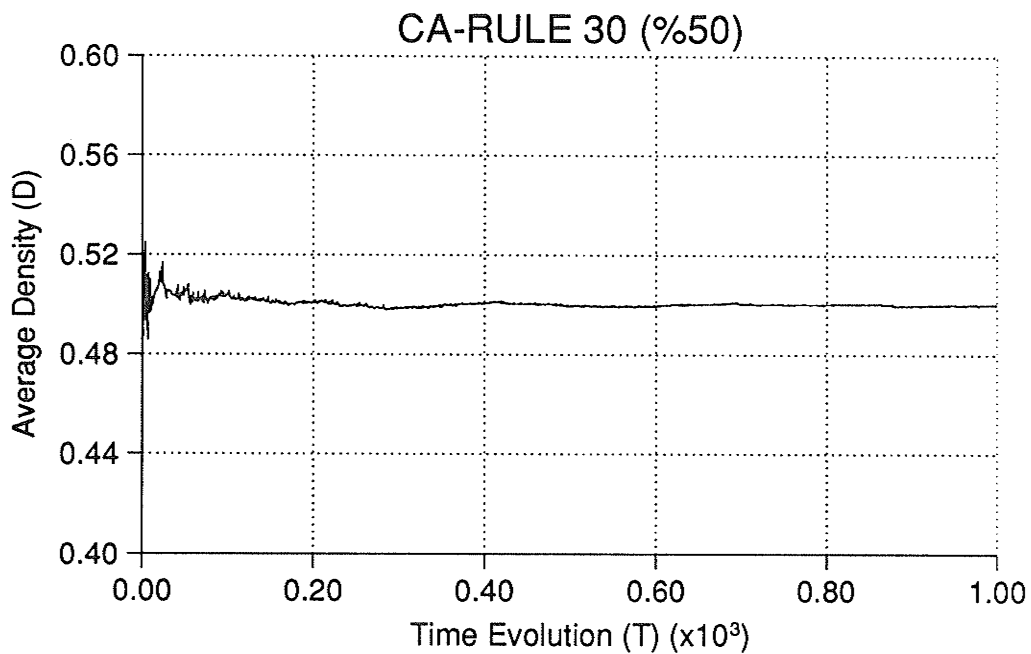

Figure D.4b): *Average Density Evolution of test patterns for Rule 30 CA: W=30; L=1,000.*

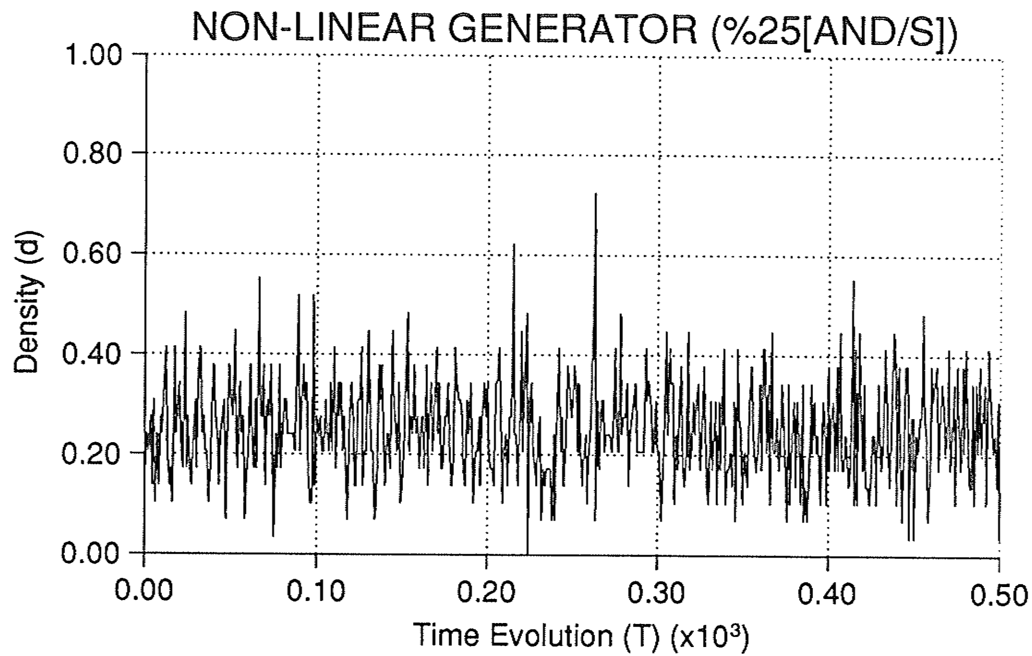

Figure D.5a): *Density Evolution of test patterns for NLFSR weighted to 25% (AND-gate bank): [S]; W=29; L=500.*

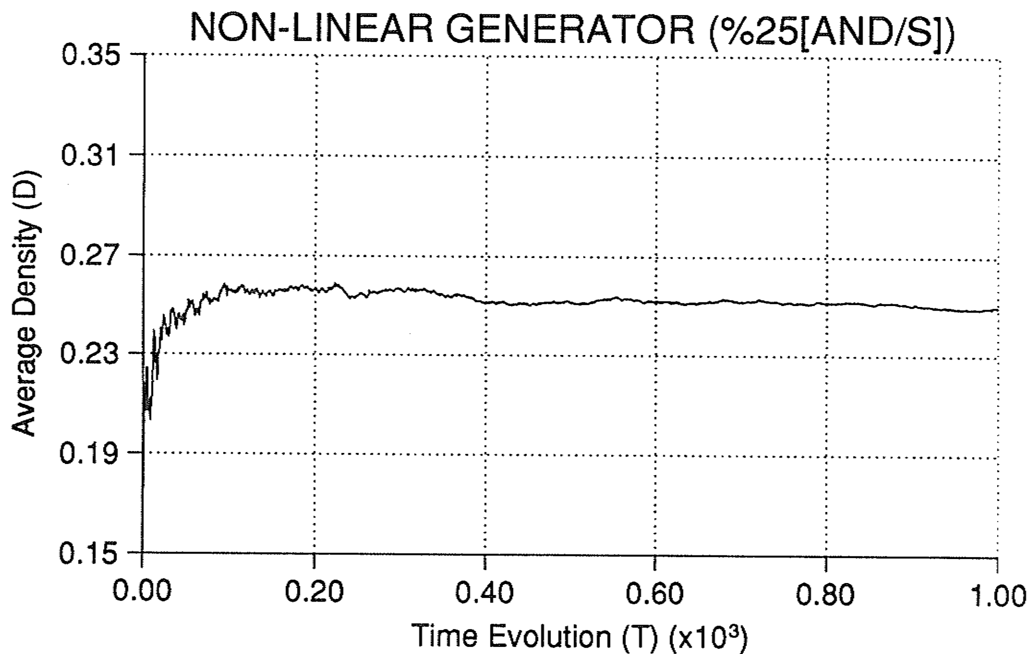

Figure D.5b): *Average Density Evolution of test patterns for NLFSR weighted to 25% (AND-gate bank): [S]; W=29; L=1,000.*

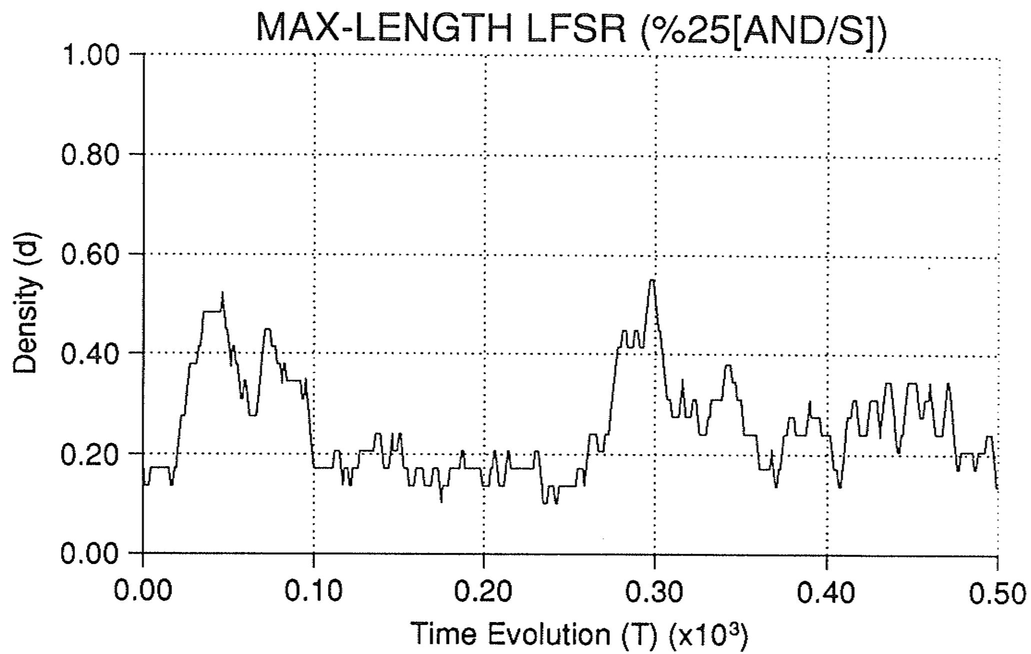

Figure D.6a): *Density Evolution of test patterns for LFSR weighted to 25% (AND-gate bank): [S]; W=29; L=500.*

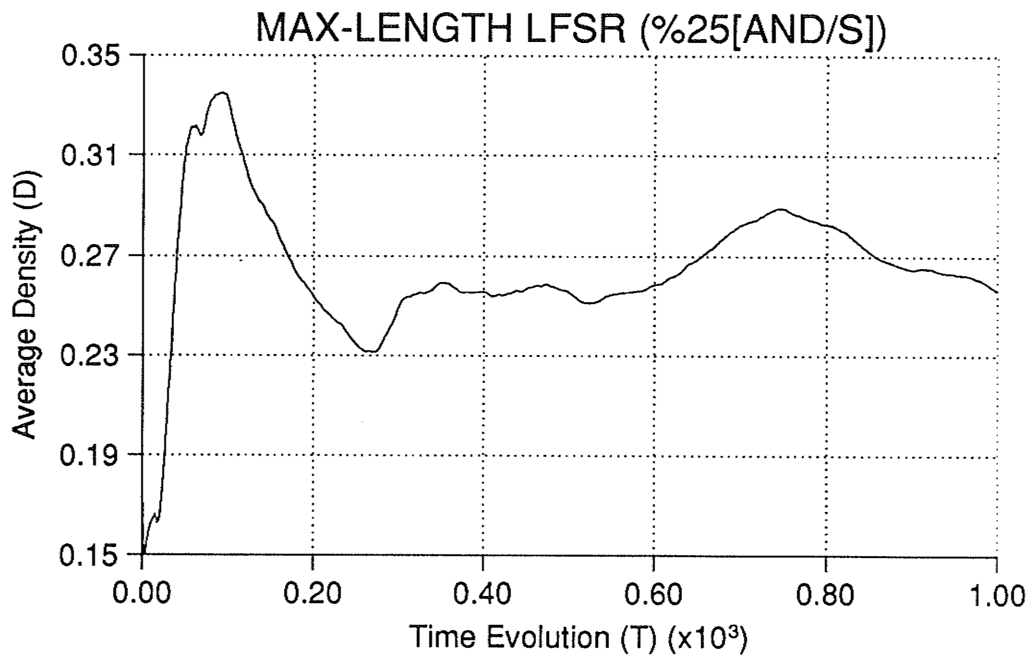

Figure D.6b): *Average Density Evolution of test patterns for LFSR weighted to 25% (AND-gate bank): [S]; W=29; L=1,000.*

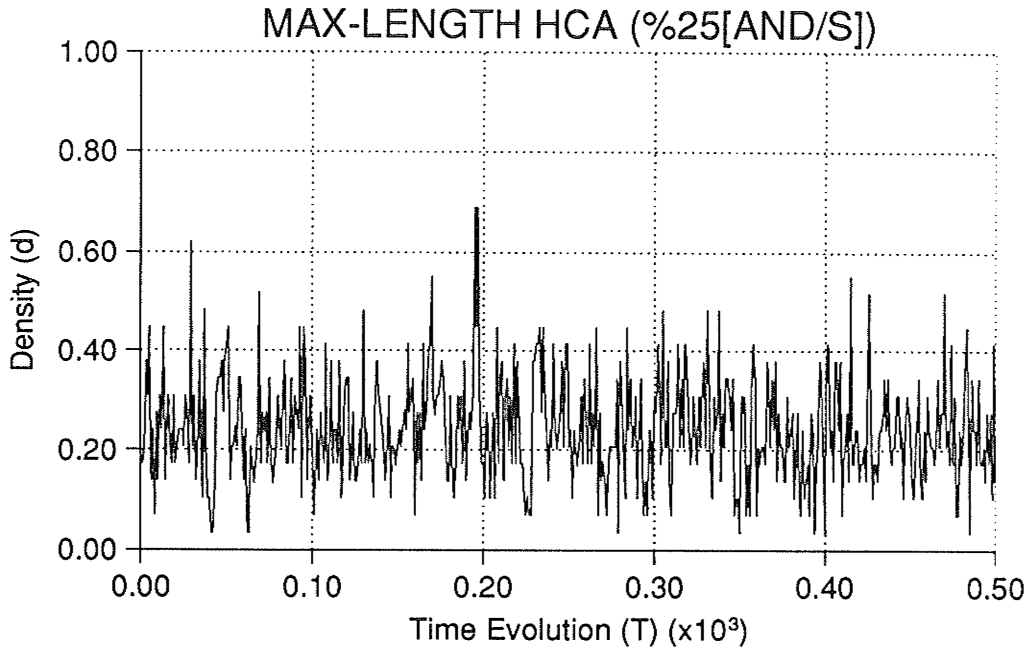

Figure D.7a): *Density Evolution of test patterns for Rule 90/150 HCA weighted to 25% (AND-gate bank): [S]; W=29; L=500.*

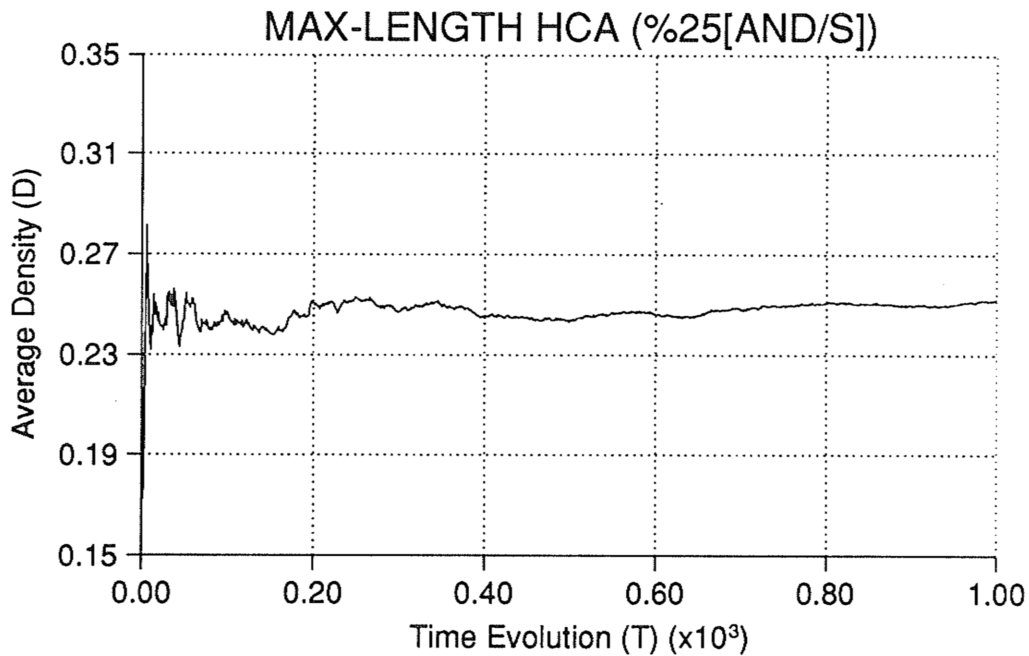

Figure D.7b): *Average Density Evolution of test patterns for Rule 90/150 HCA weighted to 25% (AND-gate bank): [S]; W=29; L=1,000.*

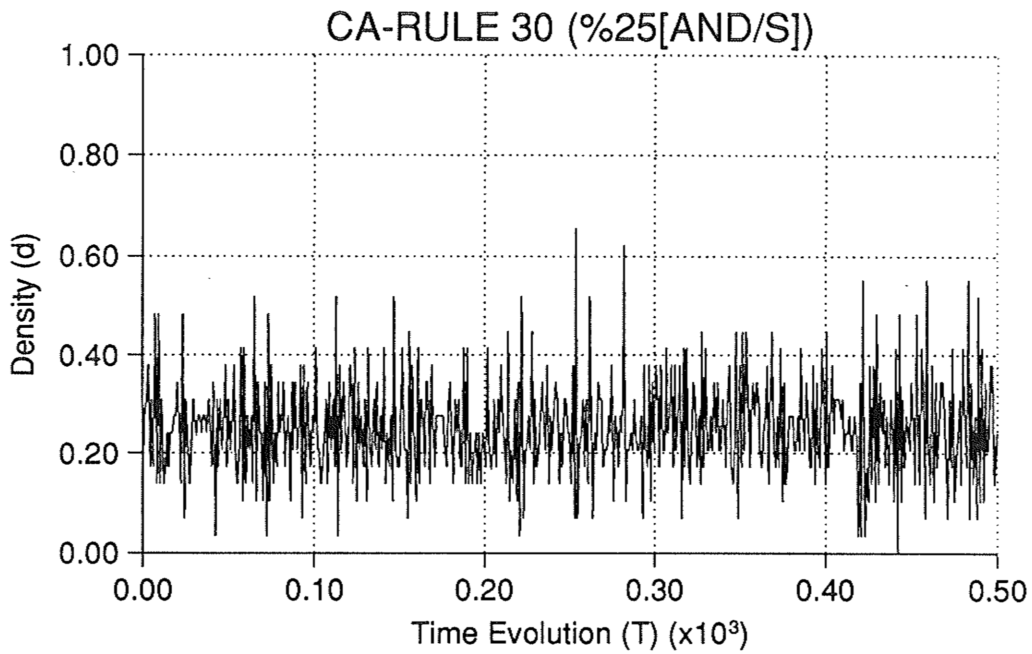

Figure D.8a): *Density Evolution of test patterns for Rule 30 CA: weighted to 25% (AND-gate bank): [S]; W=29; L=500.*

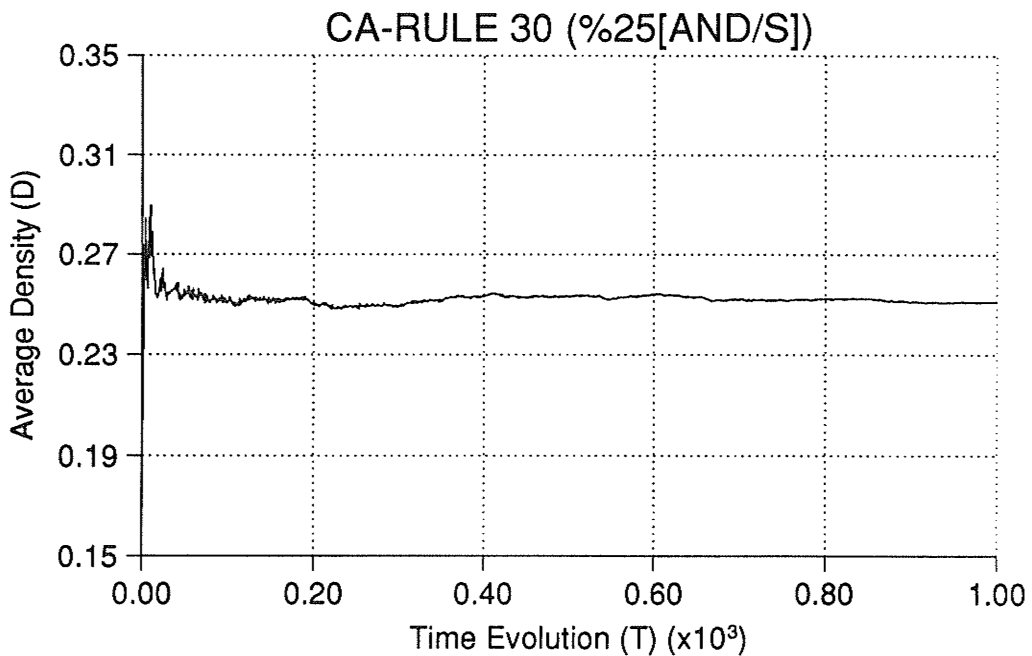

Figure D.8b): *Average Density Evolution of test patterns for Rule 30 CA weighted to 25% (AND-gate bank): [S]; W=29; L=1,000.*

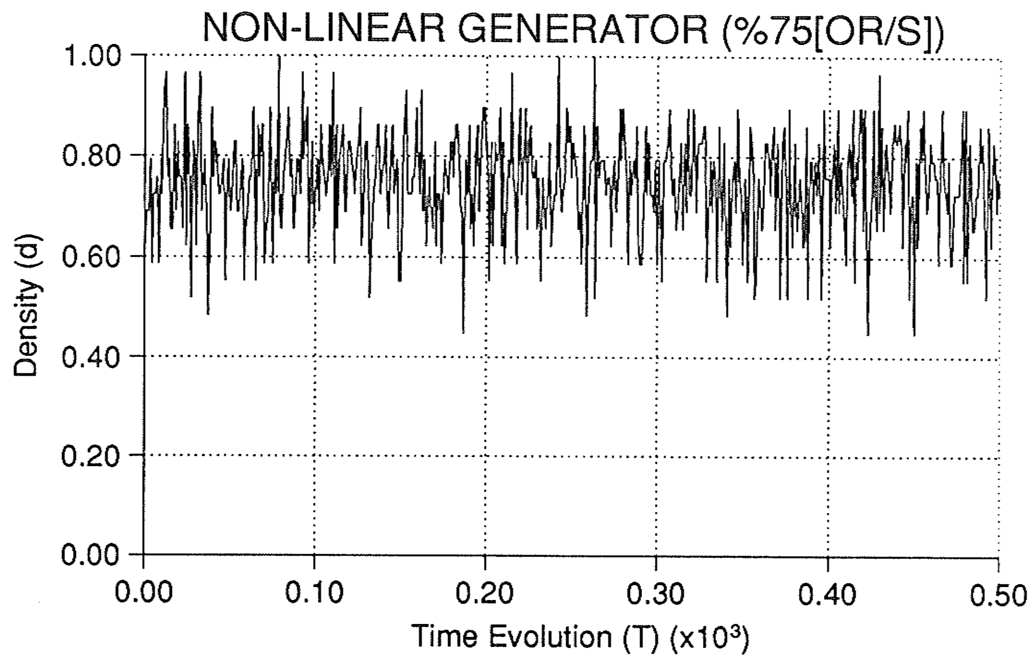

Figure D.9a): *Density Evolution of test patterns for NLFSR weighted to 75% (OR-gate bank): [S]; W=29; L=500.*

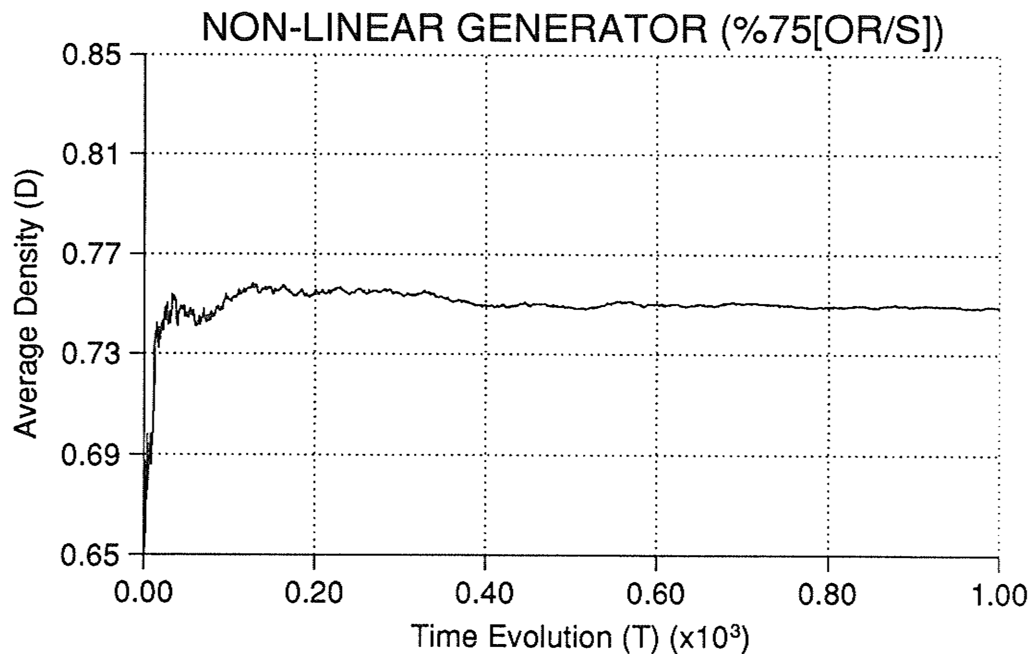

Figure D.9b): *Average Density Evolution of test patterns for NLFSR weighted to 75% (OR-gate bank): [S]; W=29; L=1,000.*

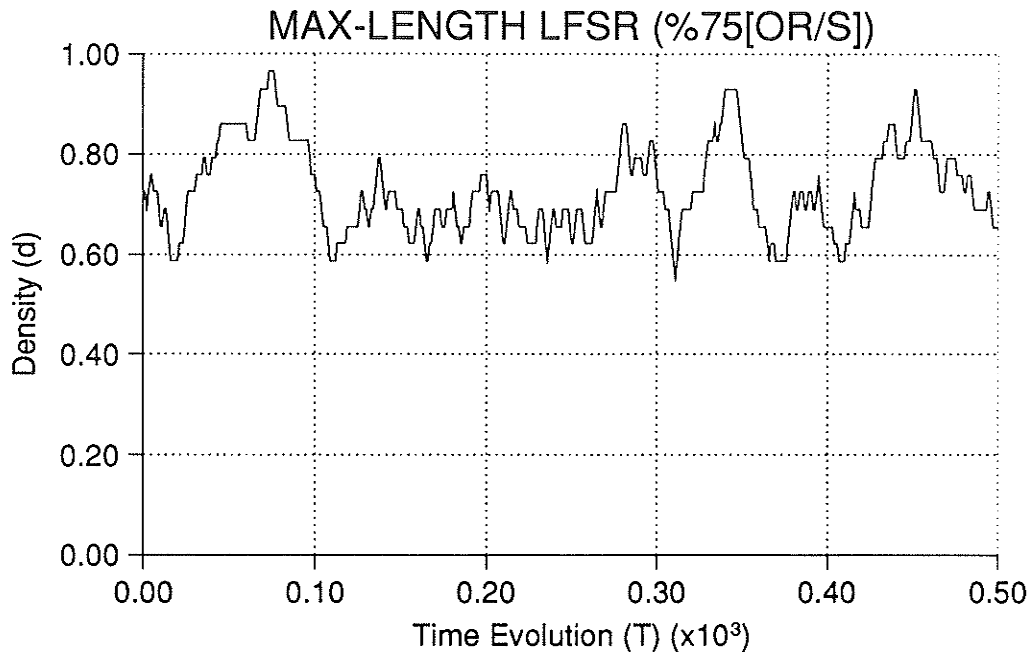

Figure D.10a): *Density Evolution of test patterns for LFSR weighted to 75% (OR-gate bank): [S]; W=29; L=500.*

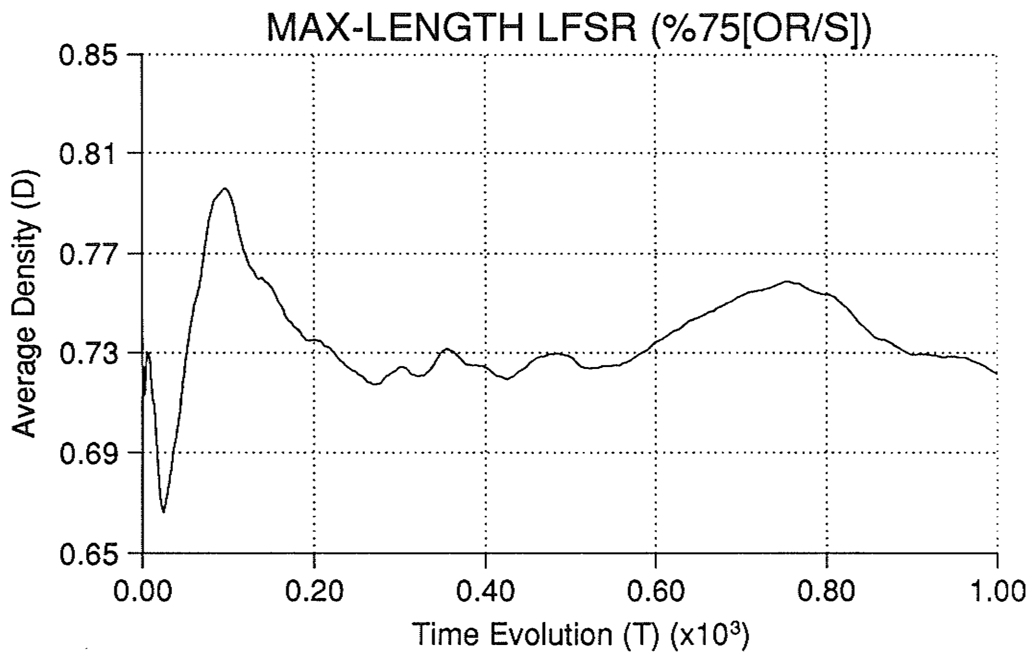

Figure D.10b): *Average Density Evolution of test patterns for LFSR weighted to 75% (OR-gate bank): [S]; W=29; L=1,000.*

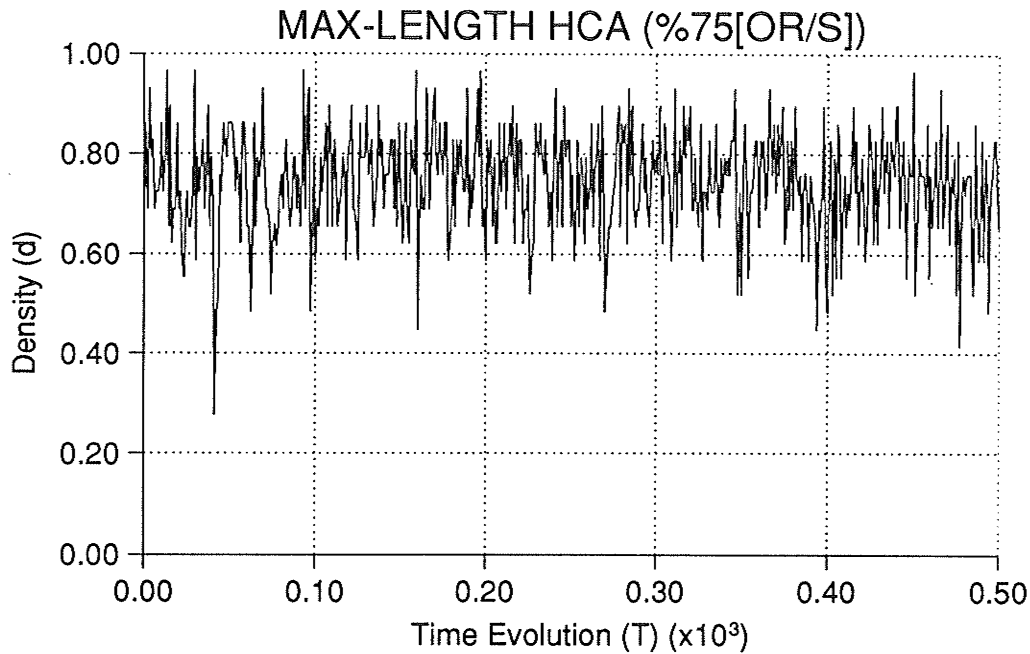

Figure D.11a): *Density Evolution of test patterns for Rule 90/150 HCA weighted to 75% (OR-gate bank): [S]; W=29; L=500.*

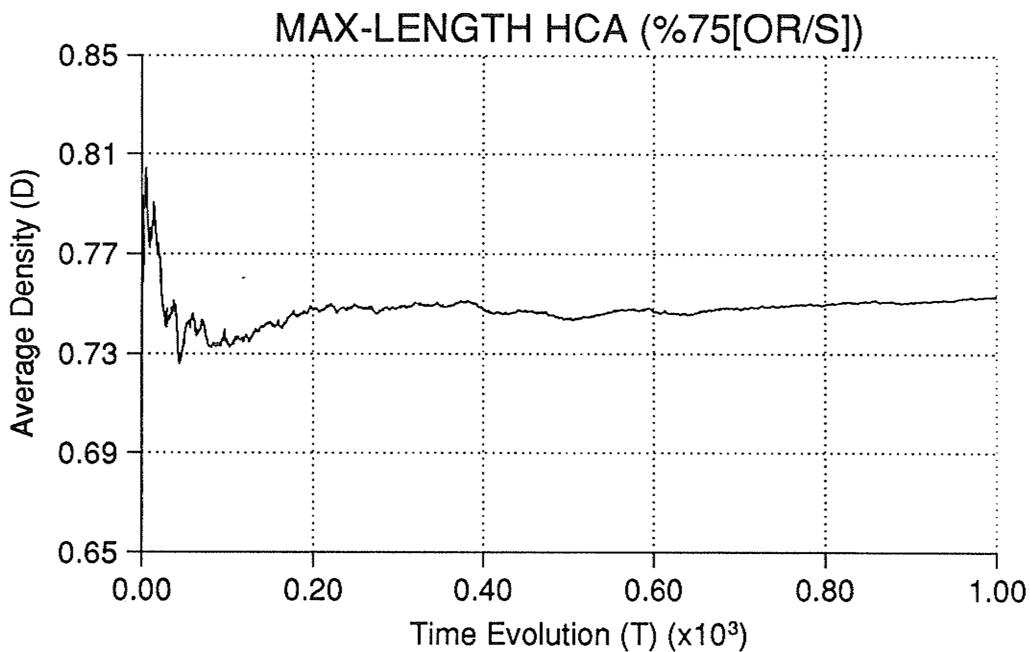

Figure D.11b): *Average Density Evolution of test patterns for Rule 90/150 HCA weighted to 75% (OR-gate bank): [S]; W=29; L=1,000.*

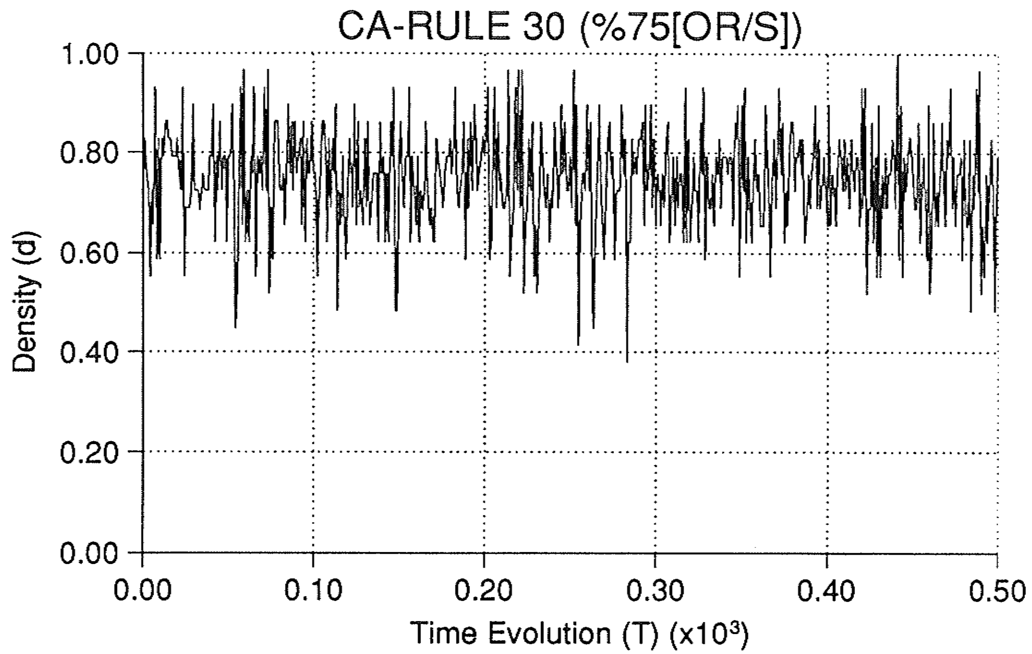

Figure D.12a): *Density Evolution of test patterns for Rule 30 CA: weighted to 75% (OR-gate bank): [S]; W=29; L=500.*

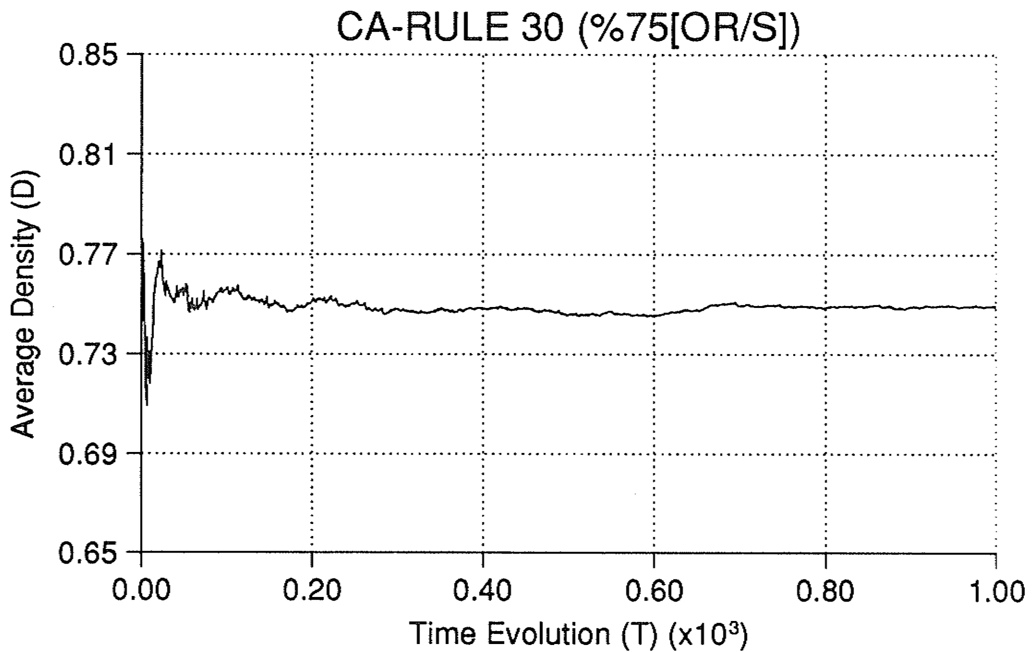

Figure D.12b): *Average Density Evolution of test patterns for Rule 30 CA weighted to 75% (OR-gate bank): [S]; W=29; L=1,000.*

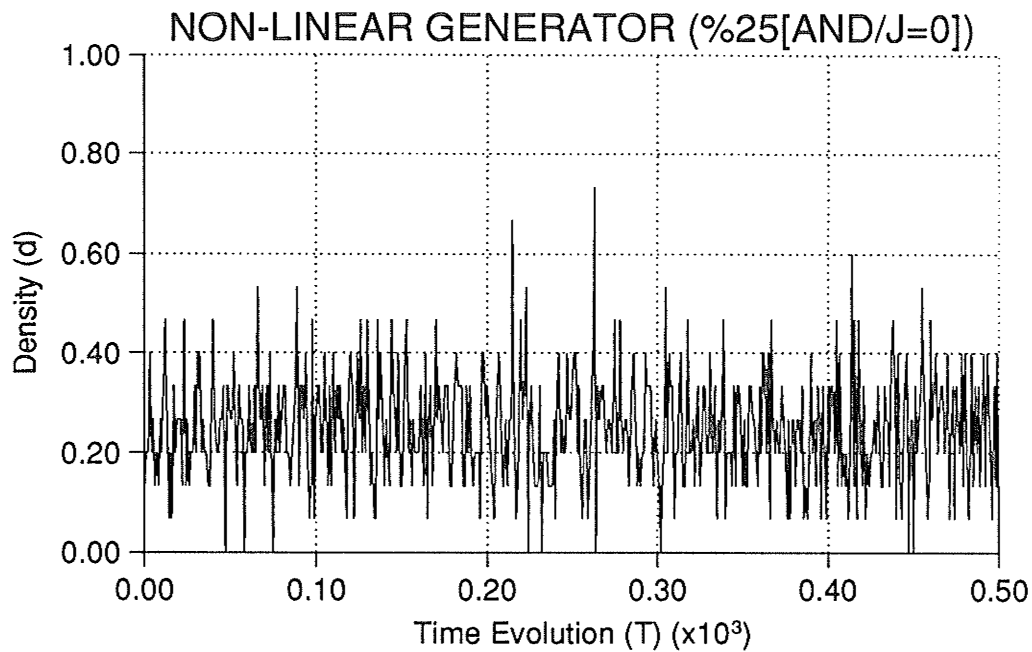

Figure D.13a): *Density Evolution of test patterns for NLFSR weighted to 25% (AND-gate bank): [J=0]; W=15; L=500.*

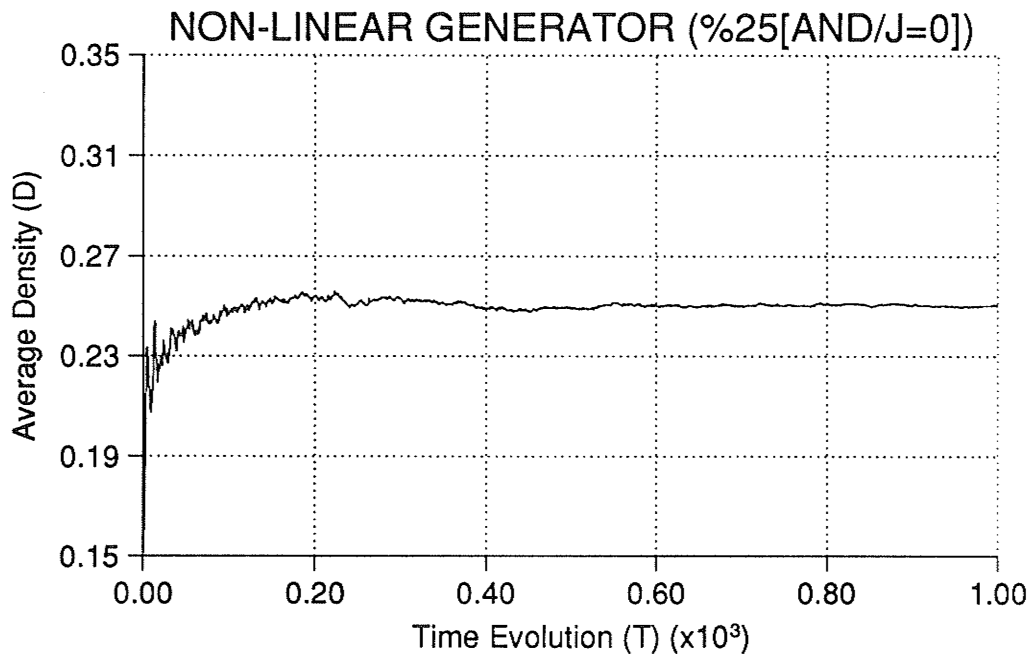

Figure D.13b): *Average Density Evolution of test patterns for NLFSR weighted to 25% (AND-gate bank): [J=0]; W=15; L=1,000.*

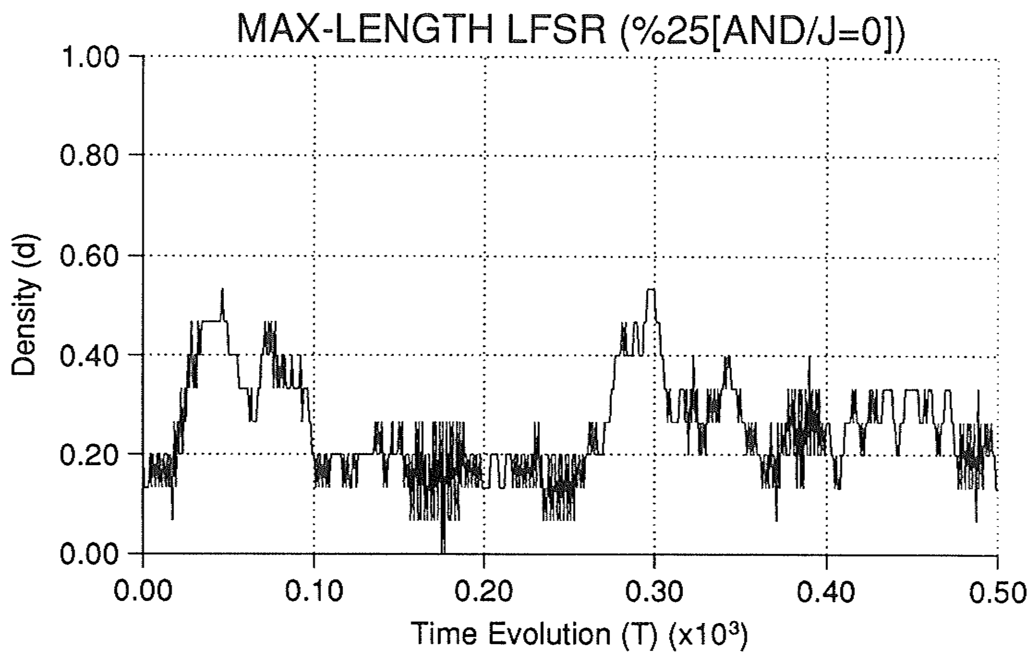

Figure D.14a): *Density Evolution of test patterns for LFSR weighted to 25% (AND-gate bank): [J=0]; W=15; L=500.*

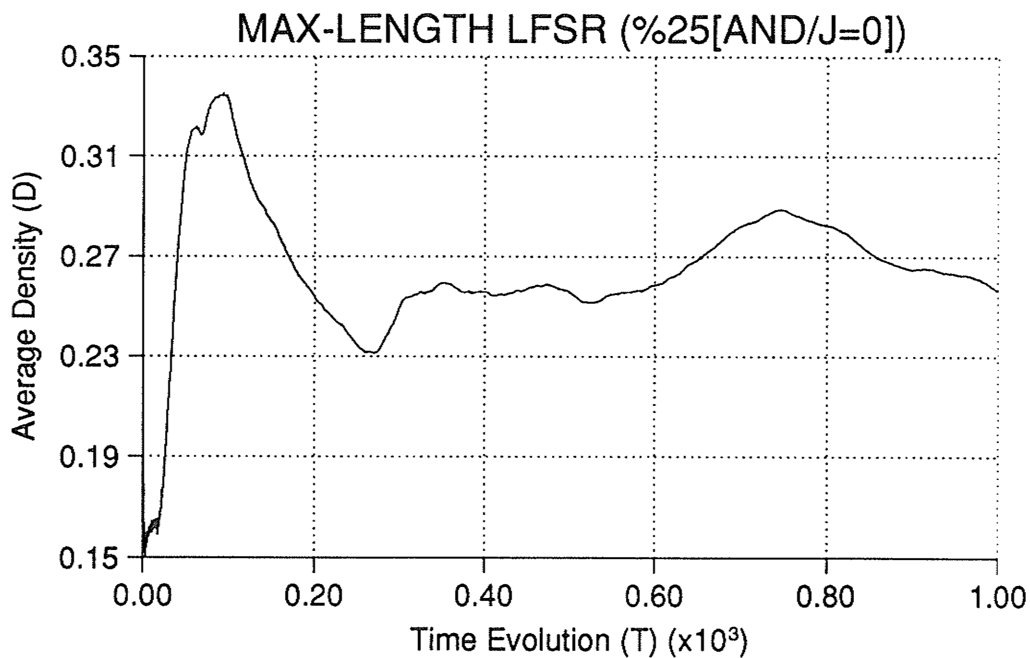

Figure D.14b): *Average Density Evolution of test patterns for LFSR weighted to 25% (AND-gate bank): [J=0]; W=15; L=1,000.*

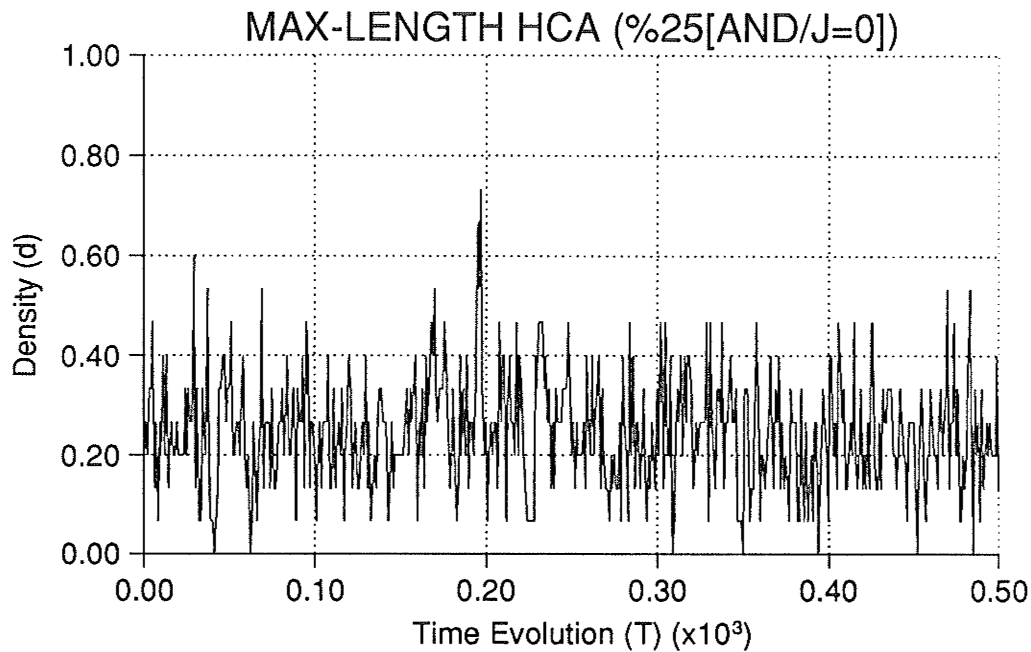

Figure D.15a): *Density Evolution of test patterns for Rule 90/150 HCA weighted to 25% (AND-gate bank): [J=0]; W=15; L=500.*

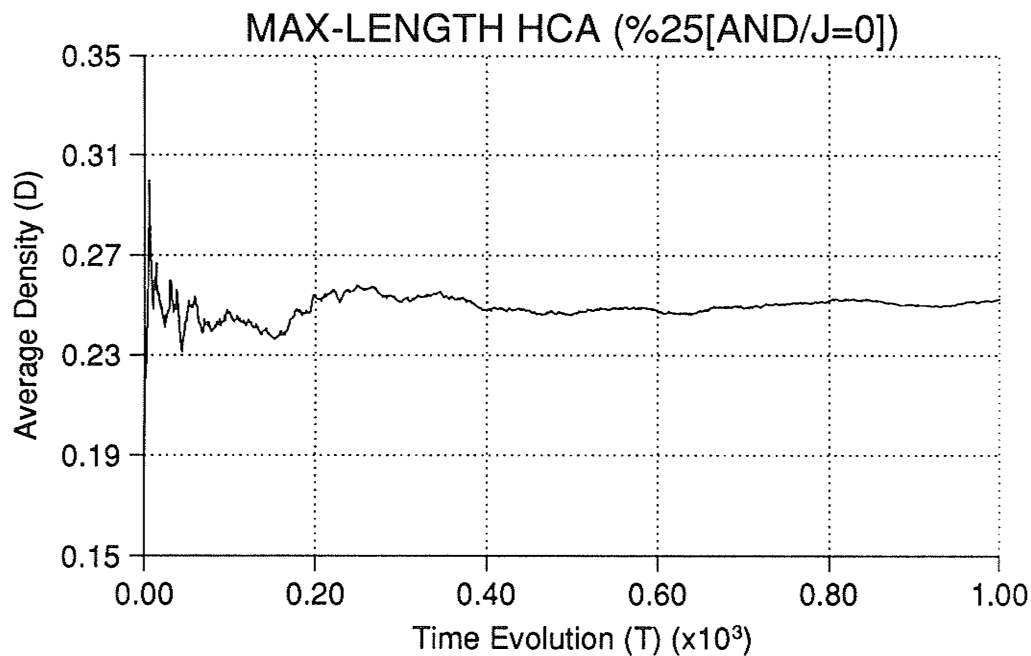

Figure D.15b): *Average Density Evolution of test patterns for Rule 90/150 HCA weighted to 25% (AND-gate bank): [J=0]; W=15; L=1,000.*

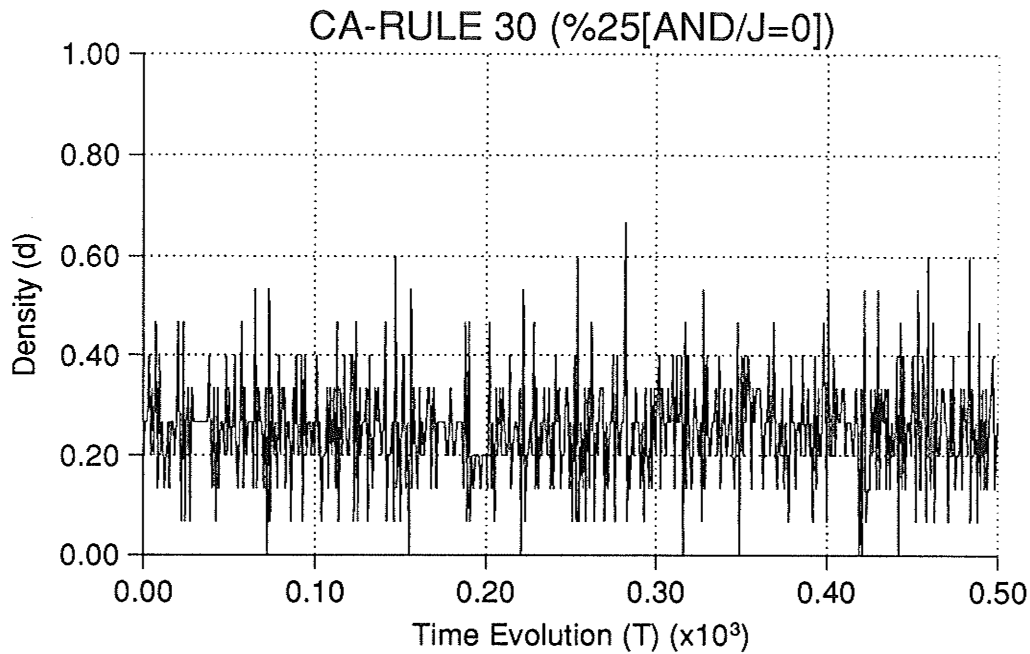

Figure D.16a): *Density Evolution of test patterns for Rule 30 CA: weighted to 25% (AND-gate bank): [J=0]; W=15; L=500.*

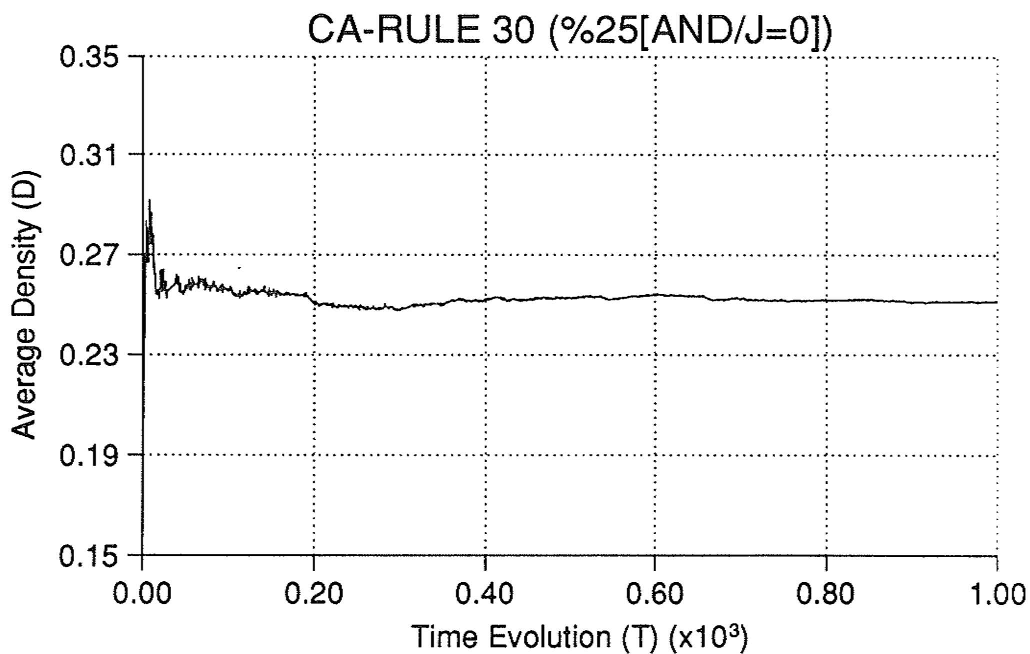

Figure D.16b): *Average Density Evolution of test patterns for Rule 30 CA weighted to 25% (AND-gate bank): [J=0]; W=15; L=1,000.*

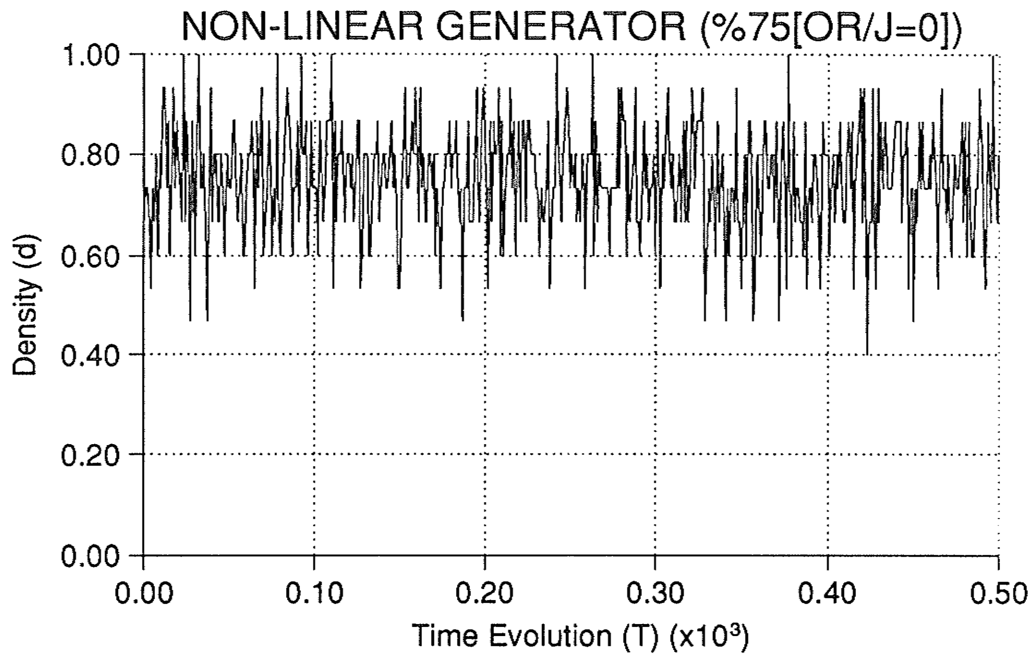

Figure D.17a): *Density Evolution of test patterns for NLFSR weighted to 75% (OR-gate bank): [J=0]; W=15; L=500.*

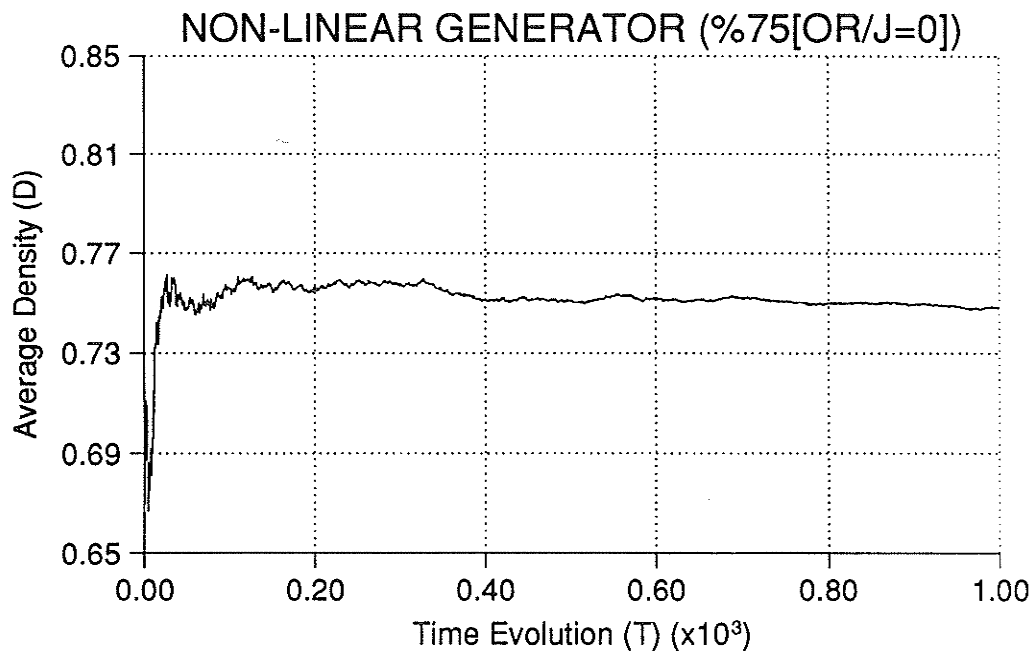

Figure D.17b): *Average Density Evolution of test patterns for NLFSR weighted to 75% (OR-gate bank): [J=0]; W=15; L=1,000.*

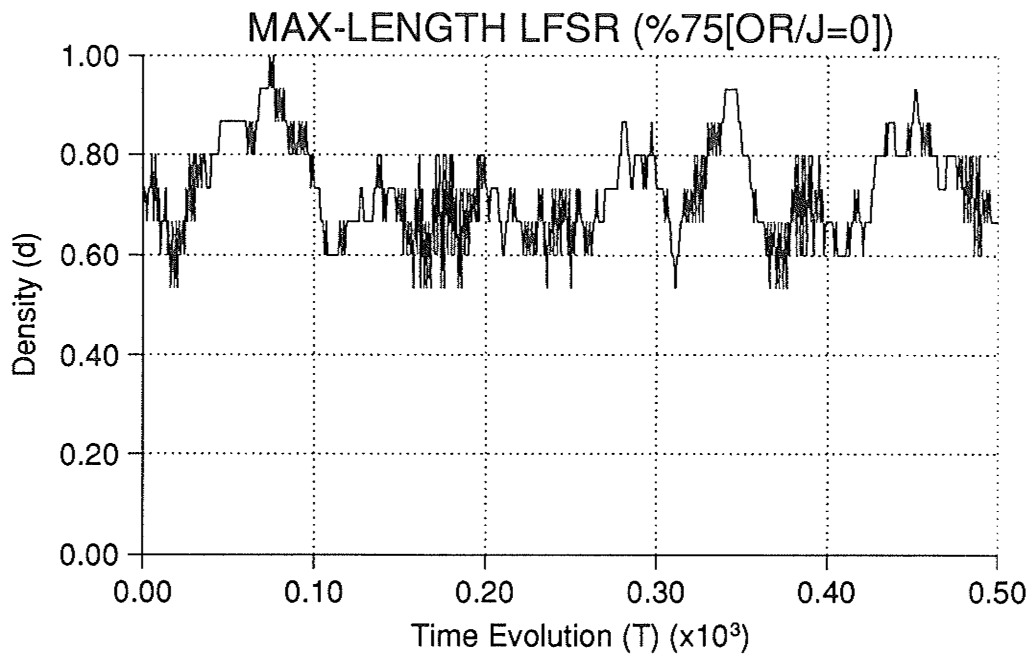

Figure D.18a): *Density Evolution of test patterns for LFSR weighted to 75% (OR-gate bank): [J=0]; W=15; L=500.*

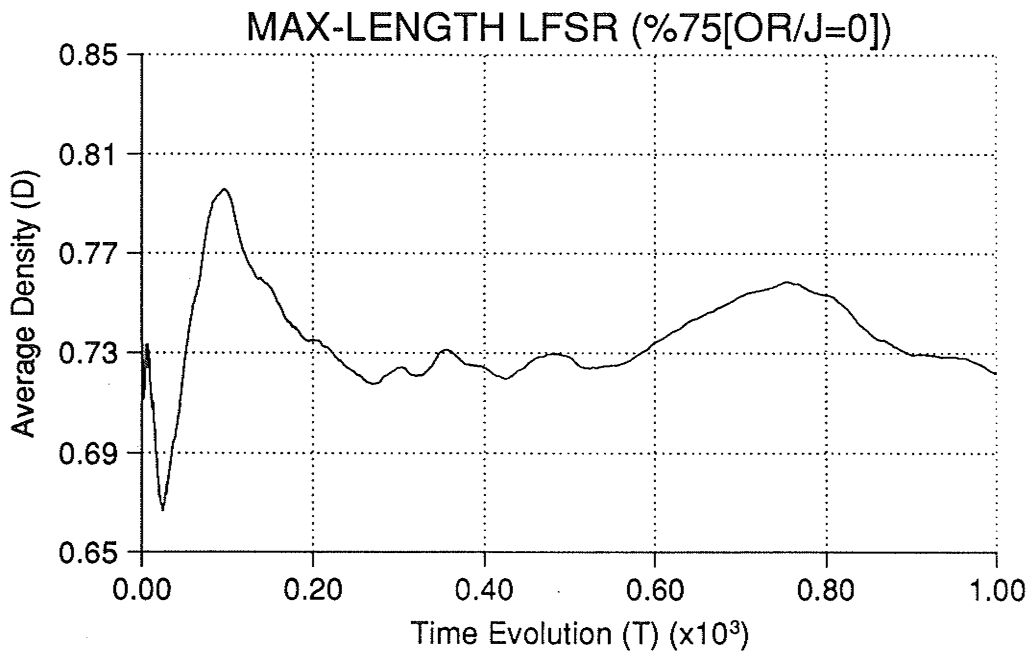

Figure D.18b): *Average Density Evolution of test patterns for LFSR weighted to 75% (OR-gate bank): [J=0]; W=15; L=1,000.*

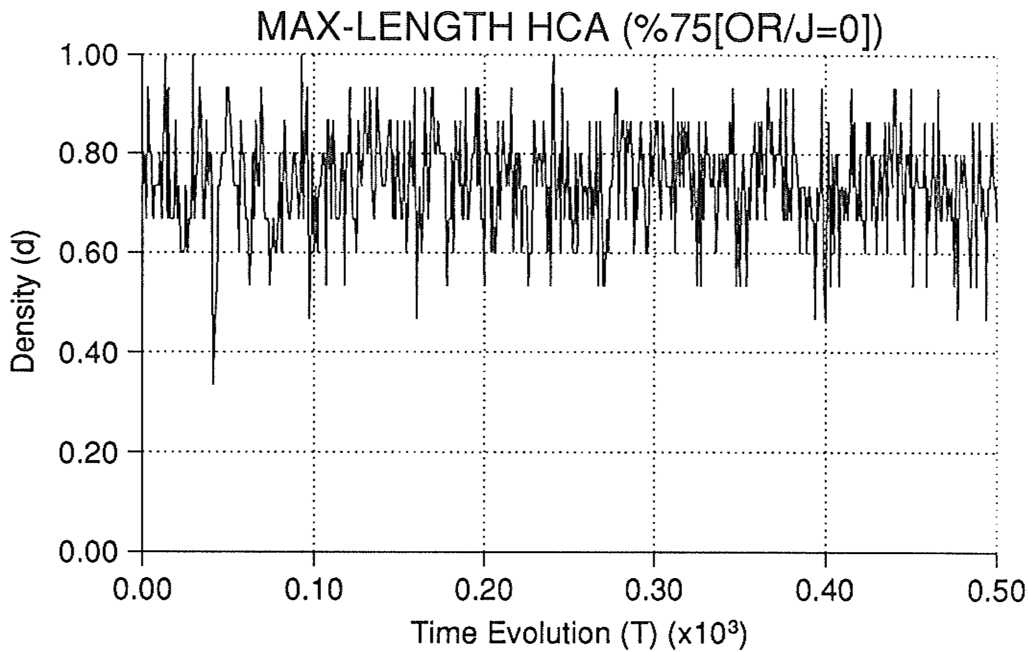

Figure D.19a): *Density Evolution of test patterns for Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=0]; W=15; L=500.*

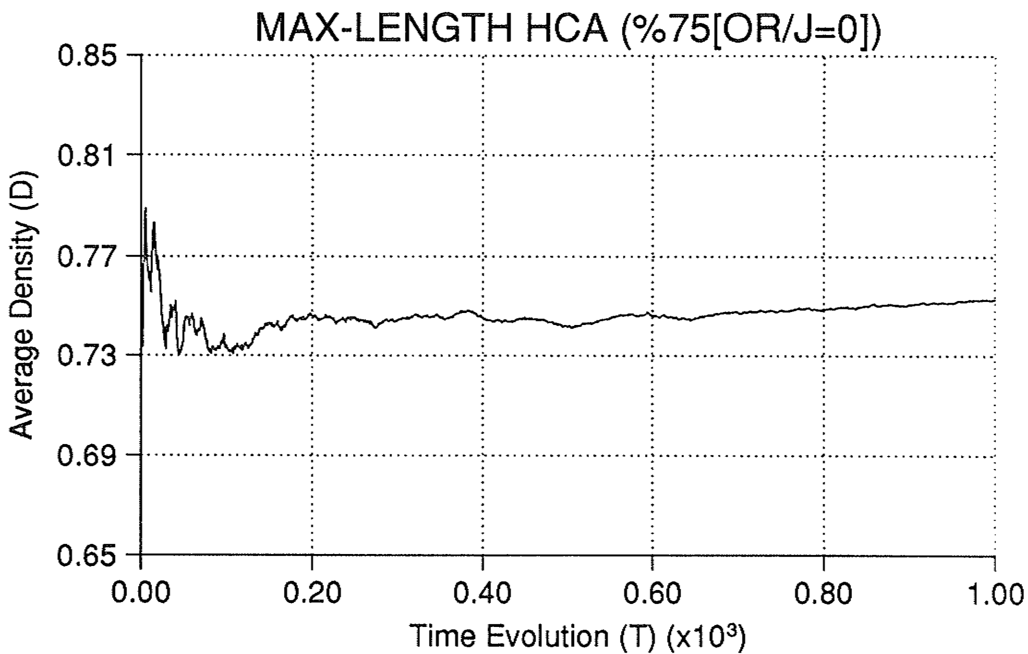

Figure D.19b): *Average Density Evolution of test patterns for Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=0]; W=15; L=1,000.*

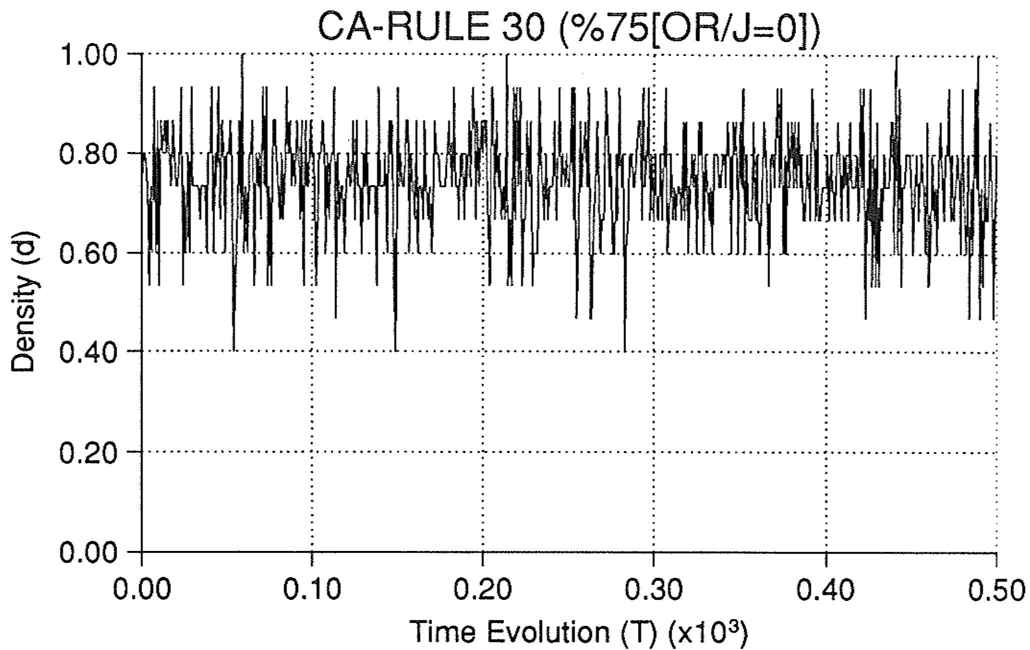

Figure D.20a): *Density Evolution of test patterns for Rule 30 CA: weighted to 75% (OR-gate bank): [J=0]; W=15; L=500.*

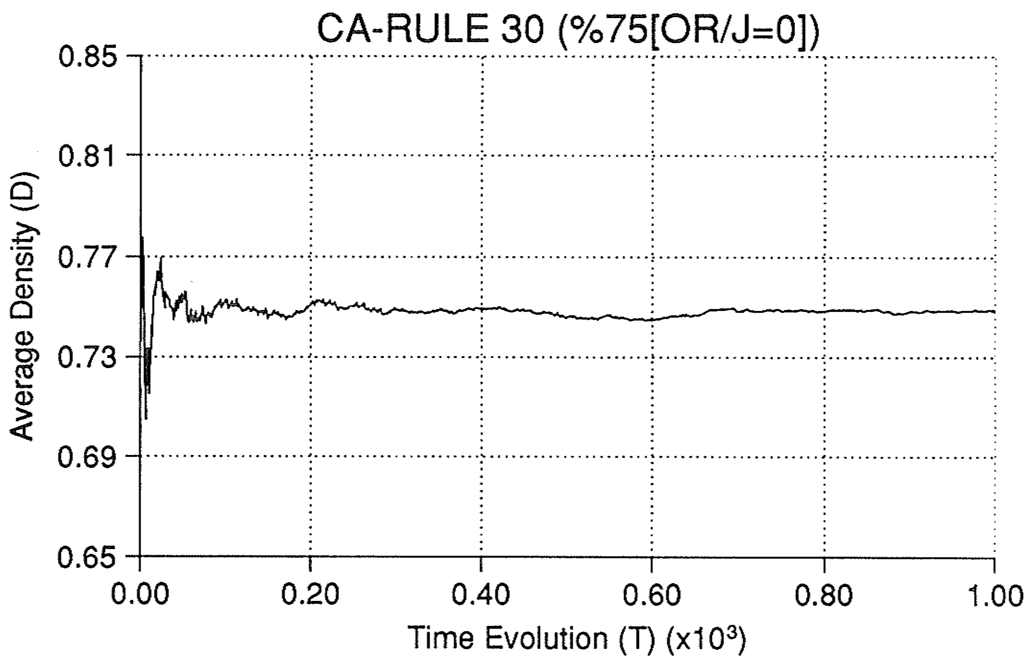

Figure D.20b): *Average Density Evolution of test patterns for Rule 30 CA weighted to 75% (OR-gate bank): [J=0]; W=15; L=1,000.*

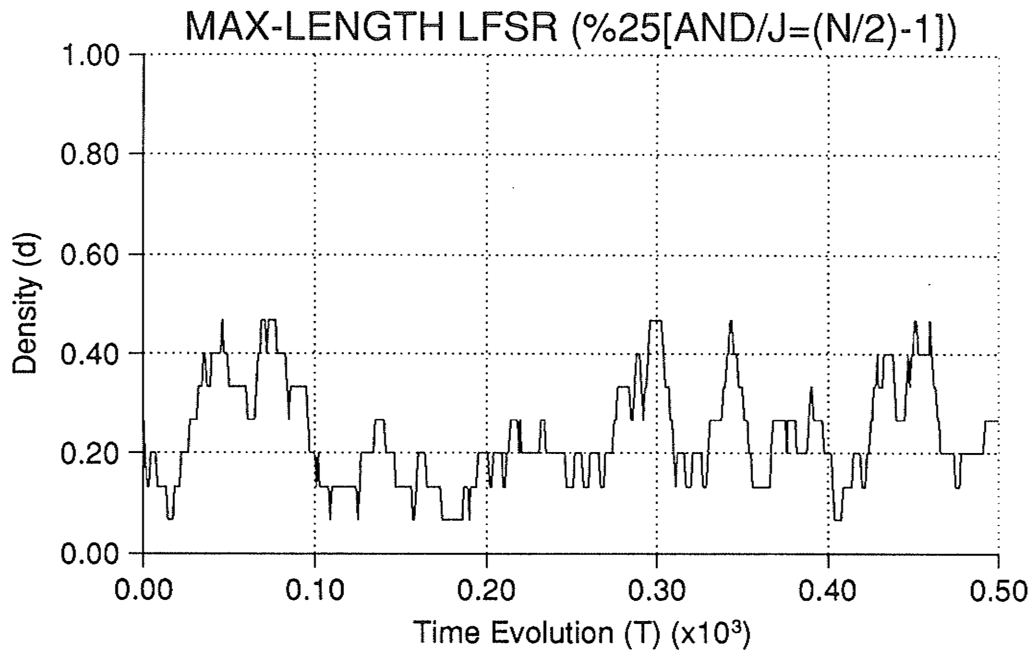

Figure D.21a): *Density Evolution of test patterns for LFSR weighted to 25% (AND-gate bank): [J=14]; W=15; L=500.*

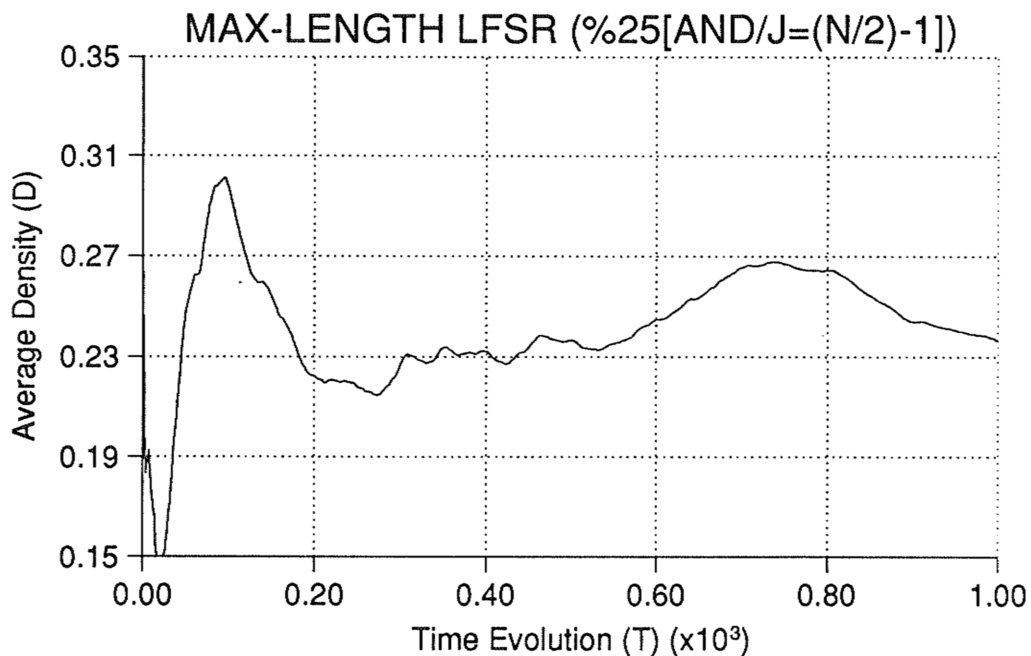

Figure D.21b): *Average Density Evolution of test patterns for LFSR weighted to 25% (AND-gate bank): [J=14]; W=15; L=1,000.*

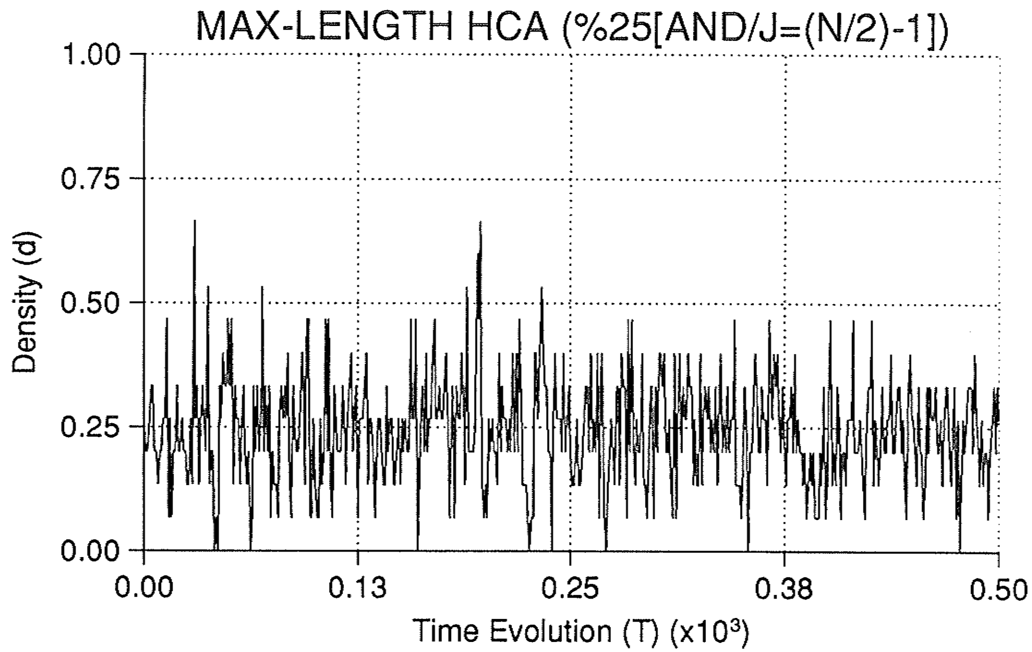

Figure D.22a): *Density Evolution of test patterns for Rule 90/150 HCA weighted to 25% (AND-gate bank): [J=14]; W=15; L=500.*

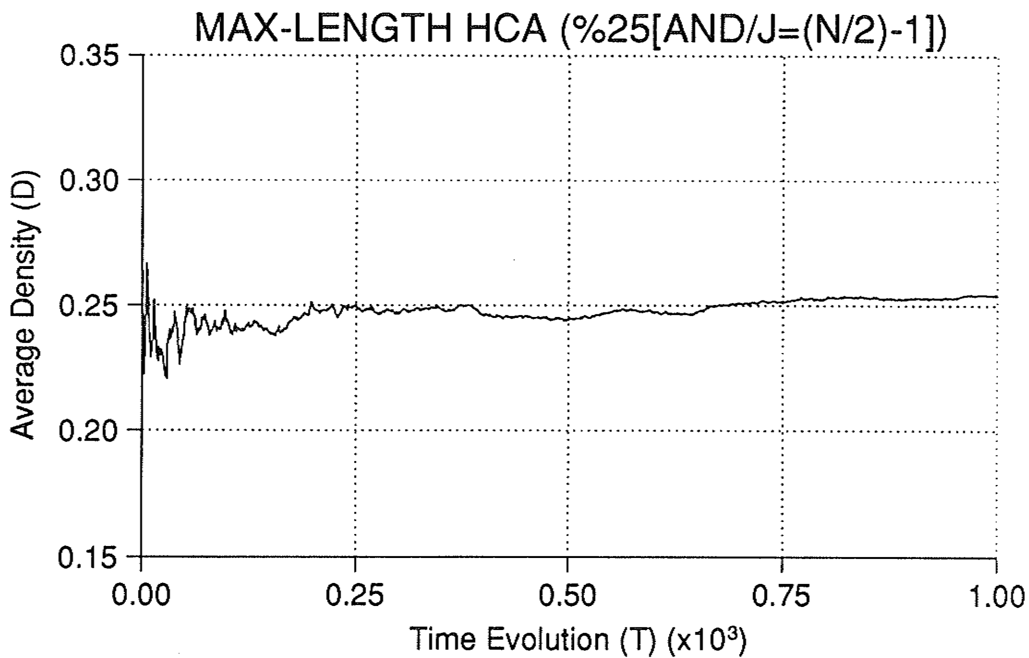

Figure D.22b): *Average Density Evolution of test patterns for Rule 90/150 HCA weighted to 25% (AND-gate bank): [J=14]; W=15; L=1,000.*

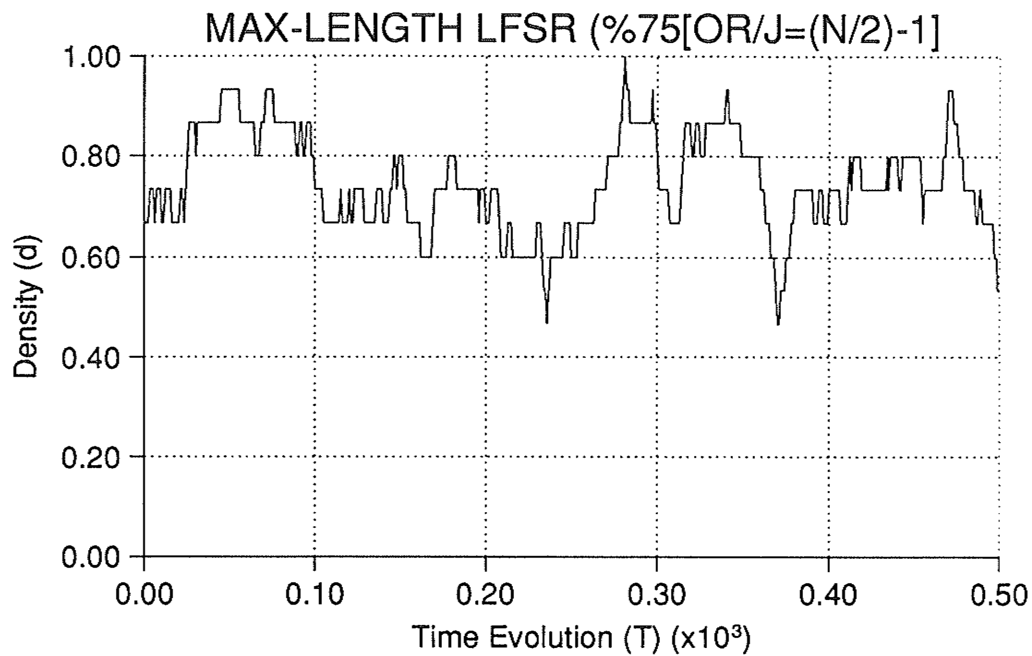

Figure D.23a): *Density Evolution of test patterns for LFSR weighted to 75% (OR-gate bank): [J=14]; W=15; L=500.*

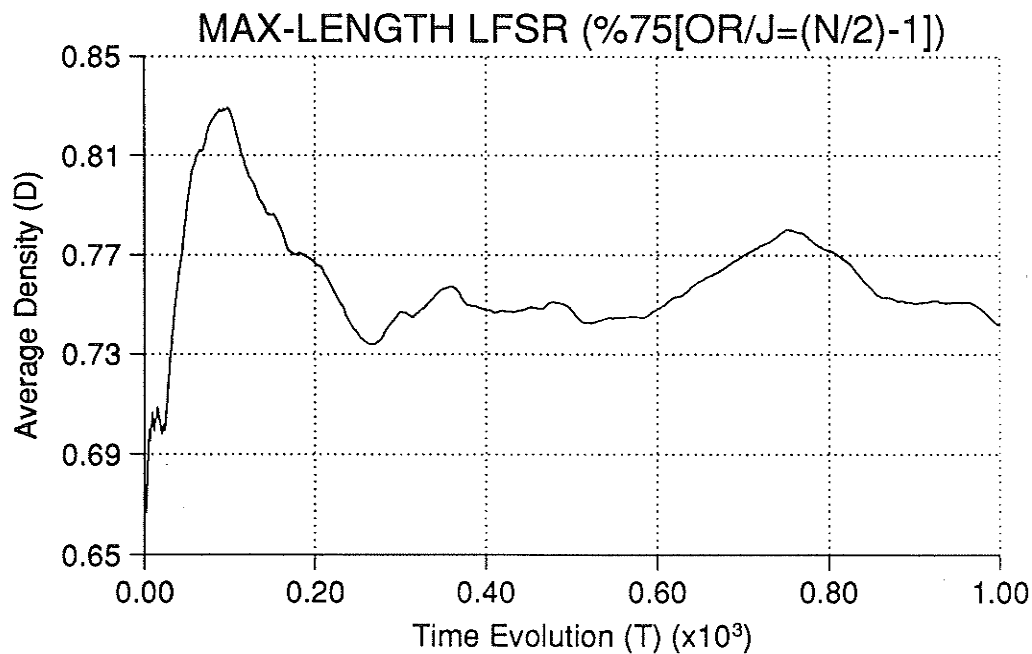

Figure D.23b): *Average Density Evolution of test patterns for LFSR weighted to 75% (OR-gate bank): [J=14]; W=15; L=1,000.*

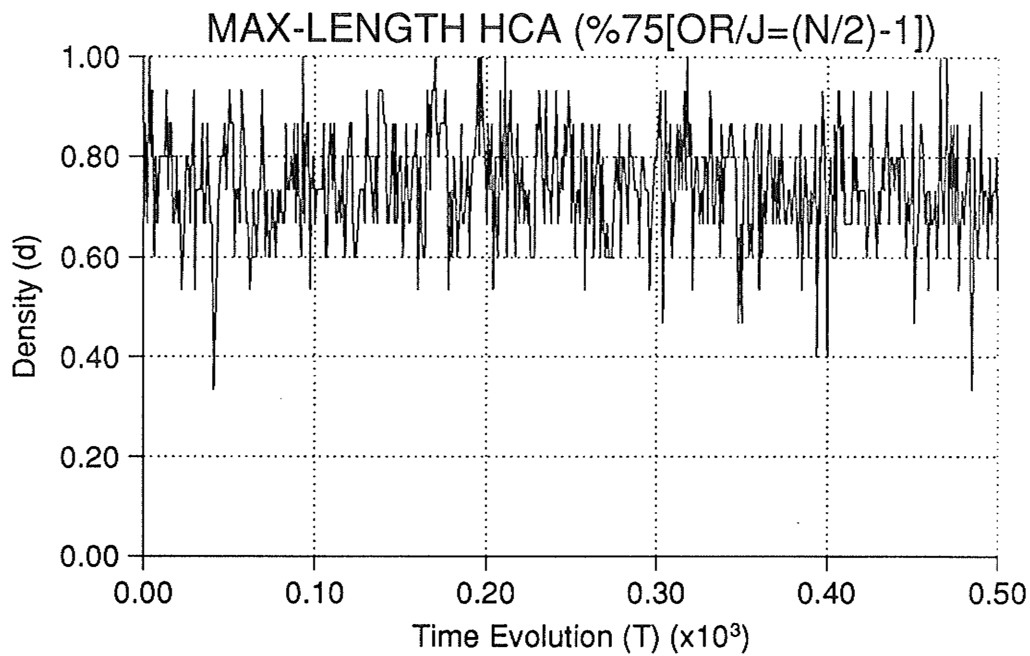

Figure D.24a): *Density Evolution of test patterns for Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=14]; W=15; L=500.*

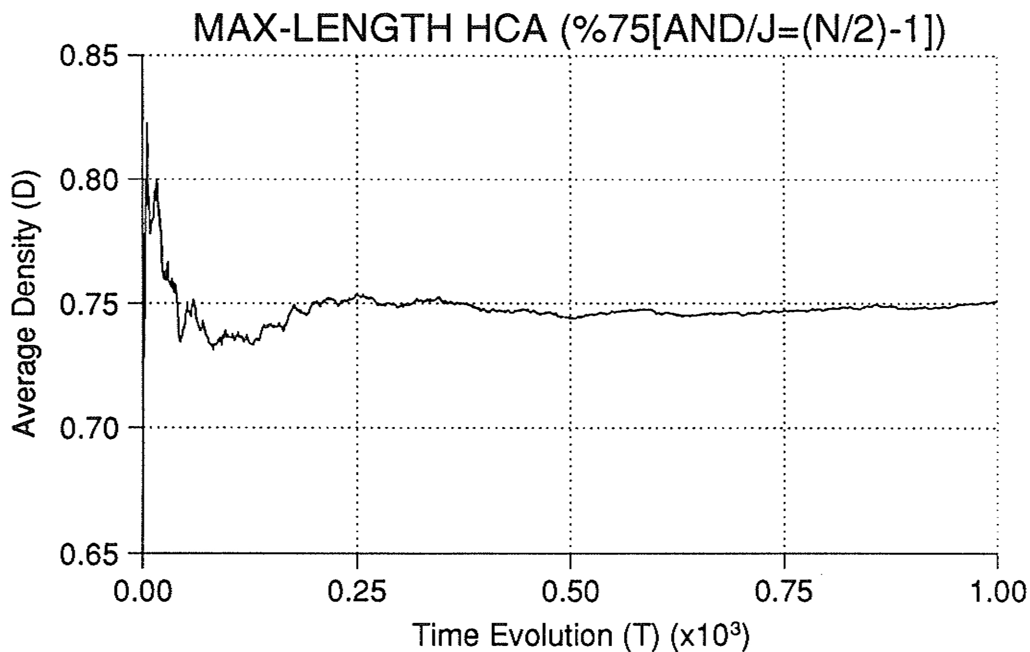

Figure D.24b): *Average Density Evolution of test patterns for Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=14]; W=15; L=1,000.*

Appendix E
PMF Evaluation

Definitions and Theorems

Definition E.1:

Let x be any discrete random variable, and $f(x)$ be an arbitrary funciton of x. If the random variable has a known probability mass function, $p(x)$, then the expected value of $f(x)$ is

$$E[f(x)] = \sum_{\text{for } all\ x} f(x)p(x). \tag{E.1}$$

Theorem E.1:

A Bernoulli random variable, x, has the occurance of only two values, that is, it is a two-valued function. For this kind of random variable, the probability mass function can be written as [Mendenhall84]

$$p_{ber}(x) = \begin{cases} p & ;\ \text{for } x = 1 \\ 1-p & ;\ \text{for } x = 0 \\ 0 & ;\ otherwise; \end{cases} \tag{E.2}$$

and the probability distribution function can be denoted by

$$F_{ber}(x) = \begin{cases} 0 & ; \text{for } x < 1 \\ 1-p & ; \text{for } 0 \le x < 1 \\ 1 & ; \text{for } x \ge 1, \end{cases} \tag{E.3}$$

where p is the probability of an outcome of 1.

Furthermore, it can be shown that a sequence generated by successive Bernoulli trials has a mean

$$\mu_{ber} = p \ , \tag{E.4}$$

and a variance

$$\sigma^2_{ber} = p(1-p) \, . \tag{E.5}$$

Proof:

By Definition E.1,

$$\begin{aligned} \mu_{ber} &= E[x] \\ &= 1 \cdot p + 0 \cdot (1-p) \\ &= p \ , \end{aligned} \tag{E.6}$$

and

$$\sigma^2_{ber} = E[(x - \mu_{ber})^2]$$
$$= (1-p)^2 \cdot p + (0-p)^2 \cdot p$$
$$= p(1-p). \qquad \text{(E.7)}$$

Definition E.2:

Let the variance of a discrete random variable, x, be denoted as

$$\sigma^2 = E[(x - \mu)^2], \qquad \text{(E.8)}$$

where μ is the mean, or expected value, of x. Then

$$\sigma^2 = E[x^2 - 2\mu x + \mu^2]$$
$$= E[x^2] - 2\mu E[x] + \mu^2$$
$$= E[x^2] - \mu^2, \qquad \text{(E.9)}$$

since $E[k] = k$, and $E[kx] = k \cdot E[x]$, where k is a constant.

Theorem E.2:

The summation of W independent Bernoulli random variables forms what is known as a single Binomial random variable [Law82],

$$X = \sum_{i=1}^{W} x_i \,, \tag{E.10}$$

where x_i represents the i-th Bernoulli random variable. A Binomial random variable has a probability mass function given by

$$p_{bin}(X) = \begin{cases} \binom{W}{X} \cdot p^X (1-p)^{W-X} & ; X \in \{0, 1, \ldots, W\} \\ 0 & ; otherwise, \end{cases} \tag{E.11}$$

where p is the probability of an outcome of 1 for each Bernoulli trial; and, $\binom{W}{X}$ is the Binomial co-efficient, defined as the combination

$$\binom{W}{X} = \frac{W!}{X!(W-X)!} \,. \tag{E.12}$$

Further to this, the probability distribution function for the Binomial random variable, X, is represented by

$$F_{bin}(X) = \begin{cases} 0 & ; \text{for } X < 0 \\ \sum_{i=0}^{X} \binom{W}{i} p^i (1-p)^{W-i} & ; \text{for } 0 \le X \le W \\ 1 & ; \text{for } X > W. \end{cases} \tag{E.13}$$

It can also be shown that a sequence generated by successive Binomial trials has a mean

$$\mu_{bin} = W \cdot p \ , \tag{E.14}$$

and a variance

$$\sigma^2_{bin} = W \cdot p(1-p) \, . \tag{E.15}$$

Proof:

Using Definition E.1,

$$\begin{aligned} \mu_{bin} &= E[X] \\ &= \sum_{X=0}^{W} X \, p_{bin}(X) \\ &= \sum_{X=0}^{W} X \cdot \frac{W!}{X!(W-X)!} \cdot p^X (1-p)^{W-X} \, . \end{aligned} \tag{E.16}$$

Because the first term of the series is equal to 0, it can be omitted from the expression. Thus,

$$\begin{aligned} \mu_{bin} &= \sum_{X=1}^{W} X \cdot \frac{W!}{X!(W-X)!} \cdot p^X (1-p)^{W-X} \, . \\ &= \sum_{X=1}^{W} \frac{W!}{(X-1)!(W-X)!} \cdot p^X (1-p)^{W-X} \, . \end{aligned} \tag{E.17}$$

By rearranging the terms in equation E.17, and setting $Y = X - 1$, a transformation is made in terms of Y such that,

$$\mu_{bin} = W \cdot p \sum_{X=1}^{W} \frac{(W-1)!}{(X-1)!(W-X)!} \cdot p^{X-1}(1-p)^{W-X}$$

$$= W \cdot p \sum_{Y=0}^{W-1} \frac{(W-1)!}{Y![(W-1)-Y]!} \cdot p^{Y}(1-p)^{[W-1]-Y} . \qquad \text{(E.18)}$$

Using the fact that

$$\sum_{Y=0}^{W-1} p_{bin}(Y) = 1 , \qquad \text{(E.19)}$$

where $p_{bin}(Y)$ is a Binomial probability mass function, equation E.18 can be simplied to

$$\mu_{bin} = W \cdot p . \qquad \text{(E.20)}$$

By Definition E.2,

$$\sigma^2_{bin} = E[X^2] - \mu^2_{bin} . \qquad \text{(E.21)}$$

But, because of equation E.20, all that is required to solve equation E.21 is knowledge of $E[X^2]$. Hence, using Definition E.1,

$$E[X^2] = \sum_{X=0}^{W} X^2 \cdot \frac{W!}{X!(W-X)!} \cdot p^X (1-p)^{W-X}$$

$$= \sum_{X=1}^{W} X \cdot \frac{W!}{(X-1)!(W-X)!} \cdot p^X (1-p)^{W-X} . \qquad \text{(E.22)}$$

Incorporating the transforation $Y = X - 1$, equation E.22 can be written as

$$E[X^2] = W \cdot p \left\{ \sum_{Y=0}^{W-1} \frac{Y(W-1)!}{Y![(W-1)-Y]!} + \sum_{Y=0}^{W-1} \frac{(W-1)!}{Y![(W-1)-Y]!} \right\} p^Y (1-p)^{[W-1]-Y}$$

$$= W \cdot p \left\{ \sum_{Y=0}^{W-1} \frac{Y(W-1)!}{Y![(W-1)-Y]!} \cdot p^Y (1-p)^{(W-1)-Y} + 1 \right\}$$

$$= W \cdot p \left\{ (W-1) \cdot p \sum_{Y=0}^{W-1} \frac{(W-2)!}{(Y-1)![(W-1)-Y]!} \cdot p^{(Y-1)} (1-p)^{[W-1]-Y} + 1 \right\} \text{(E.23)}$$

With another transform, $Z = Y - 1$,

$$E[X^2] = W \cdot p \left\{ (W-1) \cdot p \sum_{Z=0}^{W-2} \frac{(W-2)!}{Z![(W-2)-Z]!} \cdot p^Z (1-p)^{[W-2]-Z} + 1 \right\}$$

$$= W \cdot p [(W-1)p + 1]$$

$$= (W \cdot p)^2 + W \cdot p (1-p) . \qquad \text{(E.24)}$$

Finally, making the necessary substitutions into equation E.21, the variance can be expressed as

$$\sigma^2_{bin} = (W \cdot p)^2 + W \cdot p (1-p) - (W \cdot p)^2$$

$$= W \cdot p (1-p) . \qquad \text{(E.25)}$$

Appendix F

Histogram Evolutions

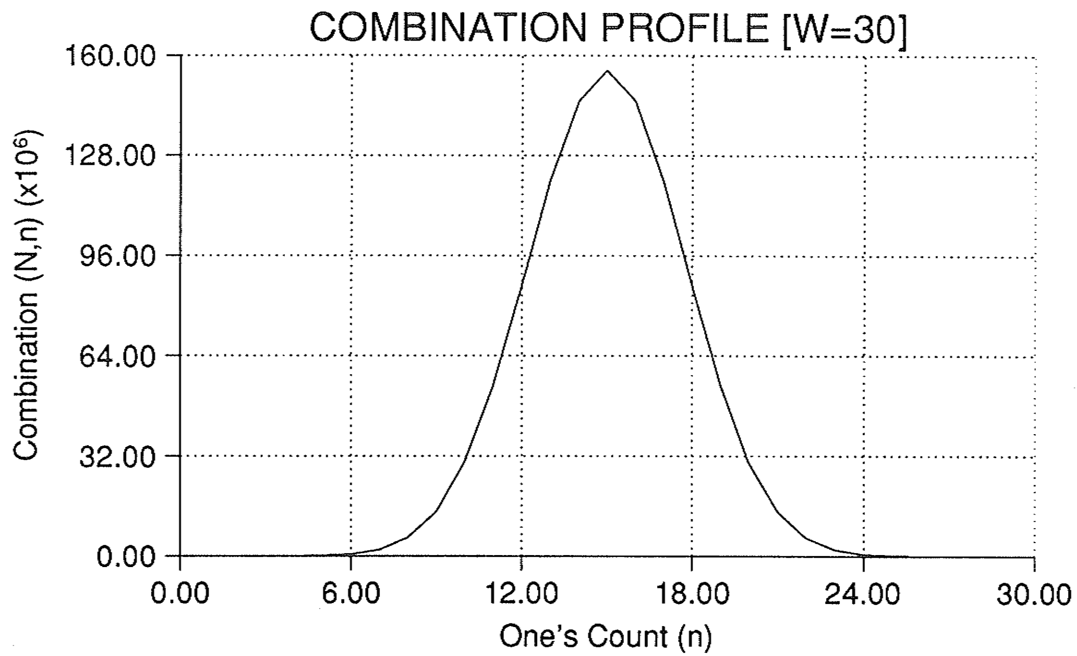

Figure F.1a): *Binomial Coefficient Combination Profile: W=30.*

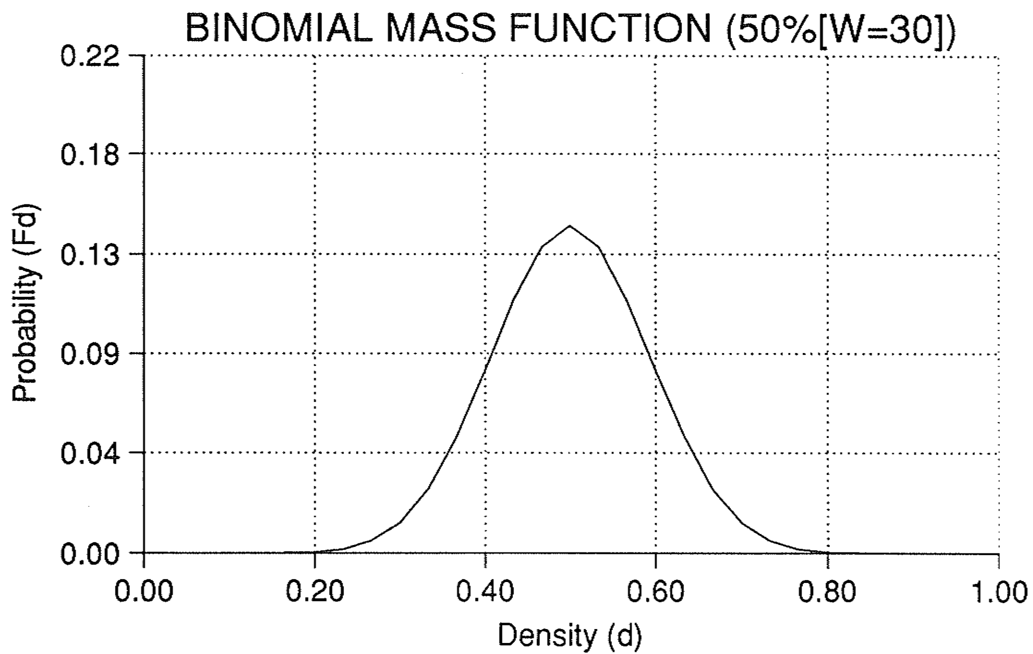

Figure F.1b): *Binomial Probability Mass Function (50%): W=30.*

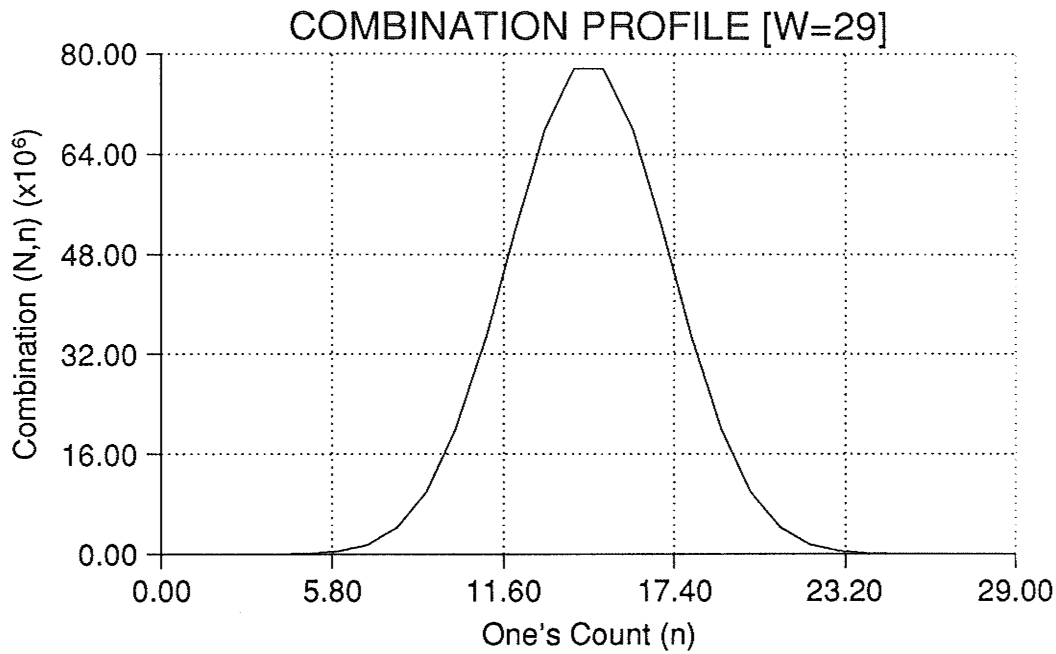

Figure F.2a): *Binomial Coefficient Combination Profile: W=29.*

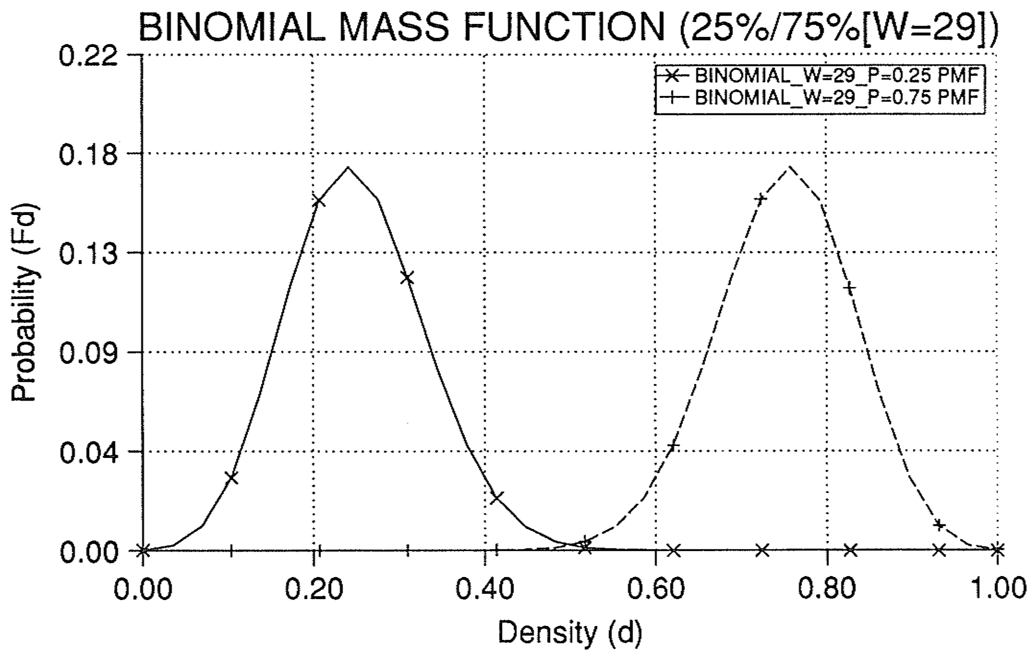

Figure F.2b): *Binomial Probability Mass Function (25% & 75%): W=29.*

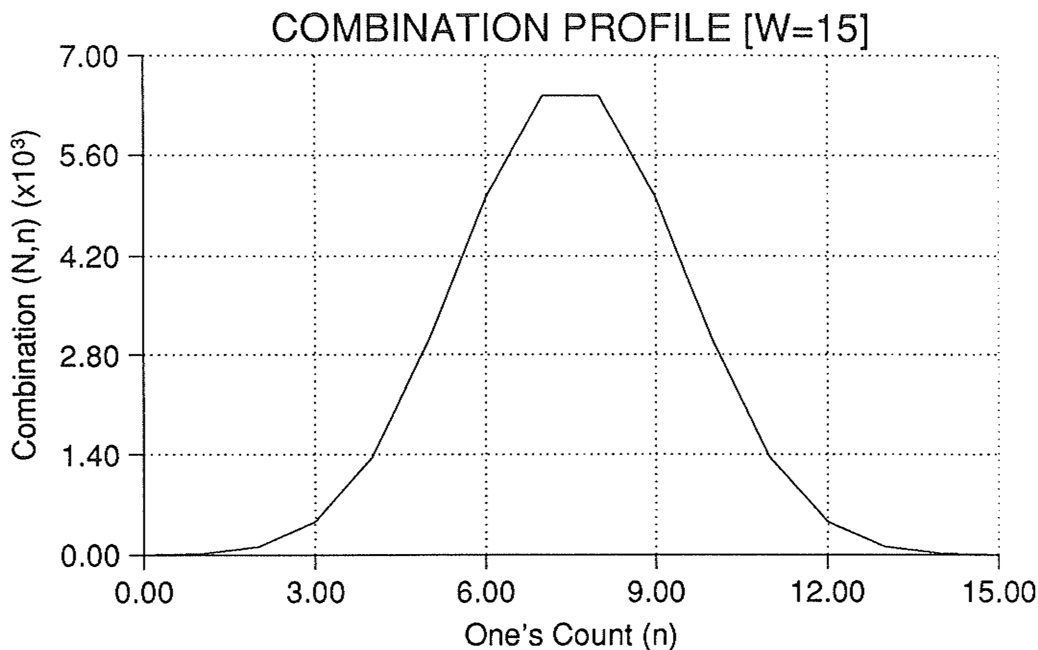

Figure F.3a): *Binomial Coefficient Combination Profile: W=15.*

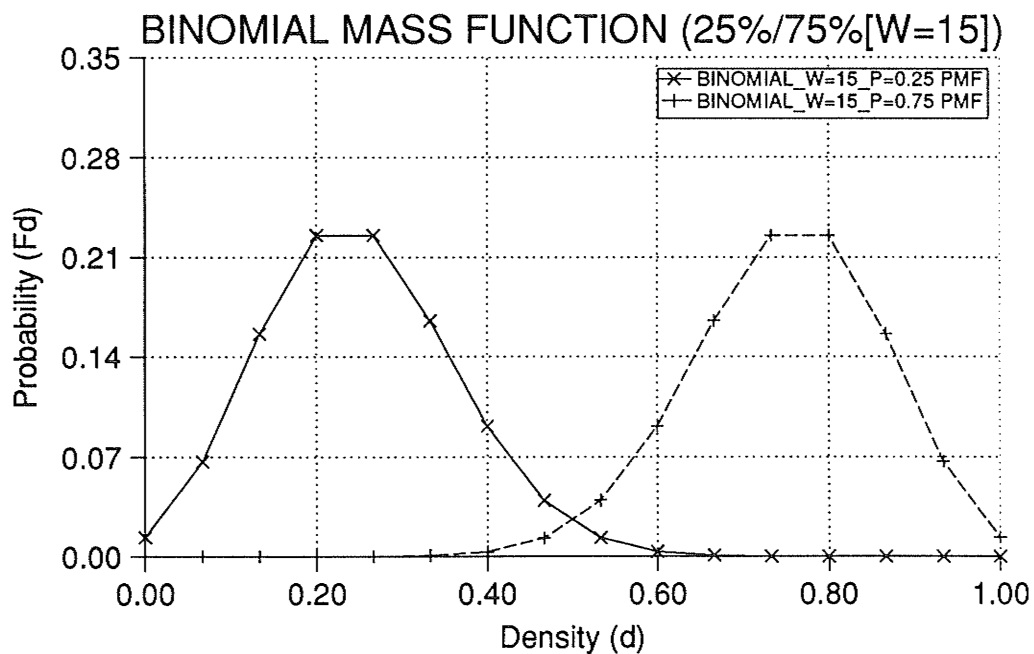

Figure F.3b): *Binomial Probability Mass Function (25% & 75%): W=15.*

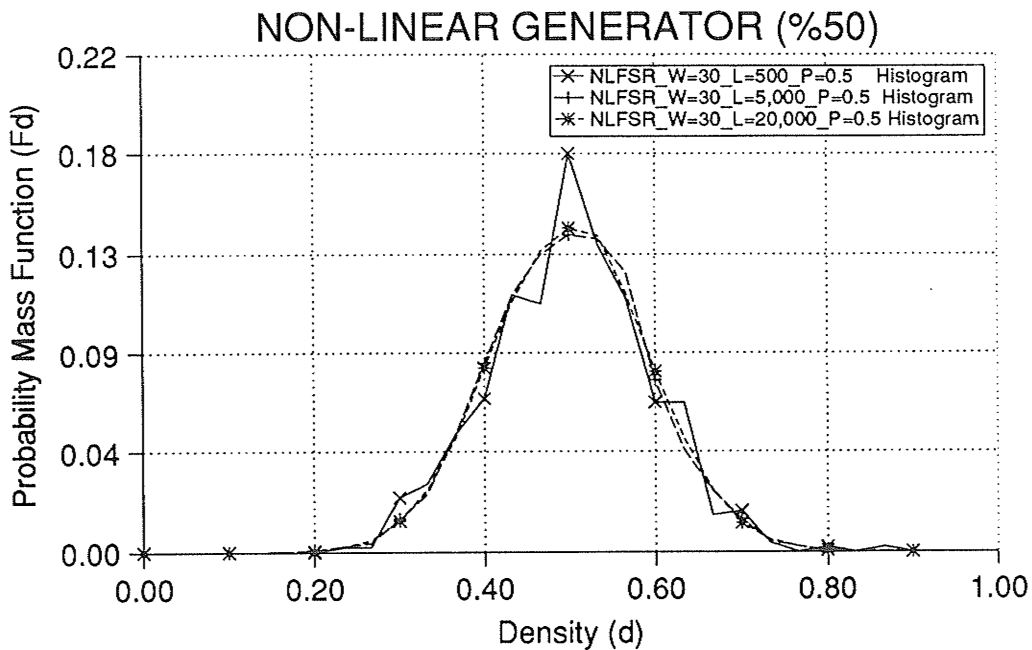

Figure F.4: *Probability Mass Function (Histogram) of NLFSR (50%): W=30.*

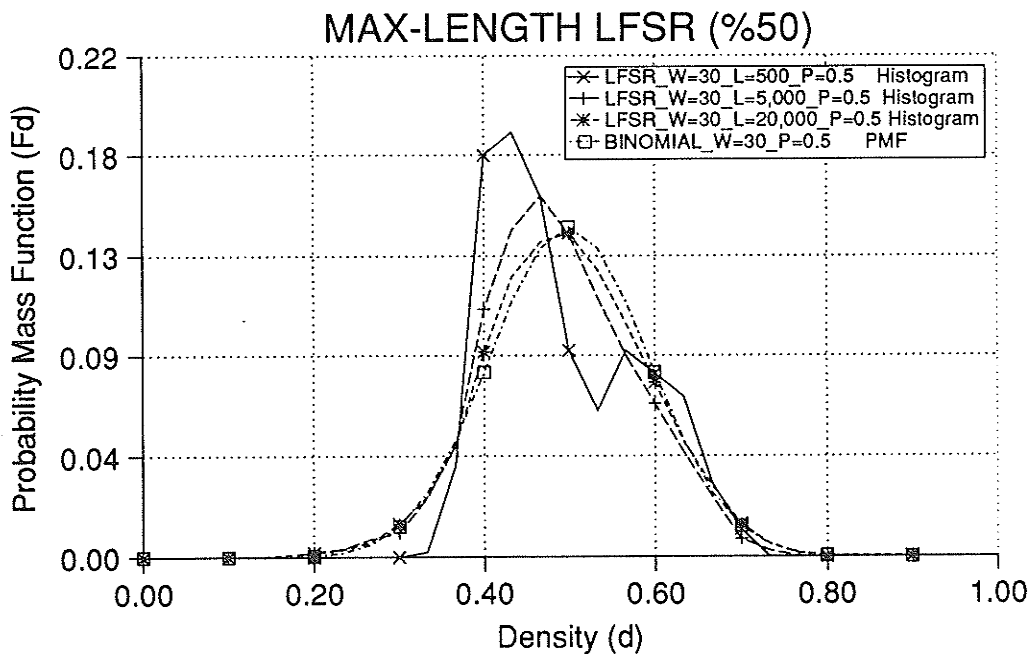

Figure F.5: *Probability Mass Function (Histogram) of LFSR (50%): W=30.*

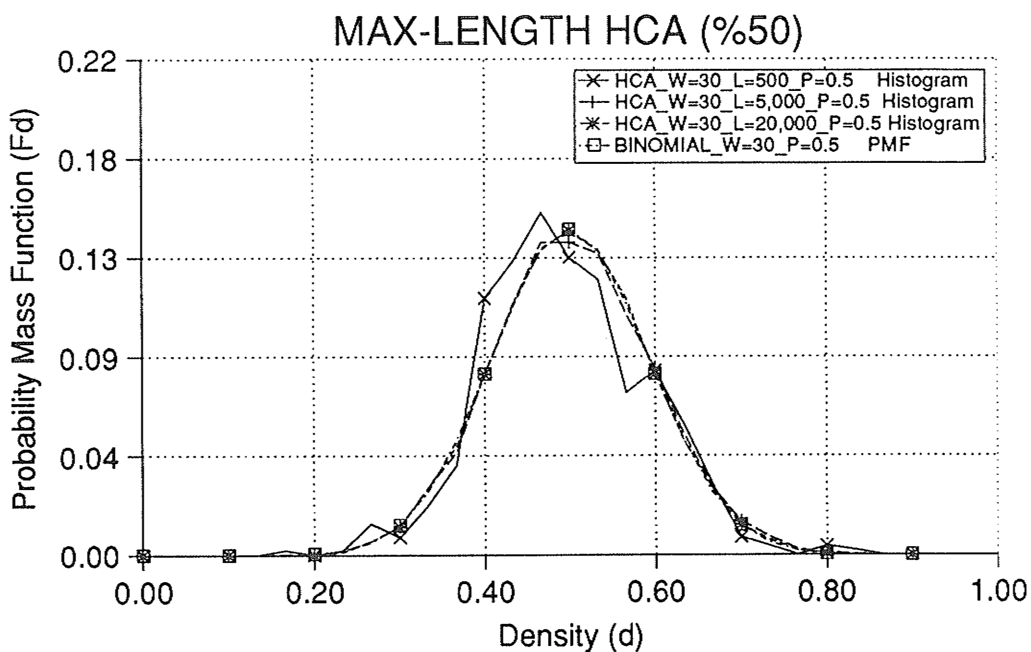

Figure F.6: *Probability Mass Function (Histogram) of Rule 90/150 HCA (50%): W=30.*

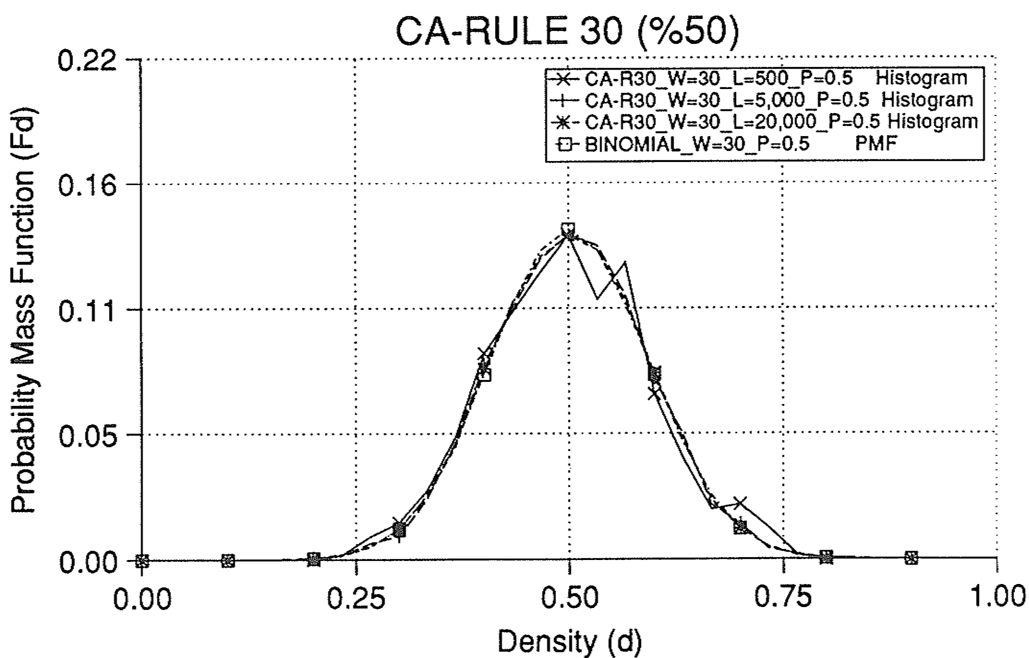

Figure F.7: *Probability Mass Function (Histogram) of Rule 30 CA (50%): W=30.*

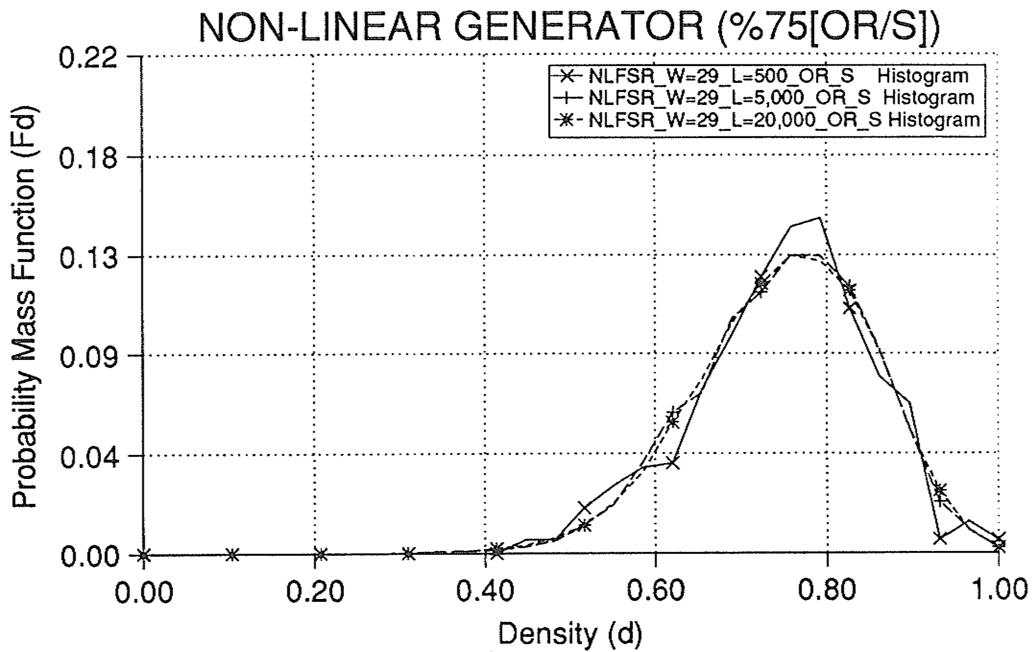

Figure F.8: *Probability Mass Function (Histogram) of NLFSR weighted to 75% (OR-gate bank): [S]; W=29.*

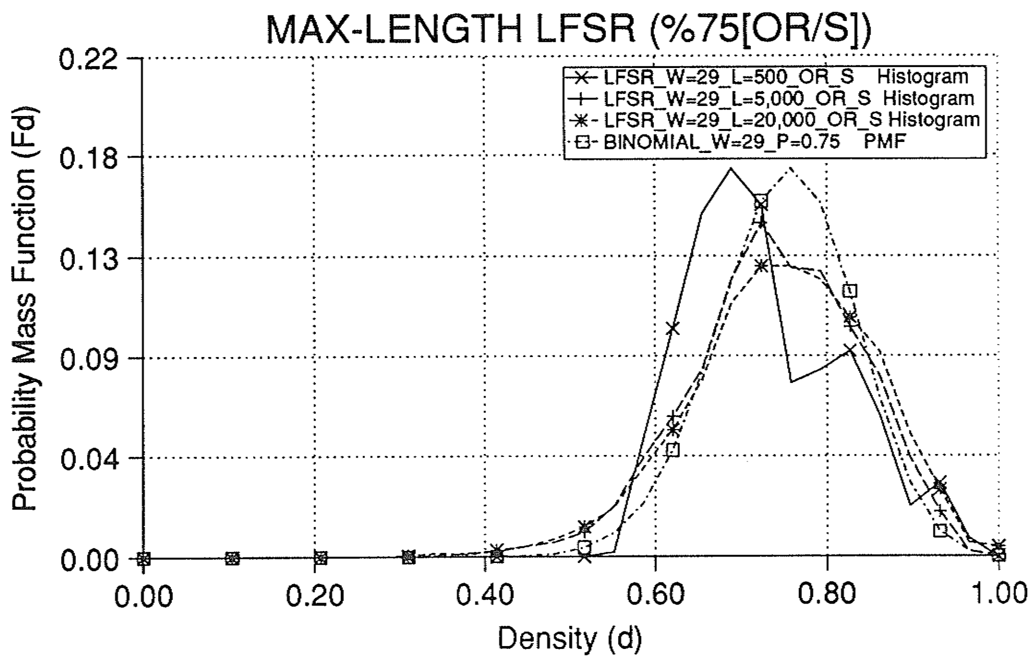

Figure F.9: *Probability Mass Function (Histogram) of LFSR weighted to 75% (OR-gate bank): [S]; W=29.*

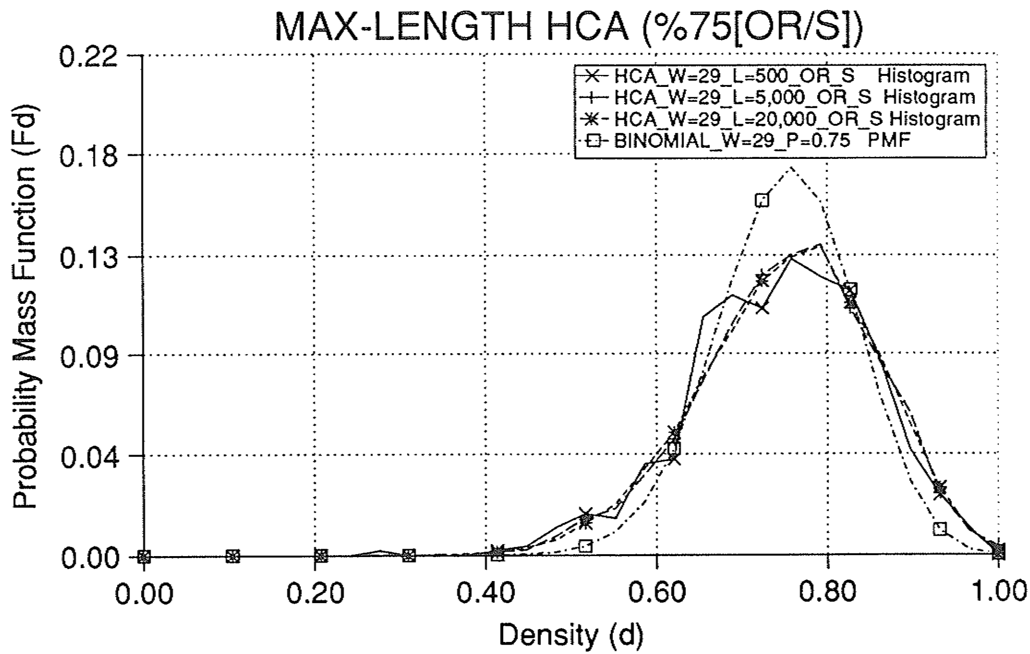

Figure F.10: *Probability Mass Function (Histogram) of Rule 90/150 HCA weighted to 75% (OR-gate bank): [S]; W=29.*

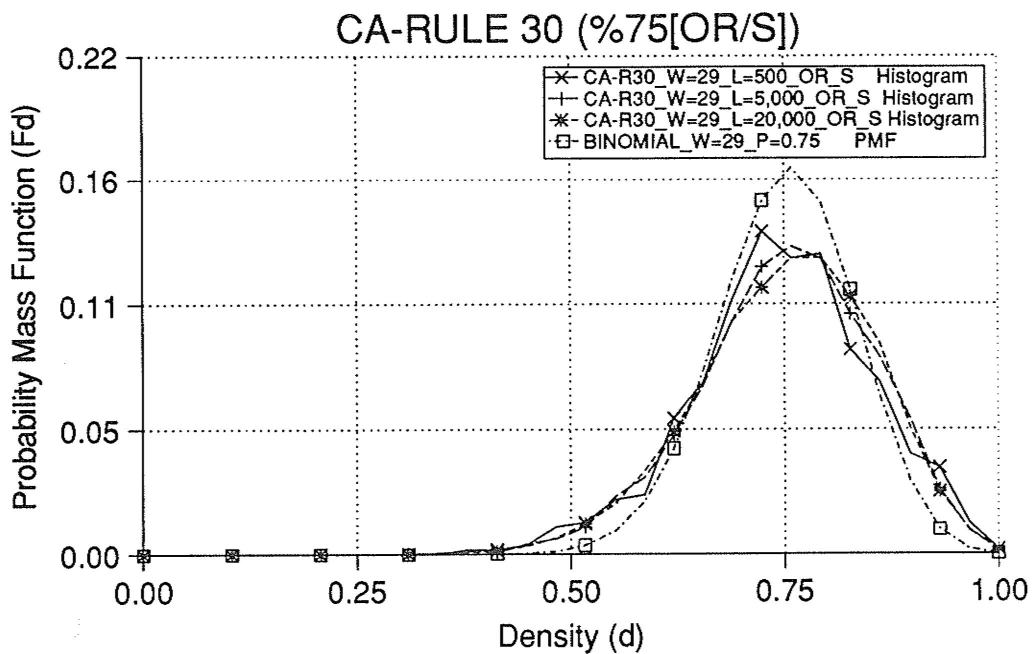

Figure F.11: *Probability Mass Function (Histogram) of Rule 30 CA weighted to 75% (OR-gate bank): [S]; W=29.*

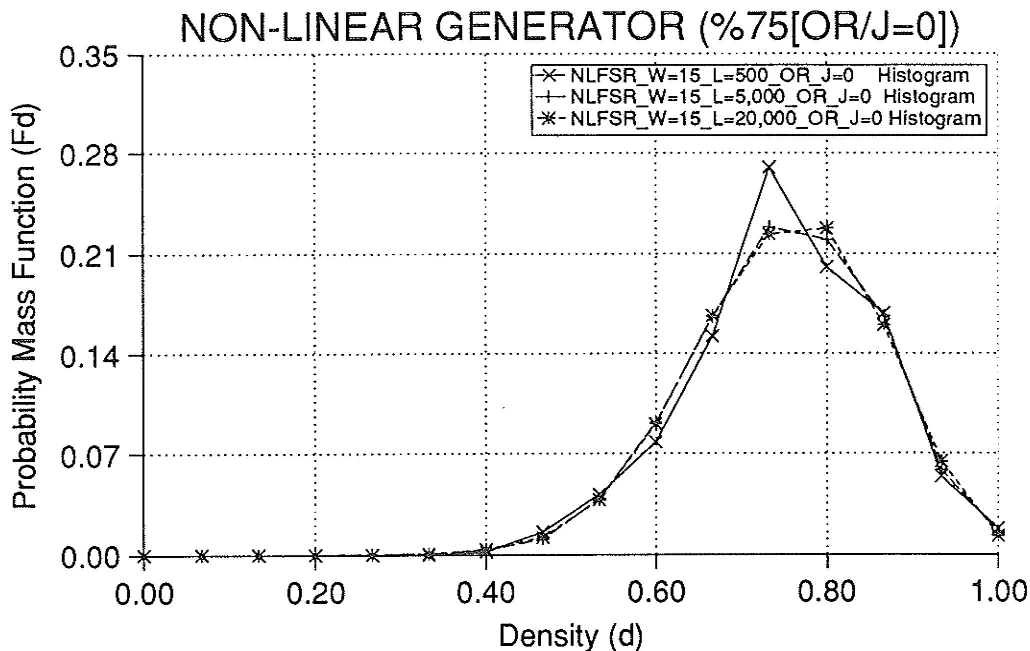

Figure F.12: *Probability Mass Function (Histogram) of NLFSR weighted to 75% (OR-gate bank): [J=0]; W=15.*

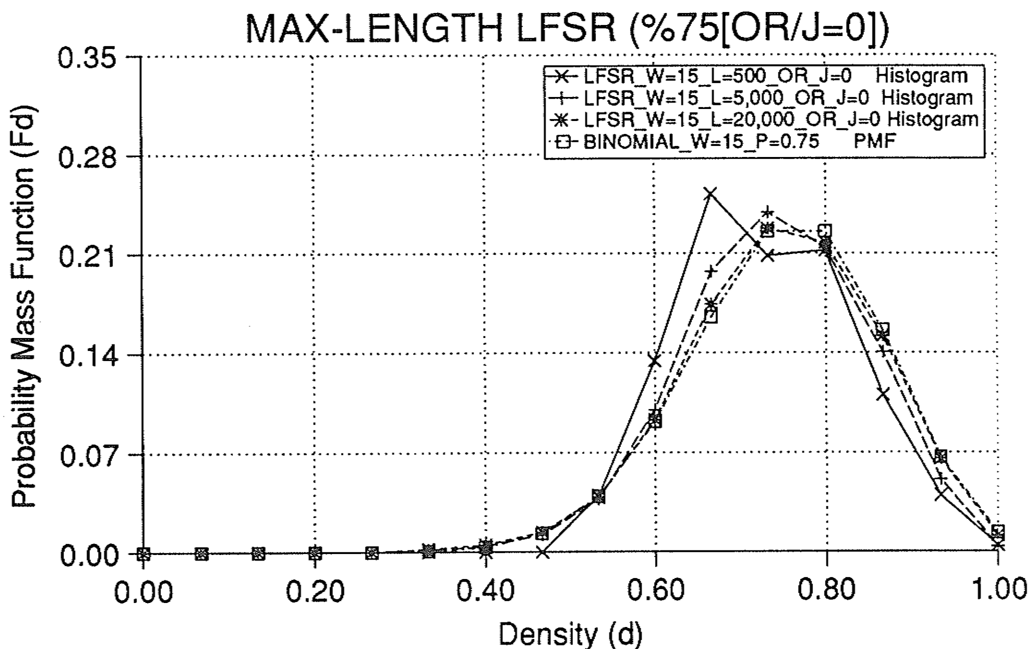

Figure F.13: *Probability Mass Function (Histogram) of LFSR weighted to 75% (OR-gate bank): [J=0]; W=15.*

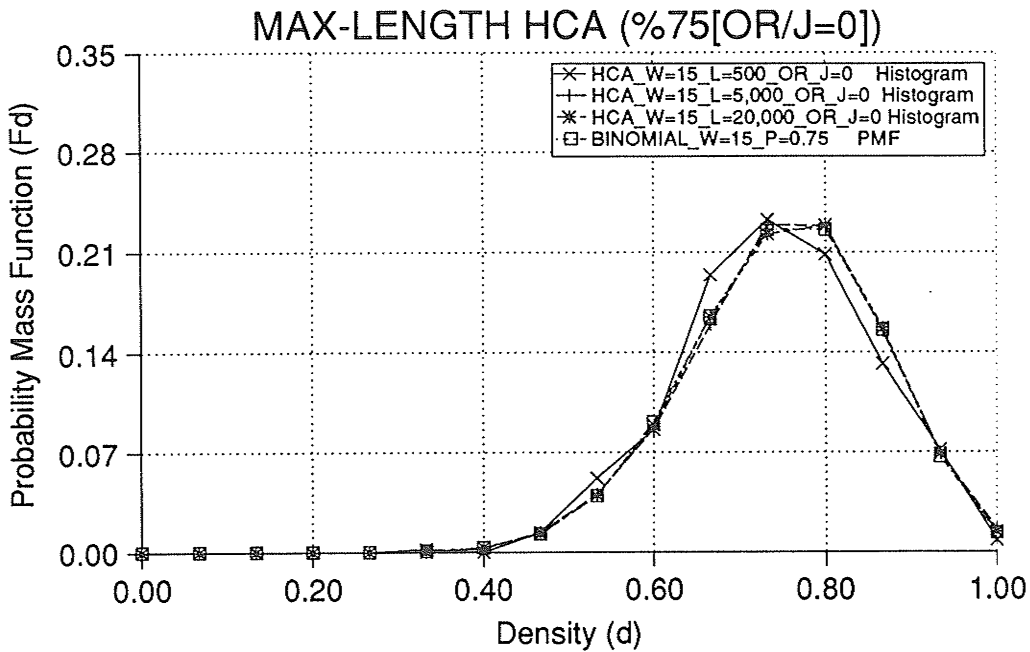

Figure F.14: *Probability Mass Function (Histogram) of Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=0]; W=15.*

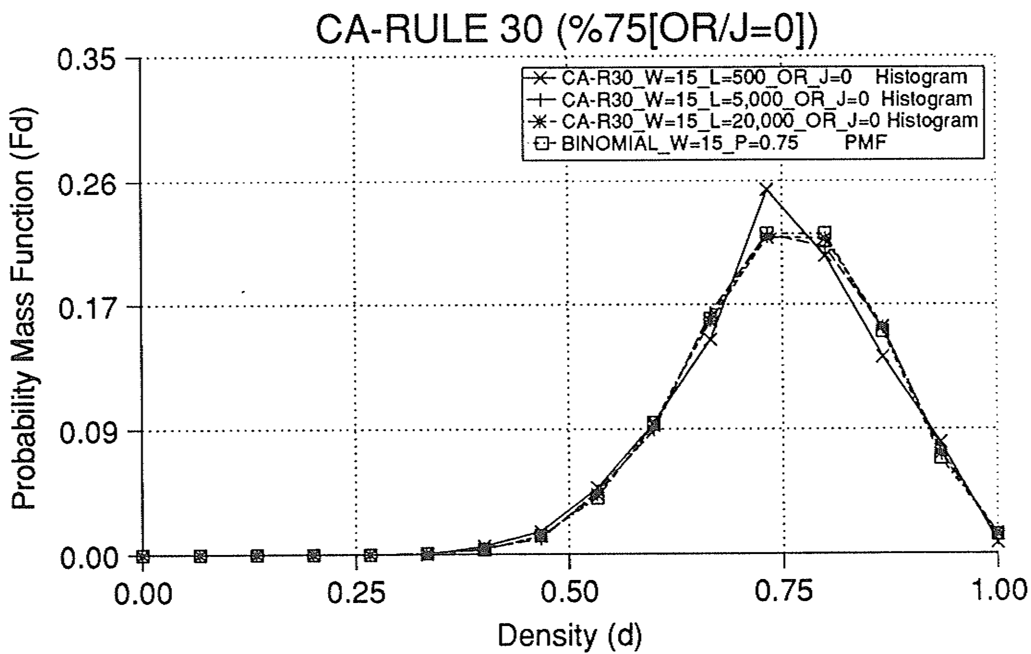

Figure F.15: *Probability Mass Function (Histogram) of Rule 30 CA weighted to 75% (OR-gate bank): [J=0]; W=15.*

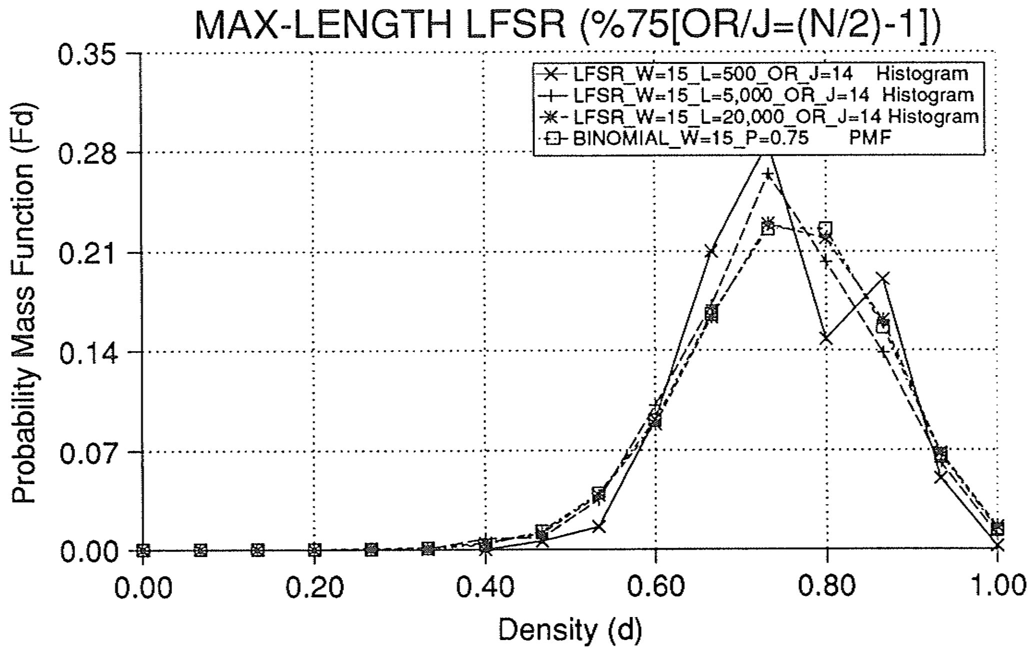

Figure F.16: *Probability Mass Function (Histogram) of LFSR weighted to 75% (OR-gate bank): [J=14]; W=15.*

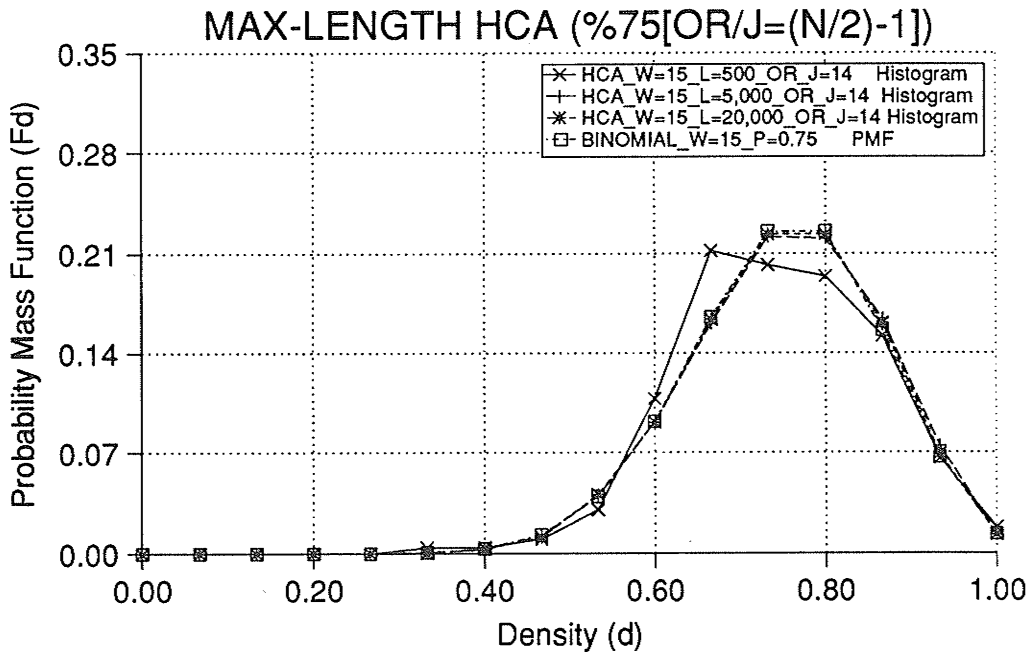

Figure F.17: *Probability Mass Function (Histogram) of Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=14]; W=15.*

Appendix G

Magnitude Spectrums

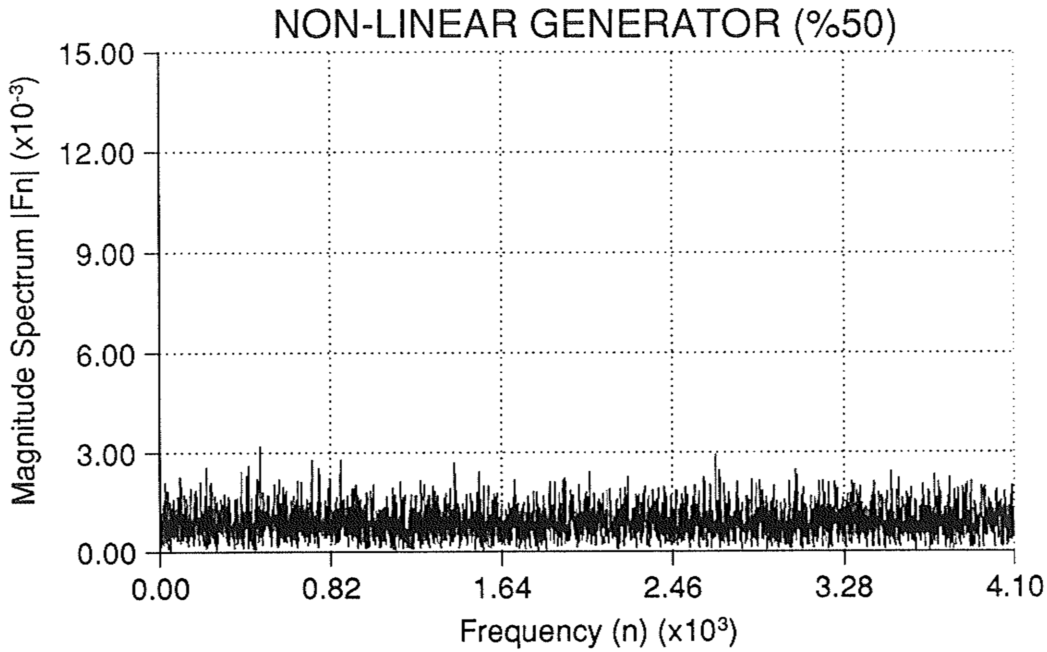

Figure G.1: *Magnitude Spectrum of NLFSR (50%): W=30; L=8,192.*

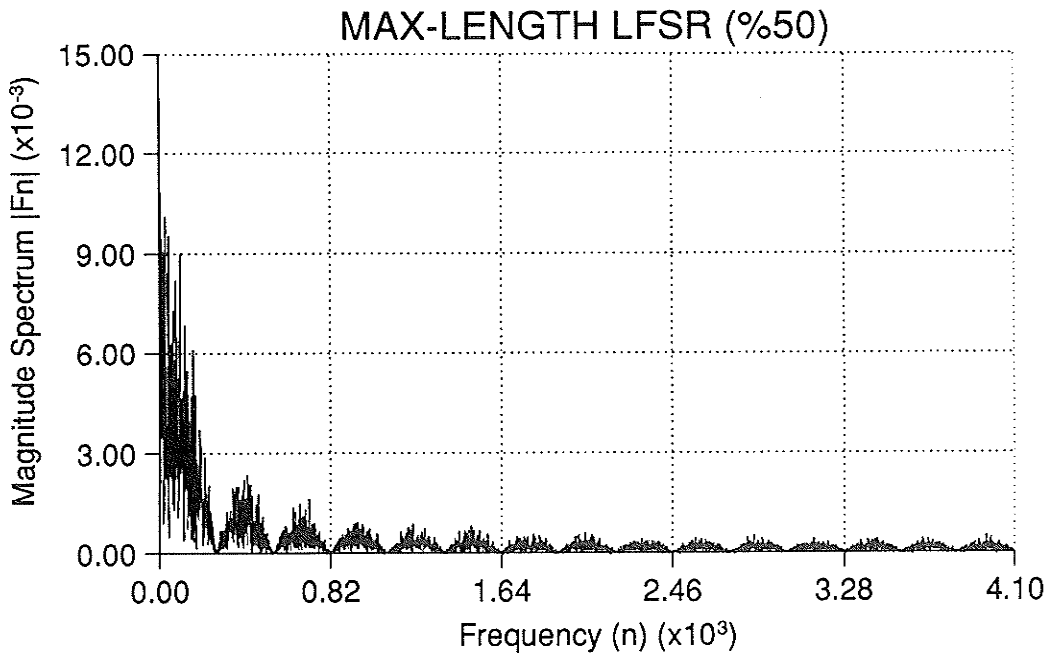

Figure G.2: *Magnitude Spectrum of LFSR (50%): W=30; L=8,192.*

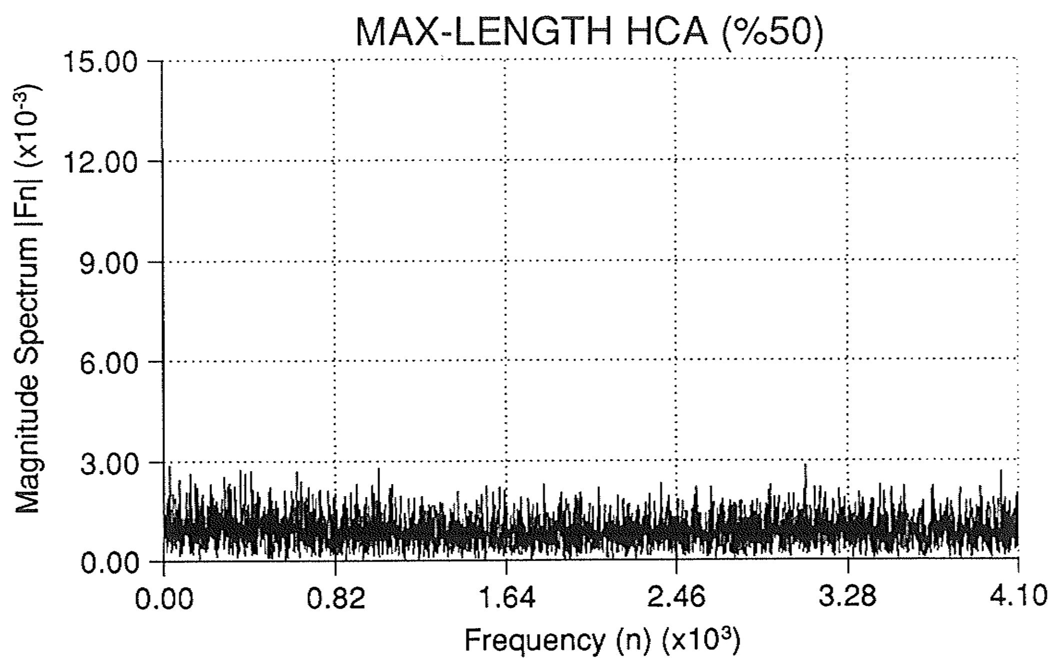

Figure G.3: *Magnitude Spectrum of Rule 90/150 HCA (50%): W=30; L=8,192.*

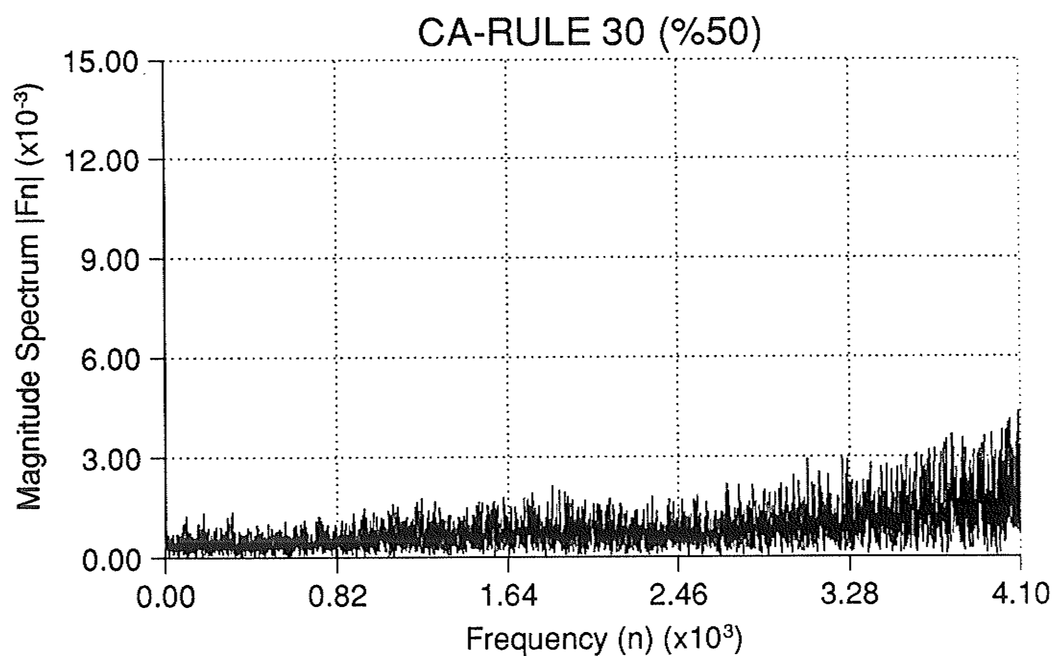

Figure G.4: *Magnitude Spectrum of Rule 30 CA (50%): W=30; L=8,192.*

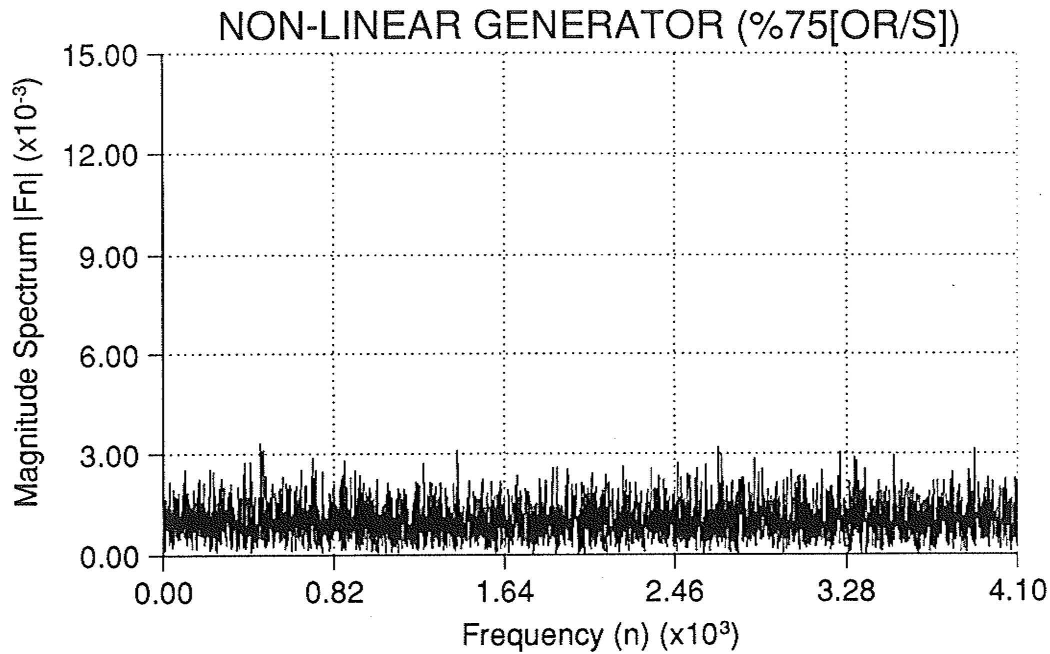

Figure G.5: *Magnitude Spectrum of NLFSR weighted to 75% (OR-gate bank): [S]; W=29; L=8,192.*

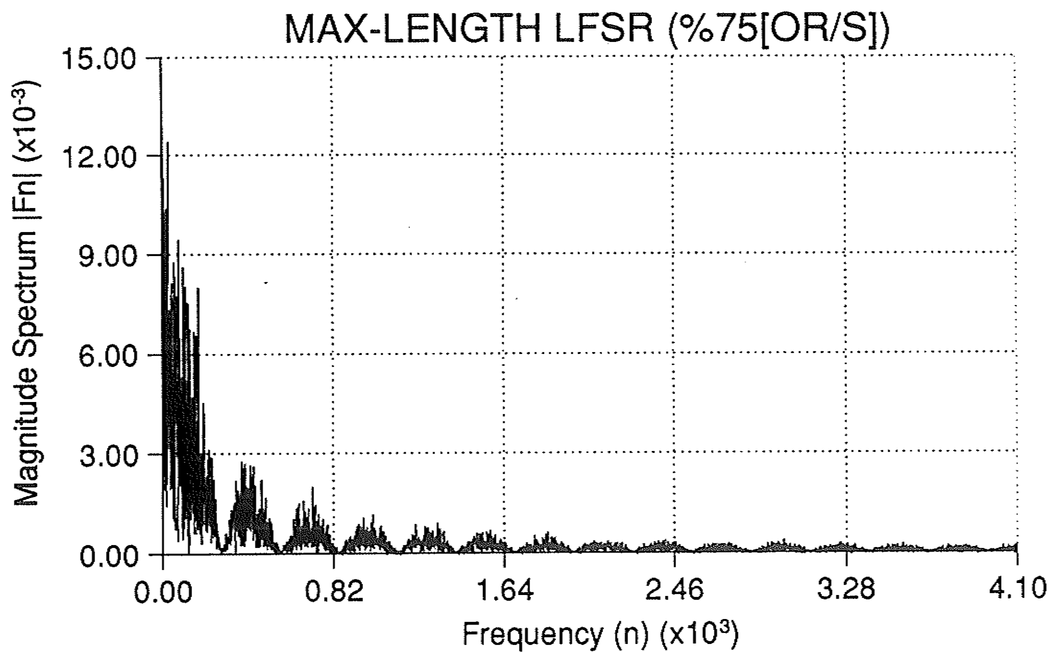

Figure G.6: *Magnitude Spectrum of LFSR weighted to 75% (OR-gate bank): [S]; W=29; L=8,192.*

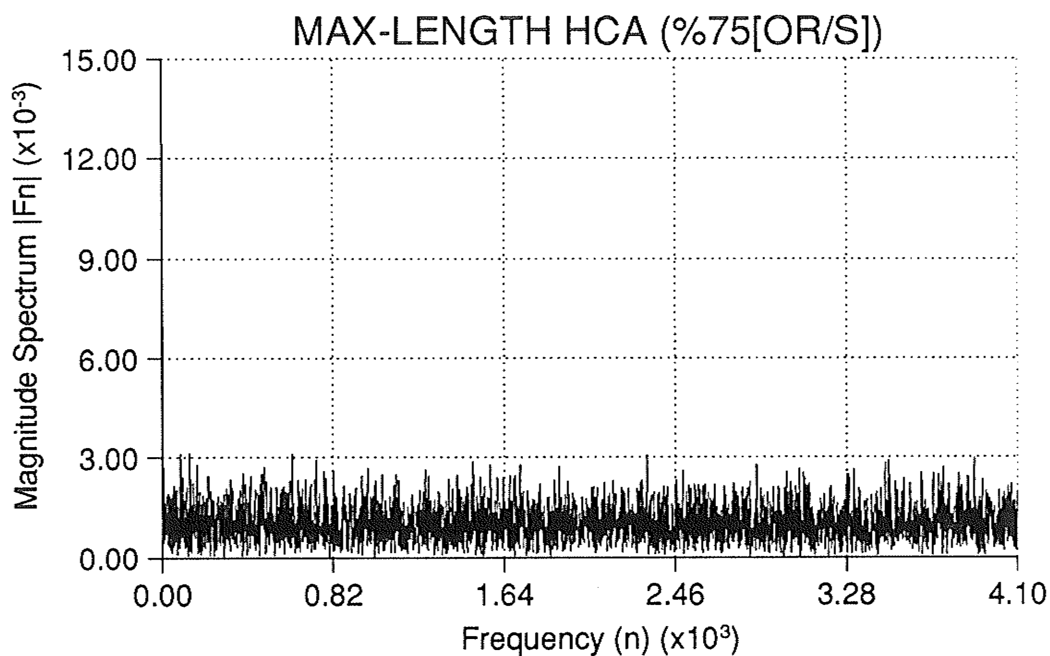

Figure G.7: *Magnitude Spectrum of Rule 90/150 weighted to 75% (OR-gate bank): [S]; W=29; L=8,192.*

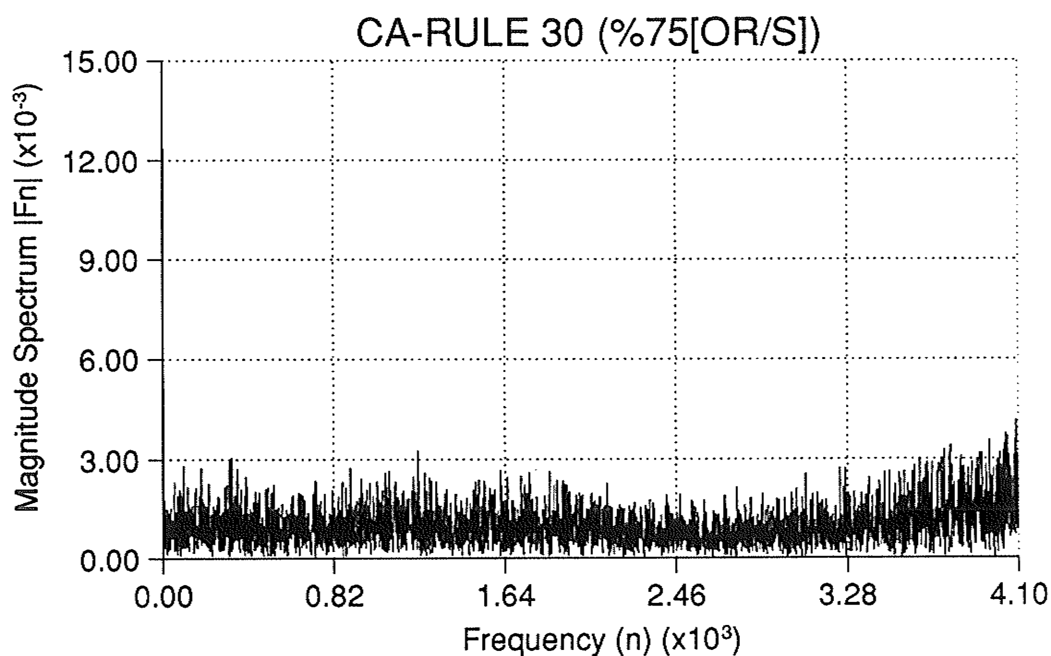

Figure G.8: *Magnitude Spectrum of Rule 30 CA weighted to 75% (OR-gate bank): [S]; W=29; L=8,192.*

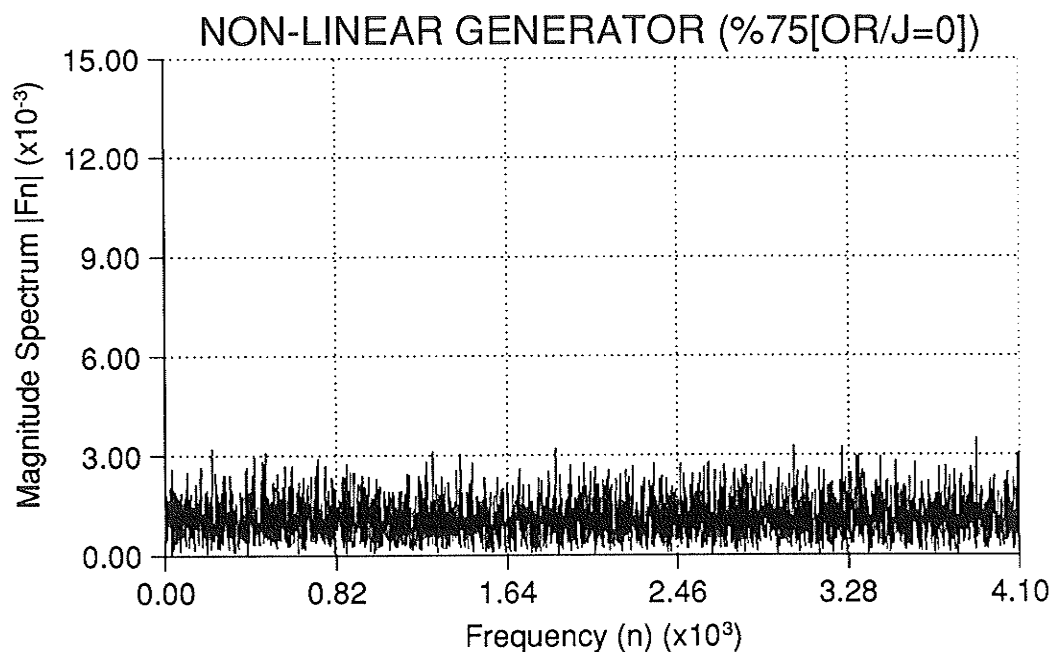

Figure G.9: *Magnitude Spectrum of NLFSR weighted to 75% (OR-gate bank): [J=0]; W=15; L=8,192.*

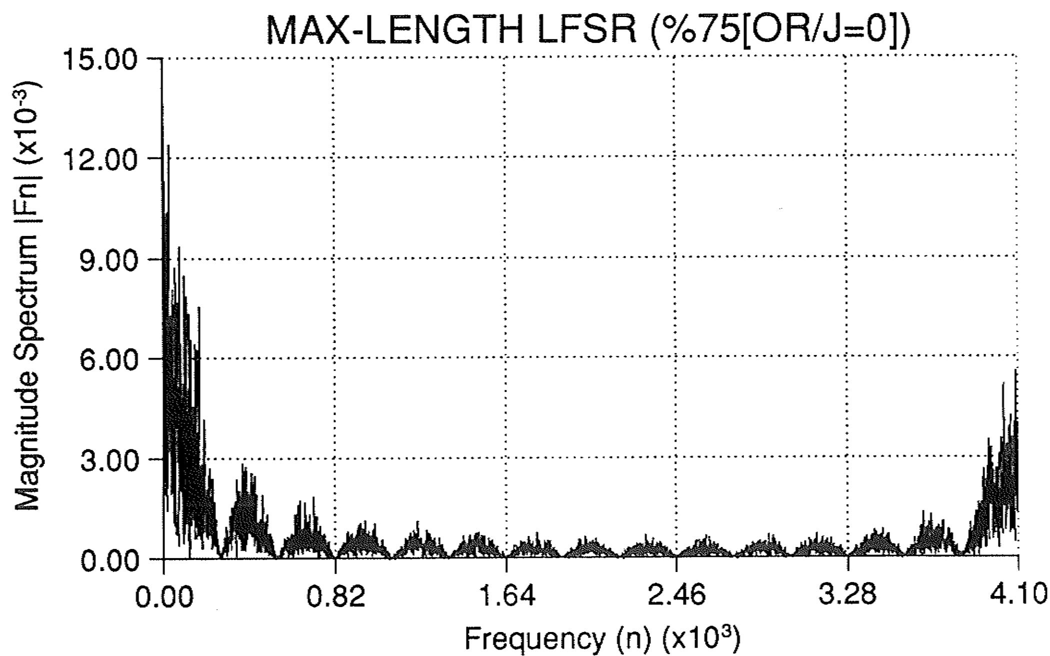

Figure G.10: *Magnitude Spectrum of LFSR weighted to 75% (OR-gate bank): [J=0]; W=15; L=8,192.*

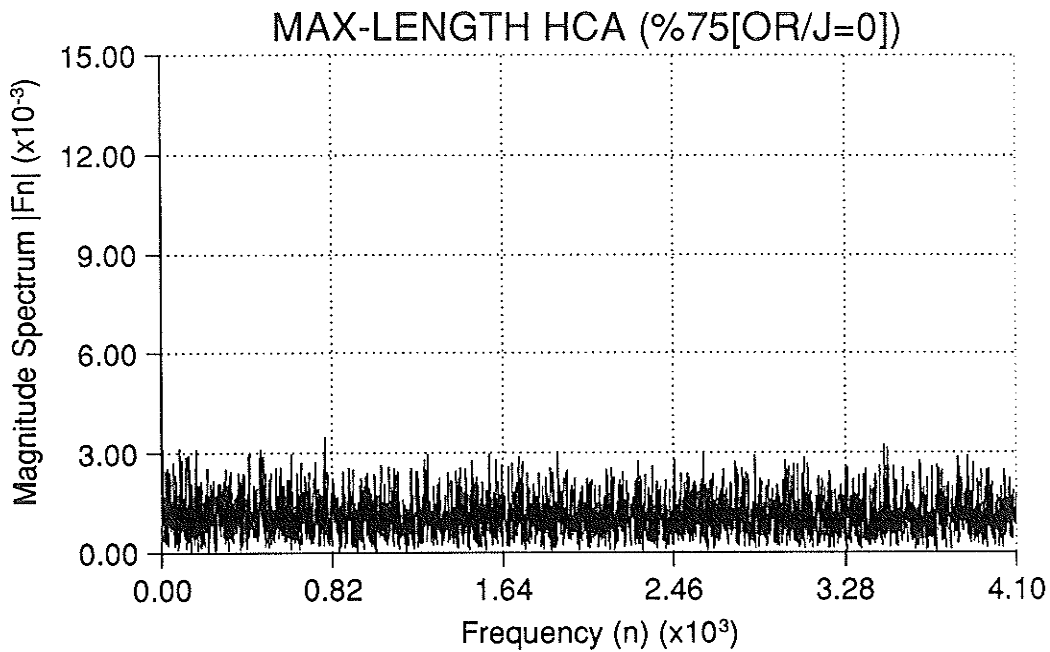

Figure G.11: *Magnitude Spectrum of Rule 90/150 weighted to 75% (OR-gate bank): [J=0]; W=15; L=8,192.*

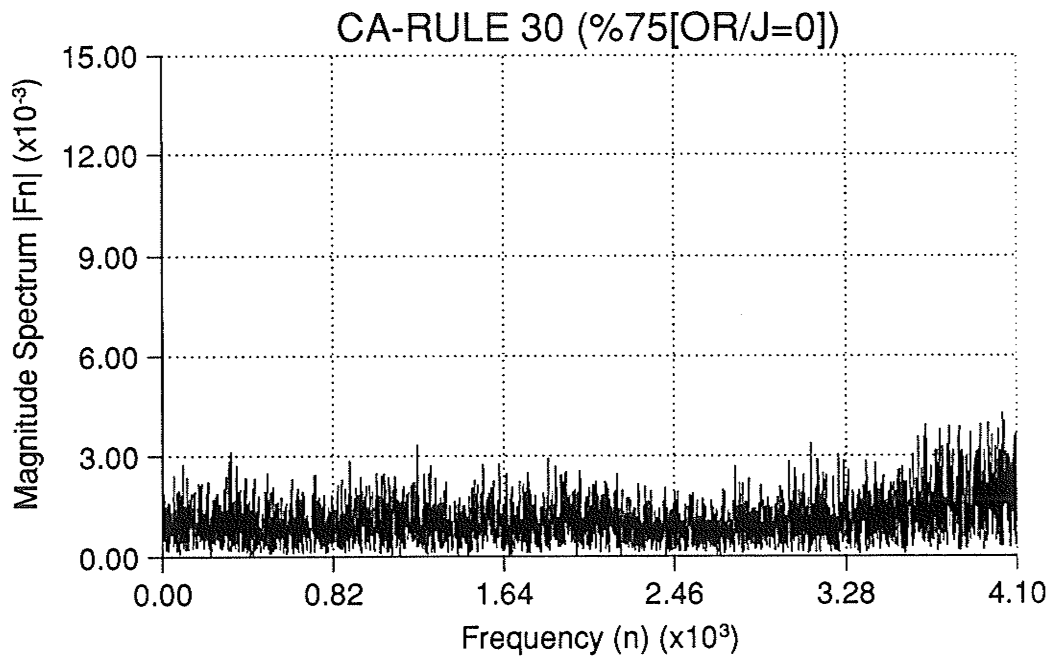

Figure G.12: *Magnitude Spectrum of Rule 30 CA weighted to 75% (OR-gate bank): [J=0]; W=15; L=8,192.*

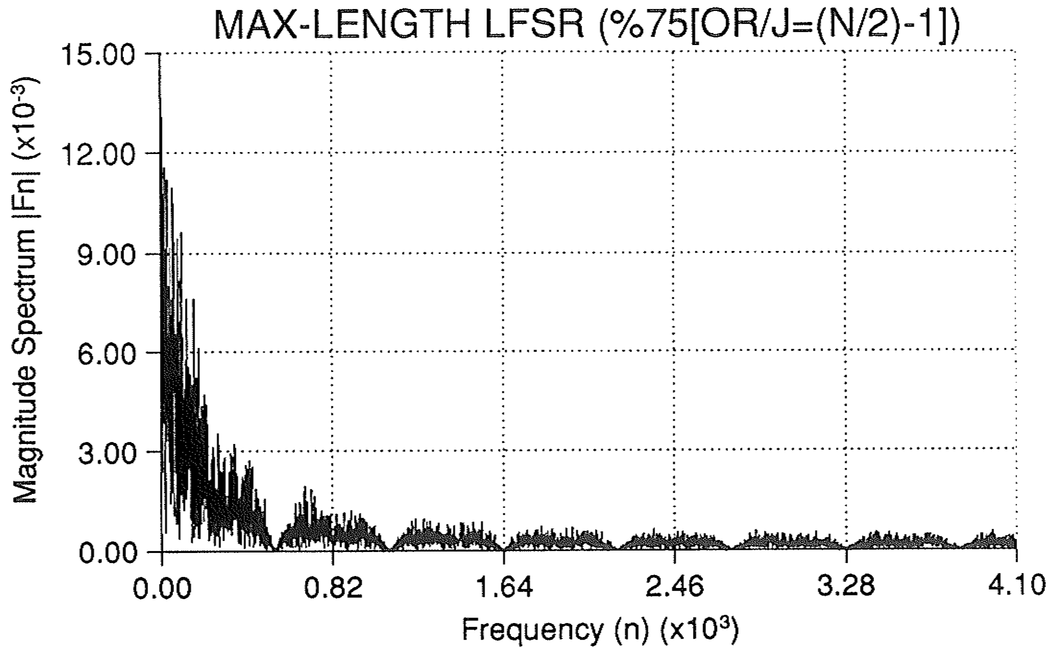

Figure G.13: *Magnitude Spectrum of LFSR weighted to 75% (OR-gate bank): [J=14]; W=15; L=8,192.*

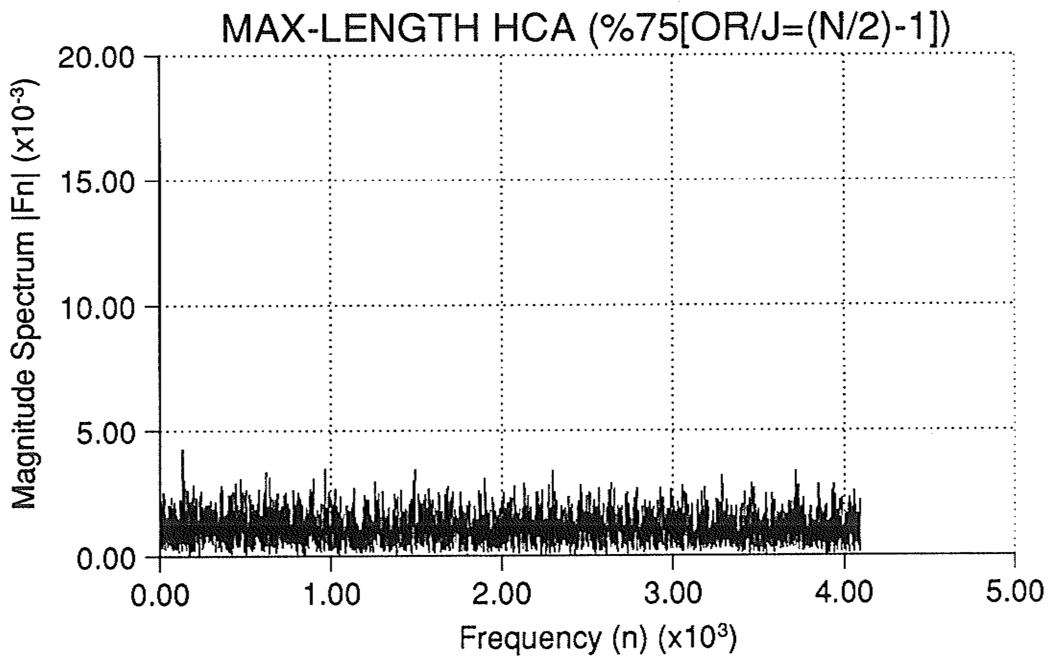

Figure G.14: *Magnitude Spectrum of Rule 90/150 weighted to 75% (OR-gate bank): [J=14]; W=15; L=8,192.*

Appendix H

Space-Phase Correlation Plots

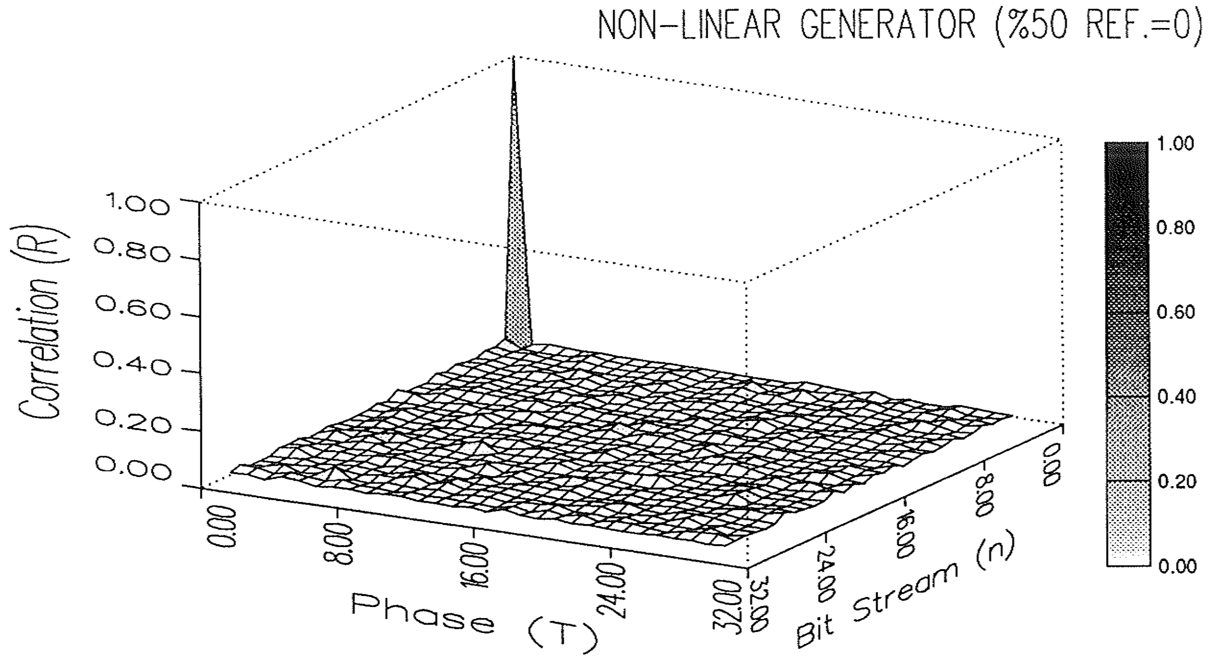

Figure H.1: *Space-phase Cross-correlation of NLFSR (50%): W=30; L=10,000; Reference n=0.*

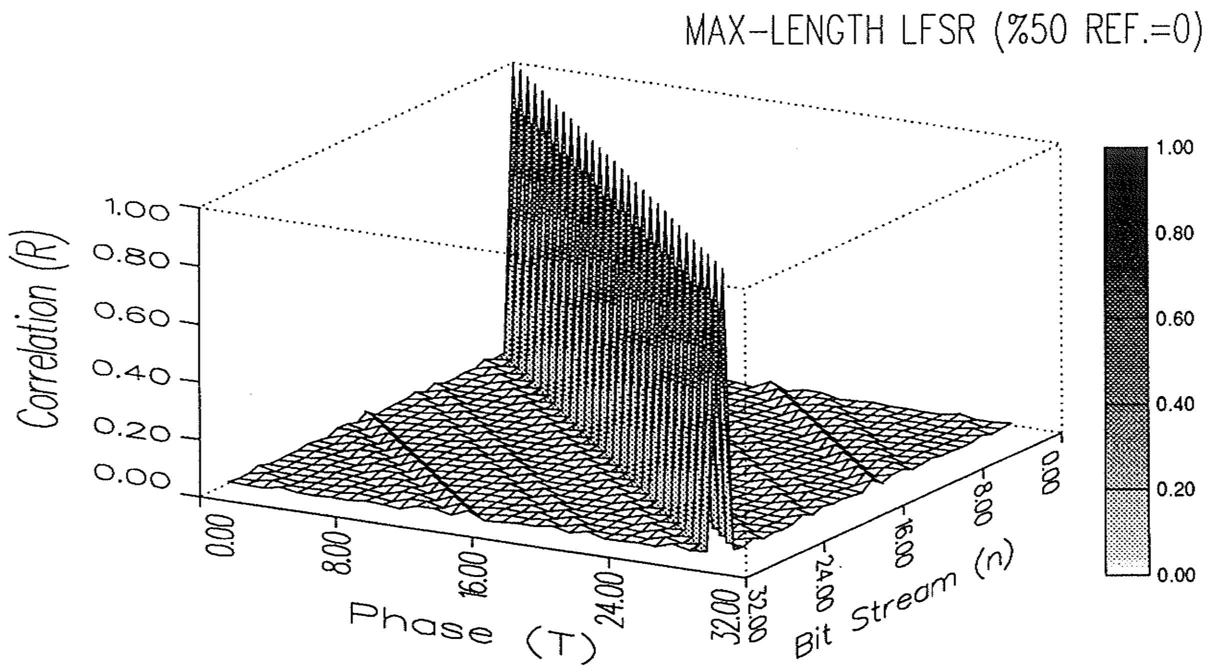

Figure H.2: *Space-phase Cross-correlation of LFSR (50%): W=30; L=10,000; Reference n=0.*

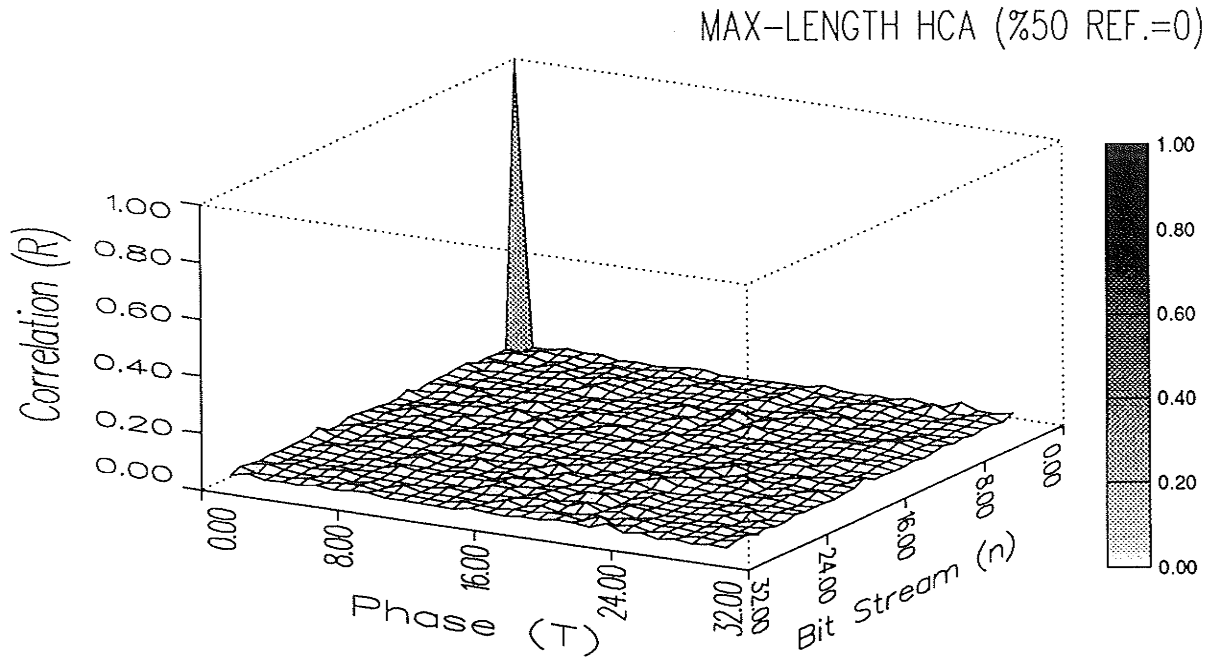

Figure H.3a): *Space-phase Cross-correlation of Rule 90/150 HCA (50%): W=30; L=10,000; Reference n=0.*

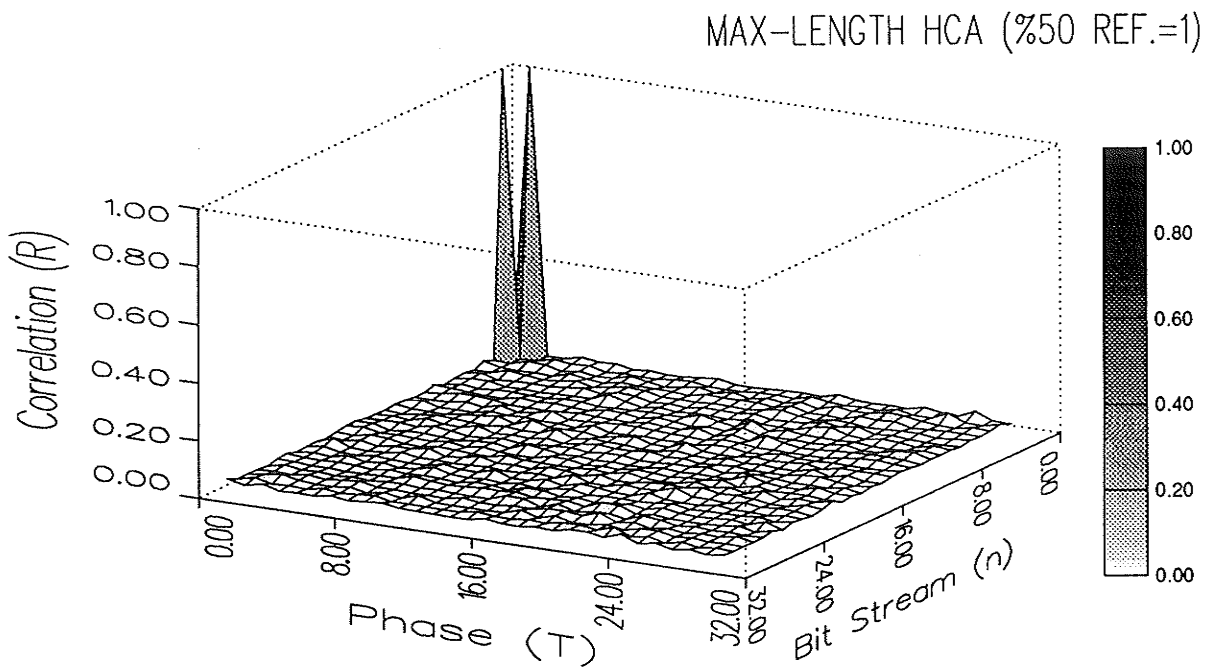

Figure H.3b): *Space-phase Cross-correlation of Rule 90/150 HCA (50%): W=30; L=10,000; Reference n=1.*

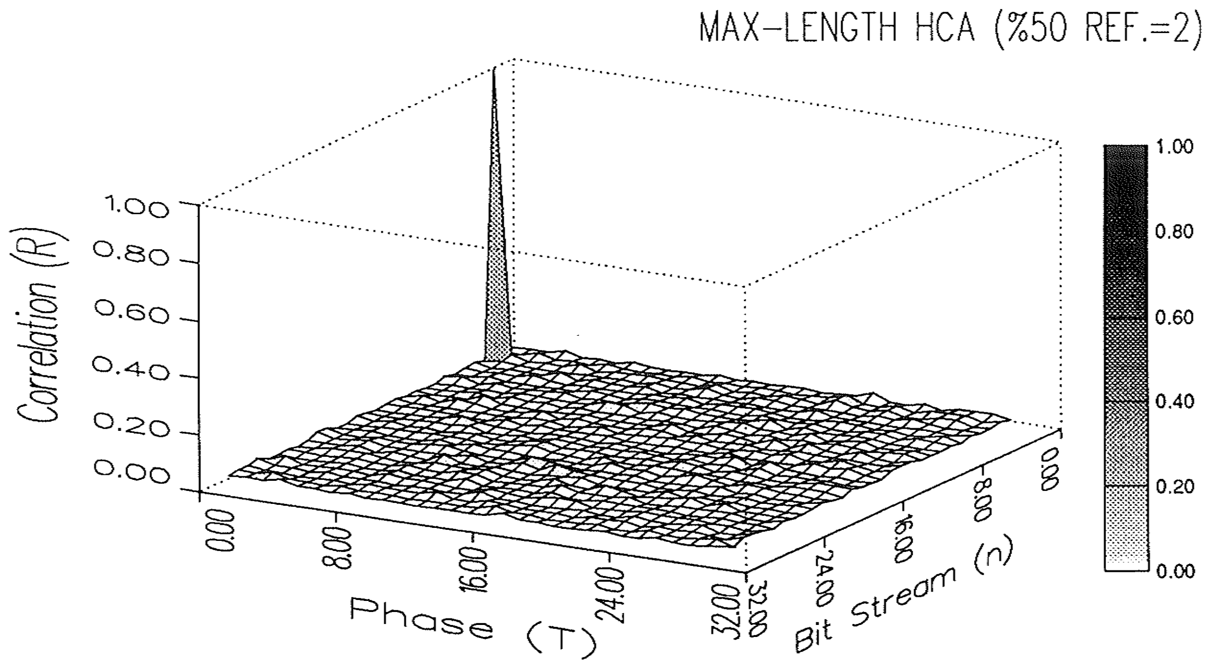

Figure H.3c): *Space-phase Cross-correlation of Rule 90/150 HCA (50%): W=30; L=10,000; Reference n=2.*

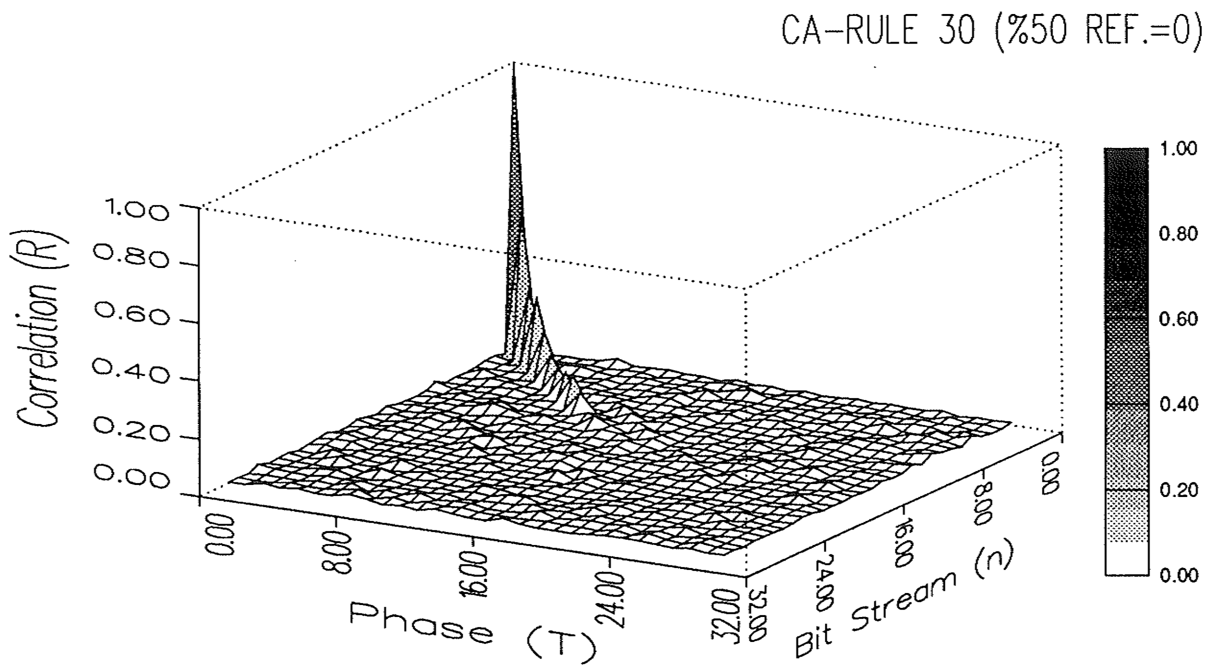

Figure H.4: *Space-phase Cross-correlation of Rule 30 CA (50%): W=30; L=10,000; Reference n=0.*

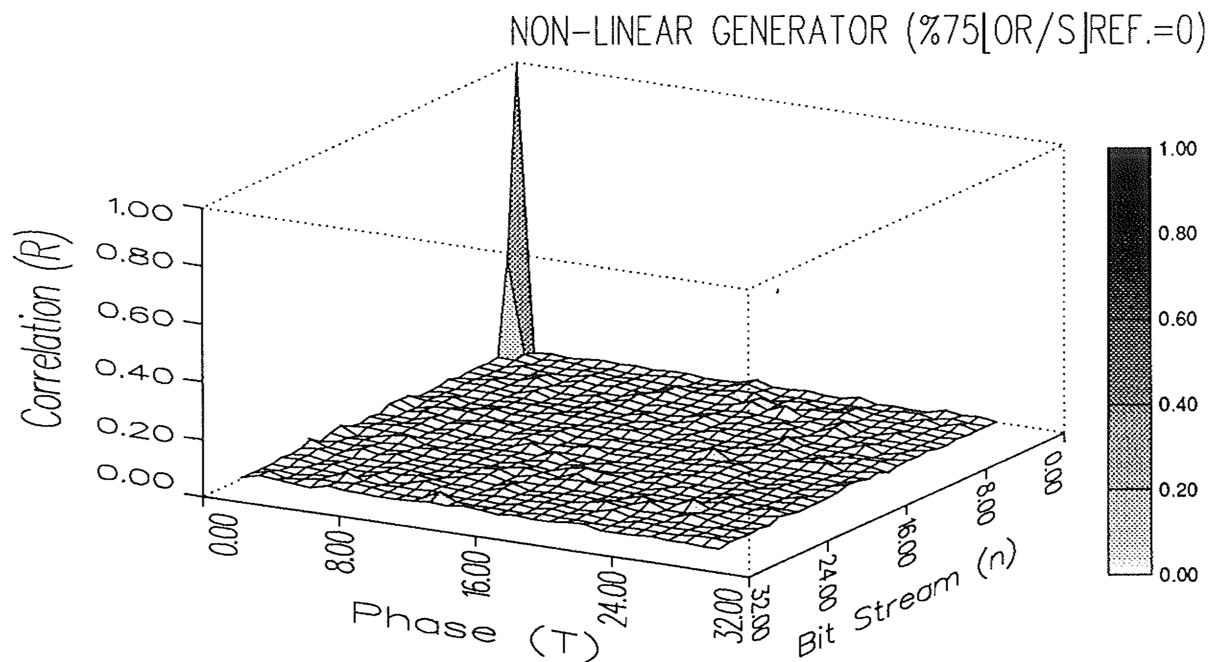

Figure H.5: *Space-phase Cross-correlation of NLFSR weighted to 75% (OR-gate bank): [S]; W=29; L=10,000; Reference n=0.*

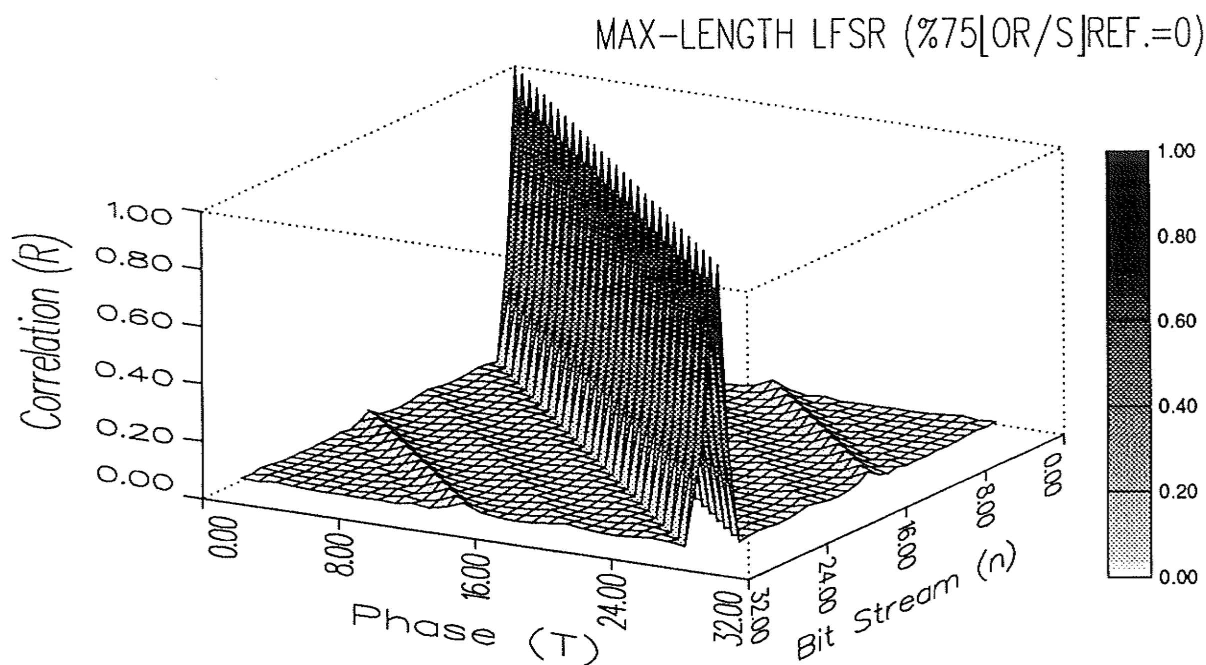

Figure H.6: *Space-phase Cross-correlation of LFSR weighted to 75% (OR-gate bank): [S]; W=29; L=10,000; Reference n=0.*

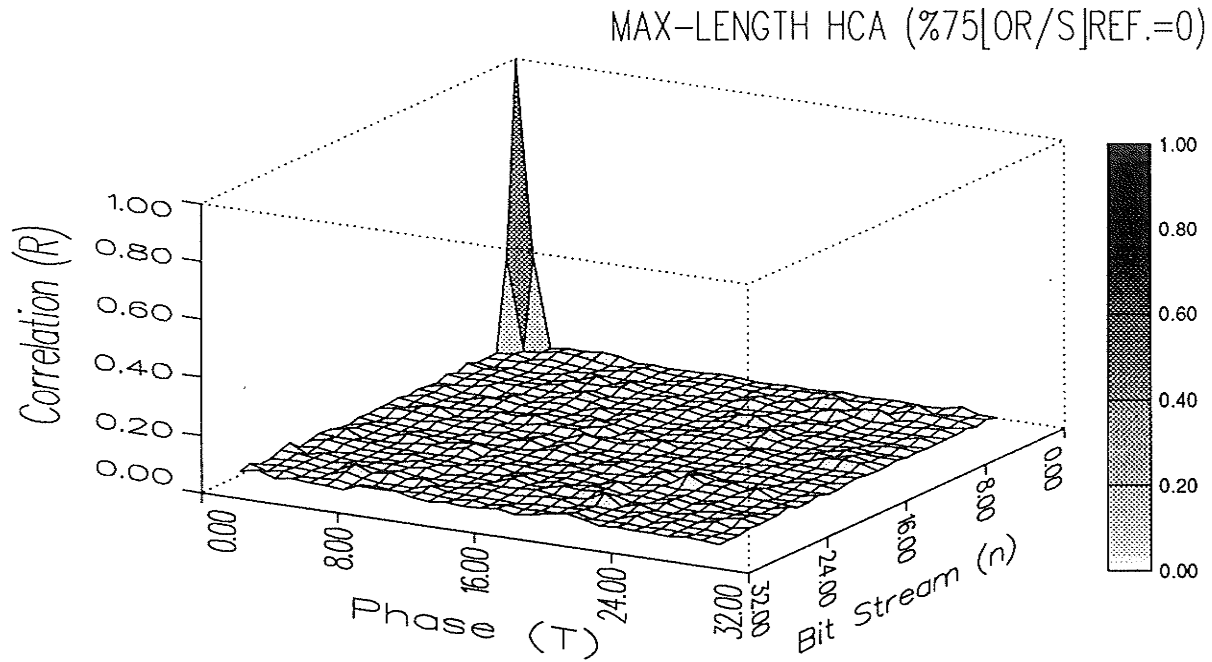

Figure H.7a): *Space-phase Cross-correlation of Rule 90/150 HCA weighted to 75% (OR-gate bank): [S]; W=29; L=10,000; Reference n=0.*

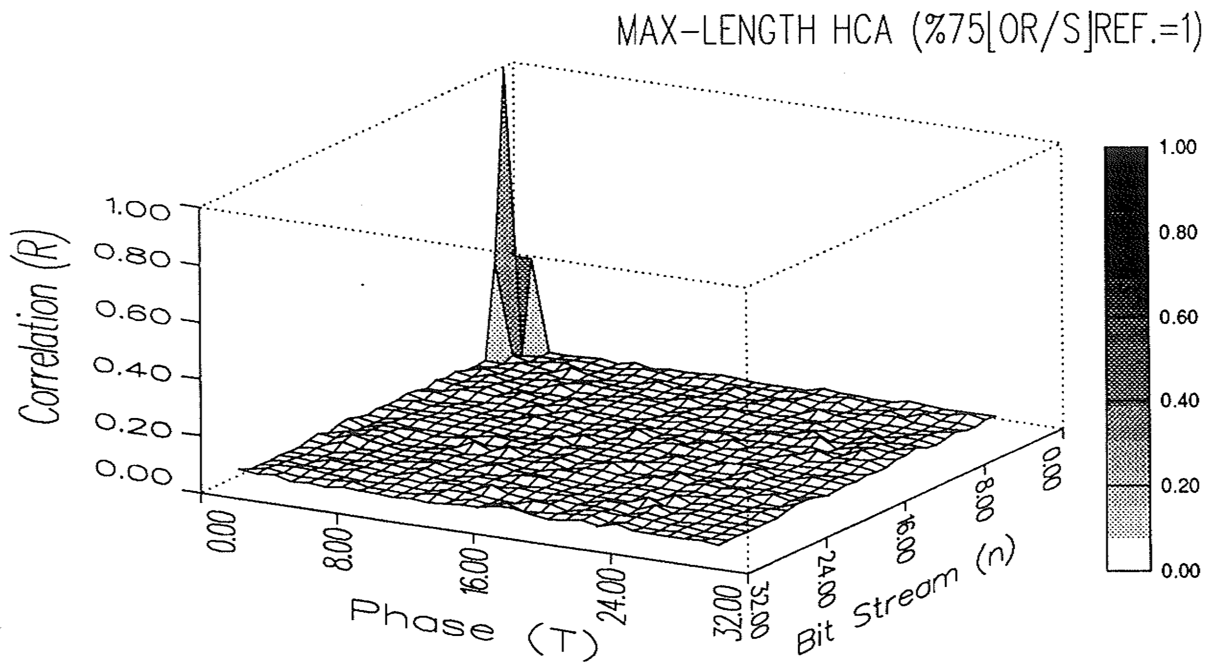

Figure H.7b): *Space-phase Cross-correlation of Rule 90/150 HCA weighted to 75% (OR-gate bank): [S]; W=29; L=10,000; Reference n=1.*

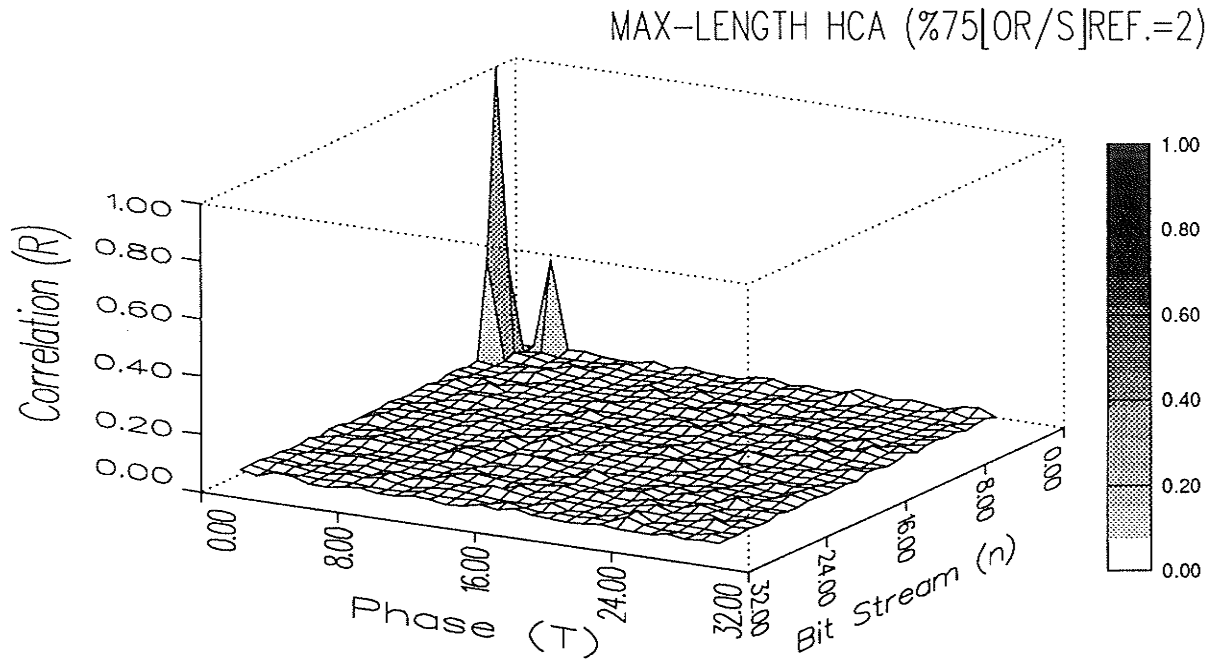

Figure H.7c): *Space-phase Cross-correlation of Rule 90/150 HCA weighted to 75% (OR-gate bank): [S]; W=29; L=10,000; Reference n=2.*

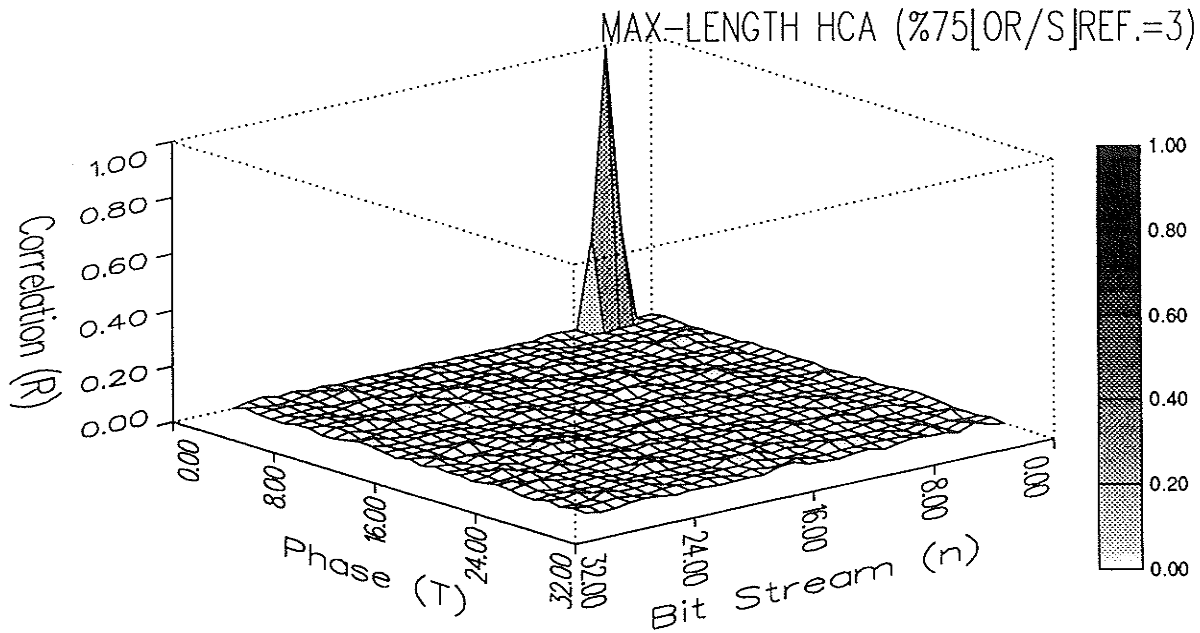

Figure H.7d): *Space-phase Cross-correlation of Rule 90/150 HCA weighted to 75% (OR-gate bank): [S]; W=29; L=10,000; Reference n=3.*

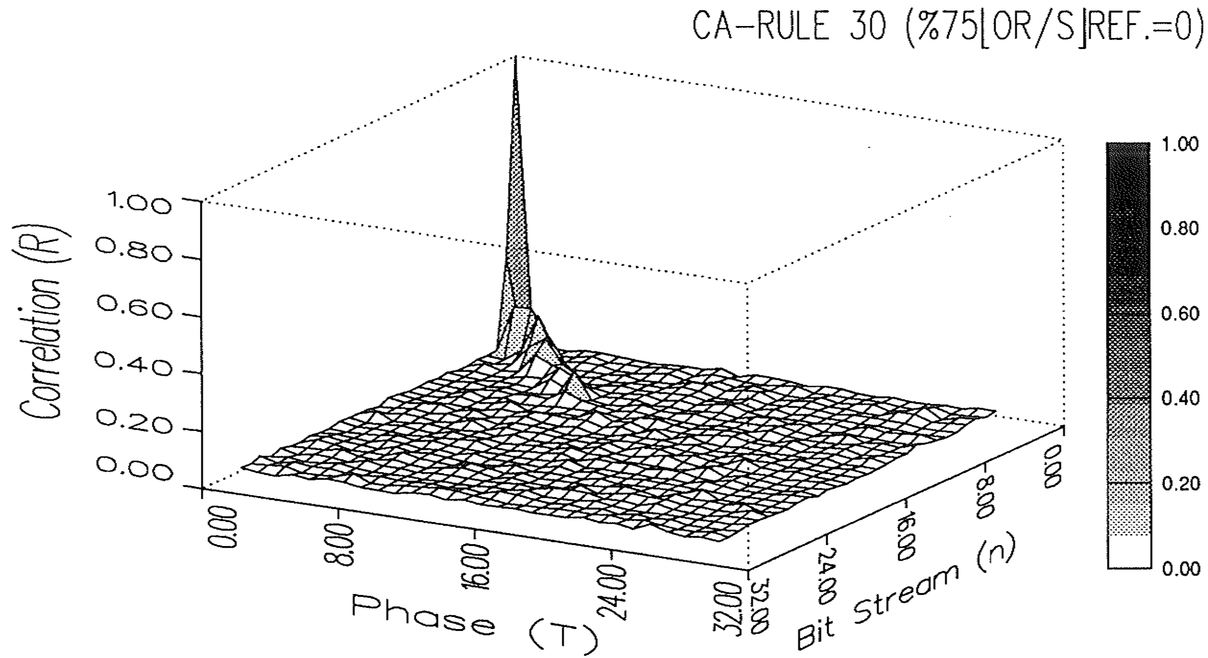

Figure H.8: *Space-phase Cross-correlation of Rule 30 CA weighted to 75% (OR-gate bank): [S]; W=29; L=10,000; Reference n=0.*

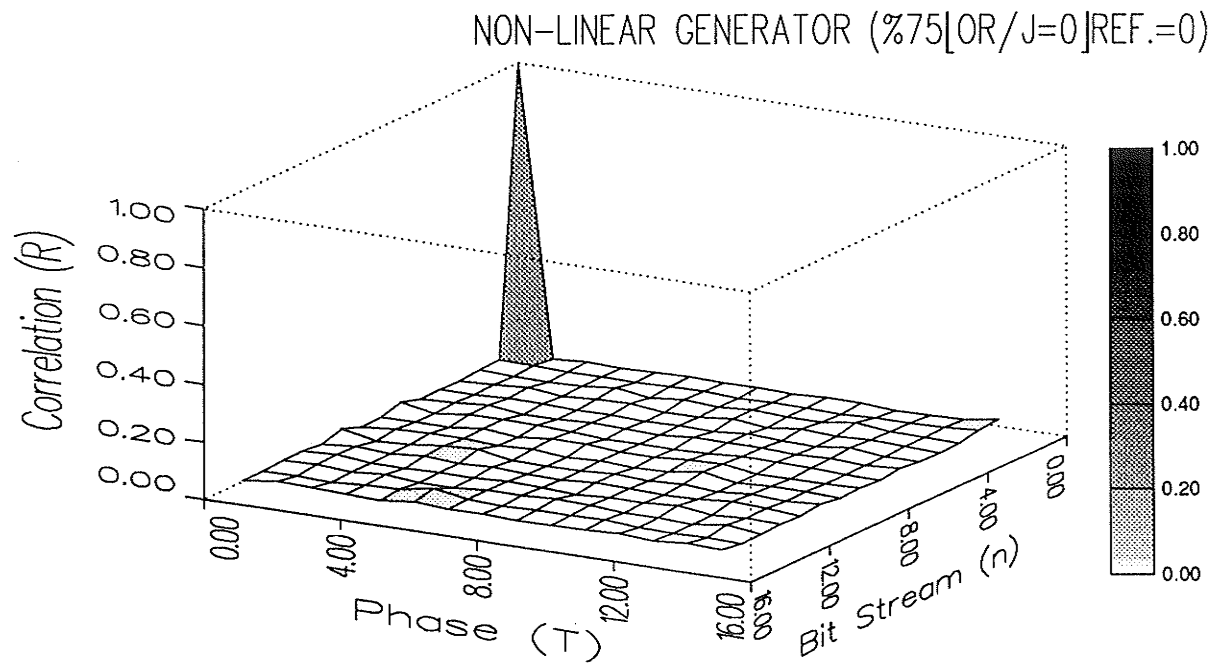

Figure H.9: *Space-phase Cross-correlation of NLFSR weighted to 75% (OR-gate bank): [J=0]; W=15; L=10,000; Reference n=0.*

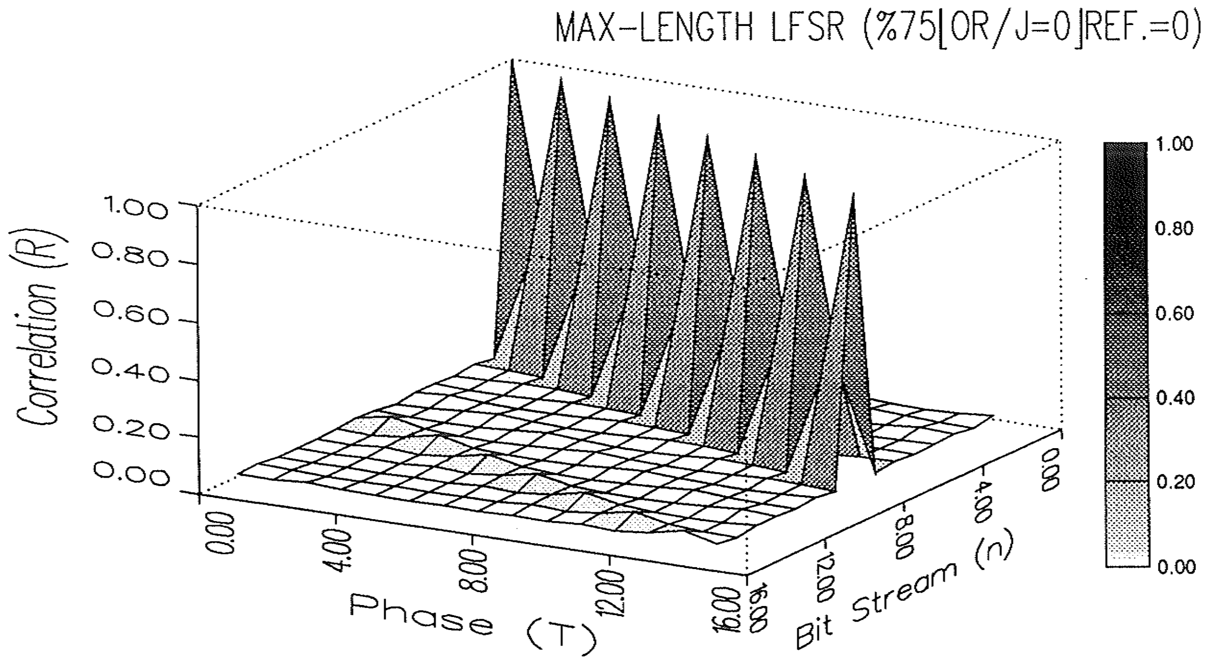

Figure H.10: *Space-phase Cross-correlation of LFSR weighted to 75% (OR-gate bank): [J=0]; W=15; L=10,000; Reference n=0.*

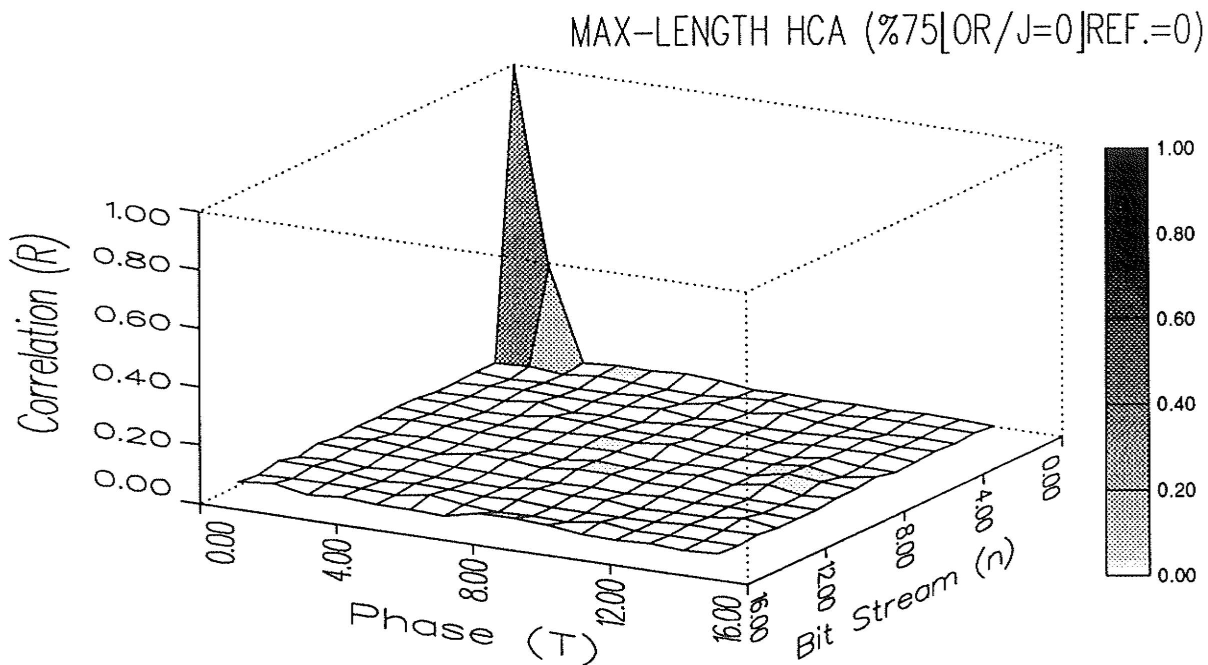

Figure H.11a): *Space-phase Cross-correlation of Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=0]; W=15; L=10,000; Reference n=0.*

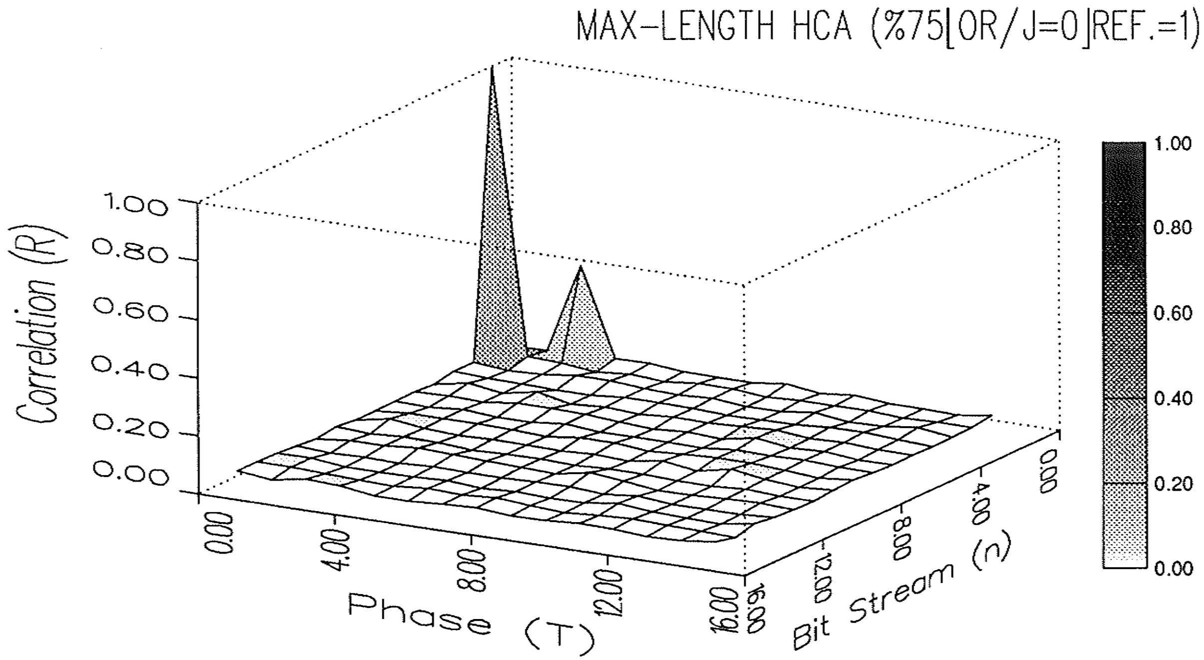

Figure H.11b): *Space-phase Cross-correlation of Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=0]; W=15; L=10,000; Reference n=1.*

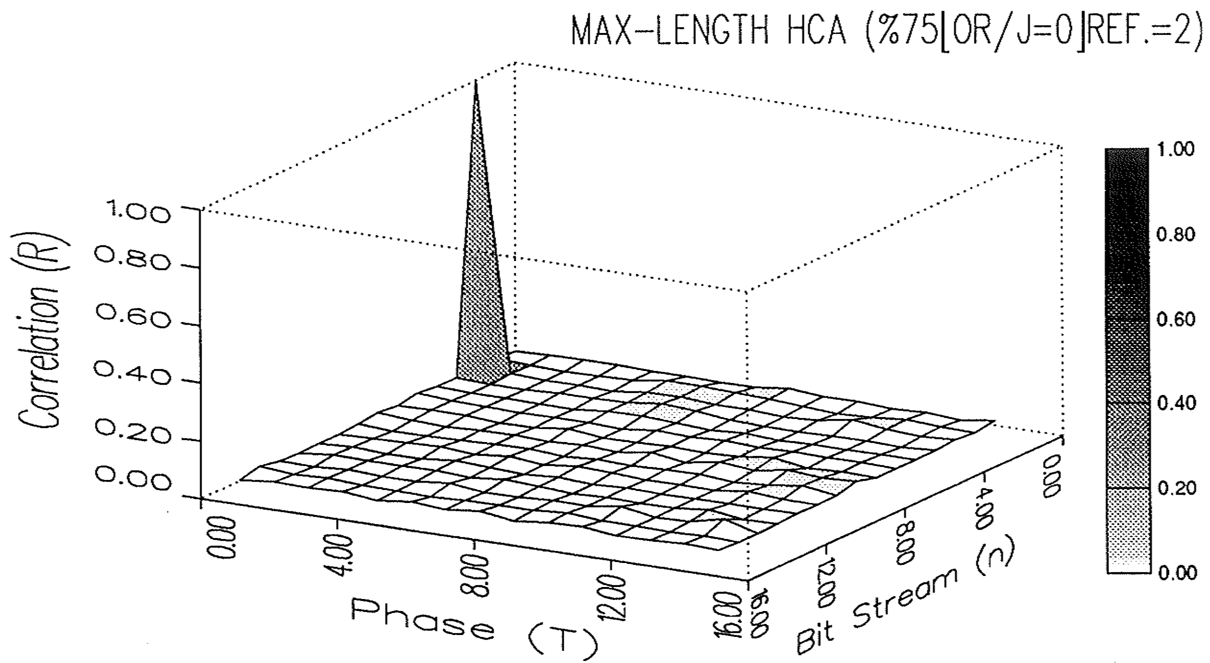

Figure H.11c): *Space-phase Cross-correlation of Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=0]; W=15; L=10,000; Reference n=2.*

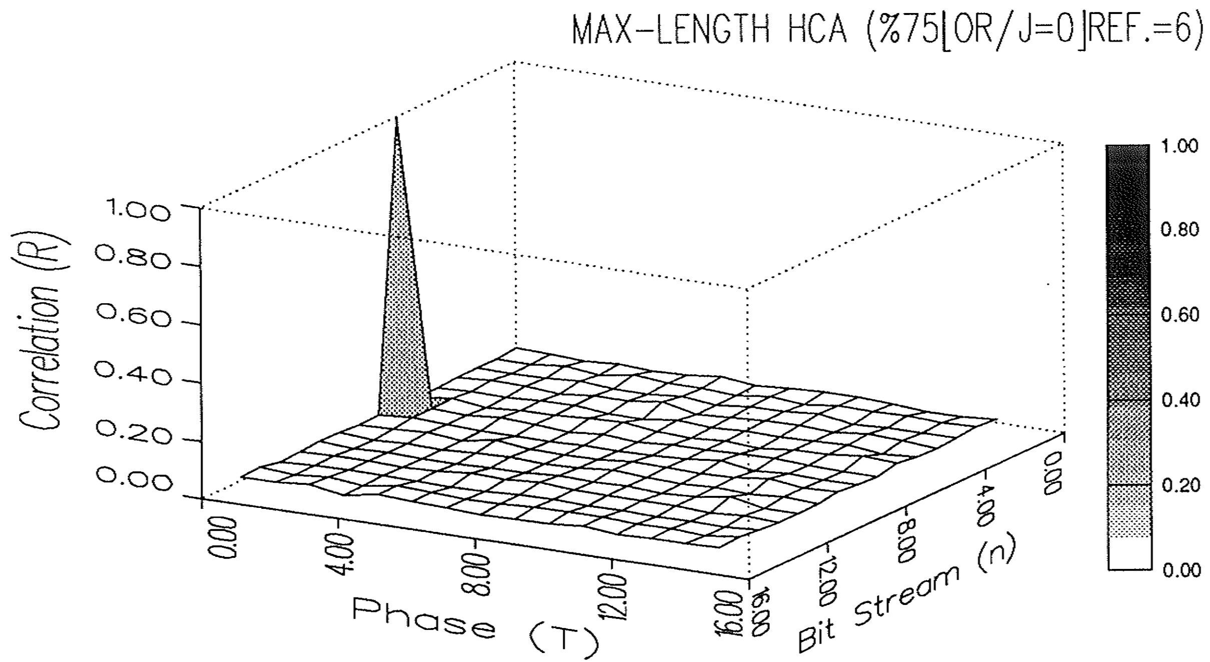

Figure H.11d): *Space-phase Cross-correlation of Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=0]; W=15; L=10,000; Reference n=6.*

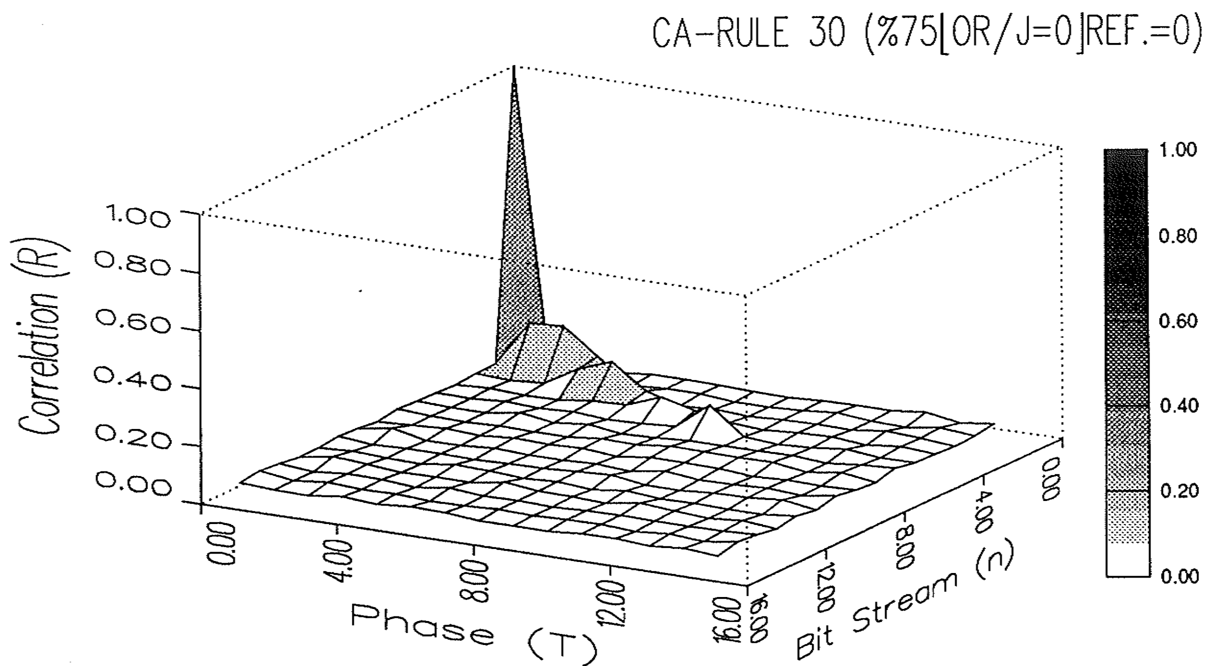

Figure H.12: *Space-phase Cross-correlation of Rule 30 CA weighted to 75% (OR-gate bank): [J=0]; W=15; L=10,000; Reference n=0.*

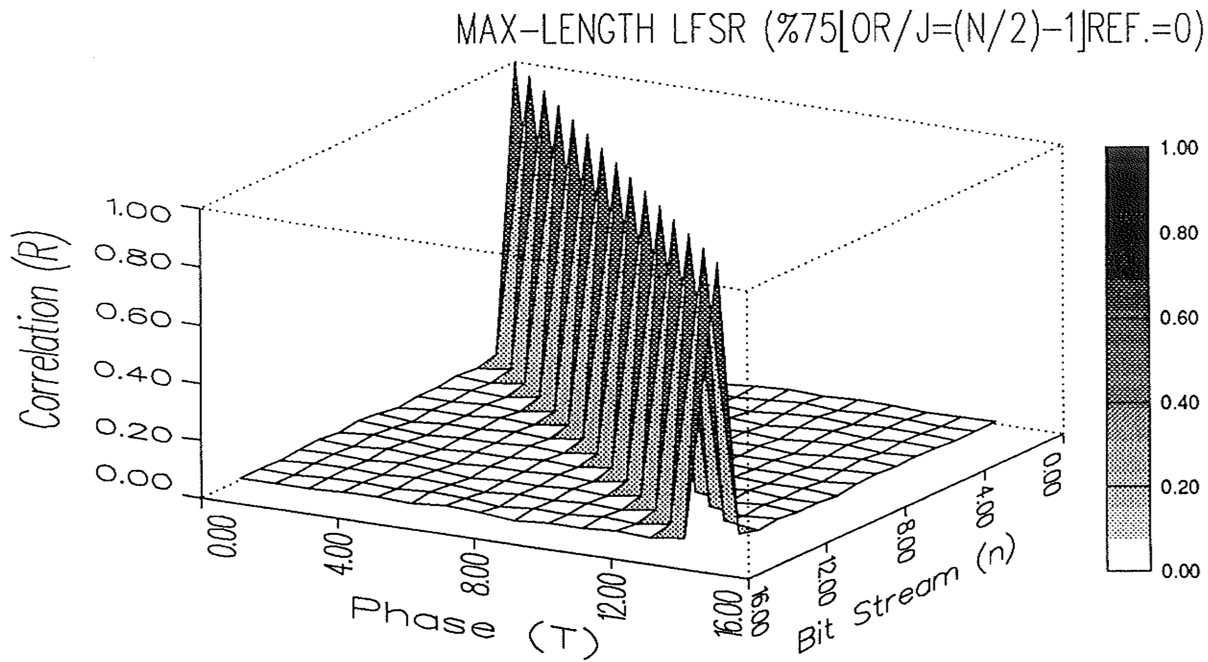

Figure H.13a): *Space-phase Cross-correlation of LFSR weighted to 75% (OR-gate bank): [J=14]; W=15; L=10,000; Reference n=0.*

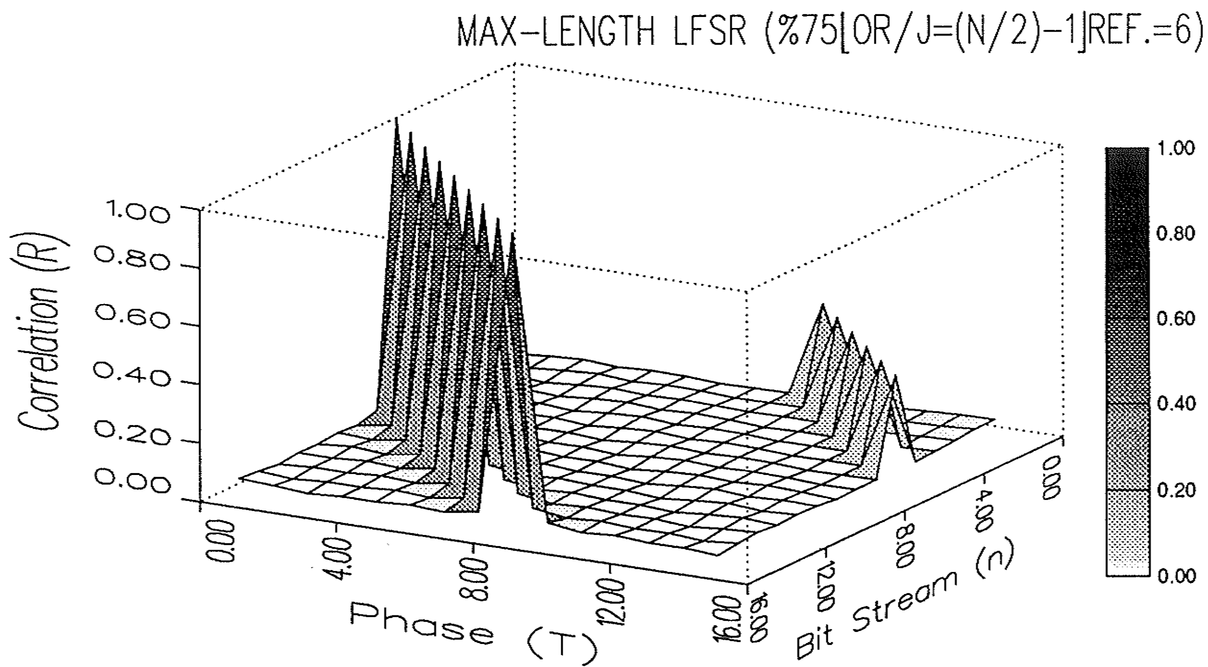

Figure H.13b): *Space-phase Cross-correlation of LFSR weighted to 75% (OR-gate bank): [J=14]; W=15; L=10,000; Reference n=6.*

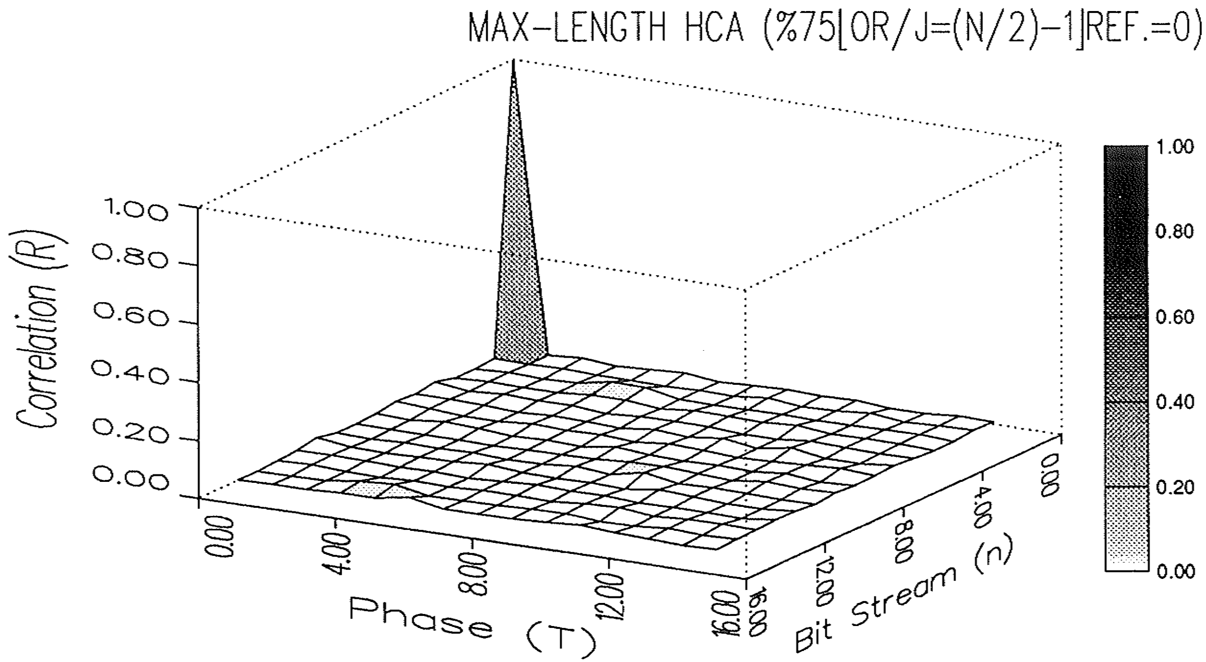

Figure H.14a): *Space-phase Cross-correlation of Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=14]; W=15; L=10,000; Reference n=0.*

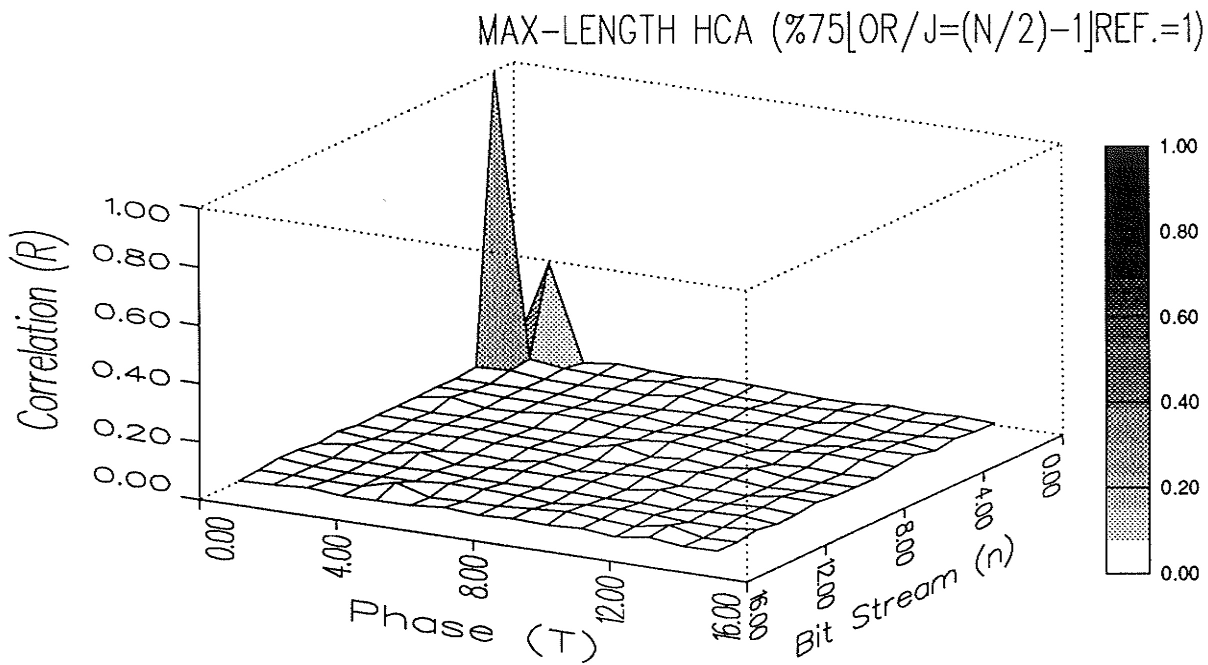

Figure H.14b): *Space-phase Cross-correlation of Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=14]; W=15; L=10,000; Reference n=1.*

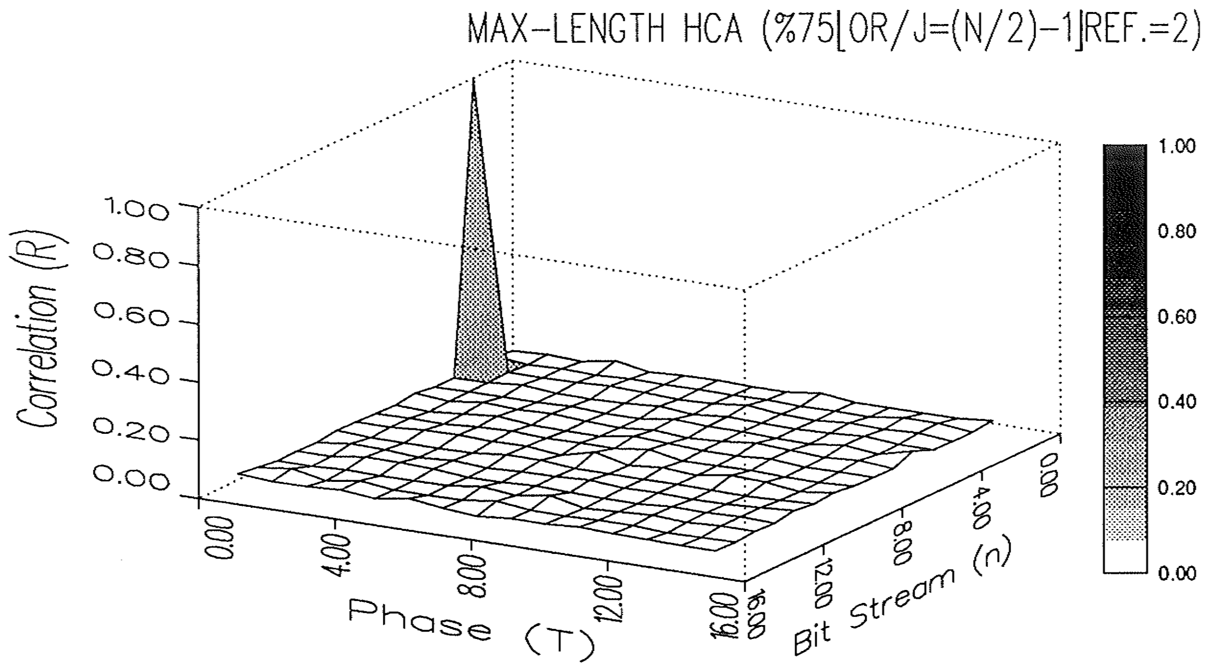

Figure H.14c): *Space-phase Cross-correlation of Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=14]; W=15; L=10,000; Reference n=2.*

Appendix I

Bit Sequence Tuple Profiles

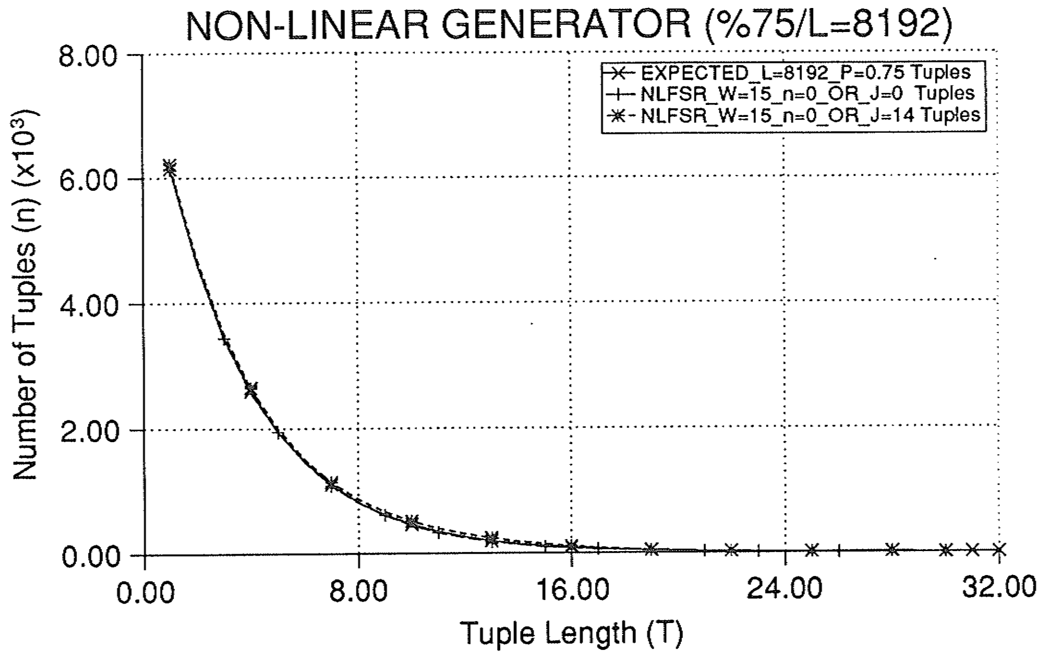

Figure I.1: *Bit Sequence Tuple Profile of NLFSR weighted to 75% (OR-gate bank): [J=0], [J=14]; W=15; L=8,192; Reference n=0.*

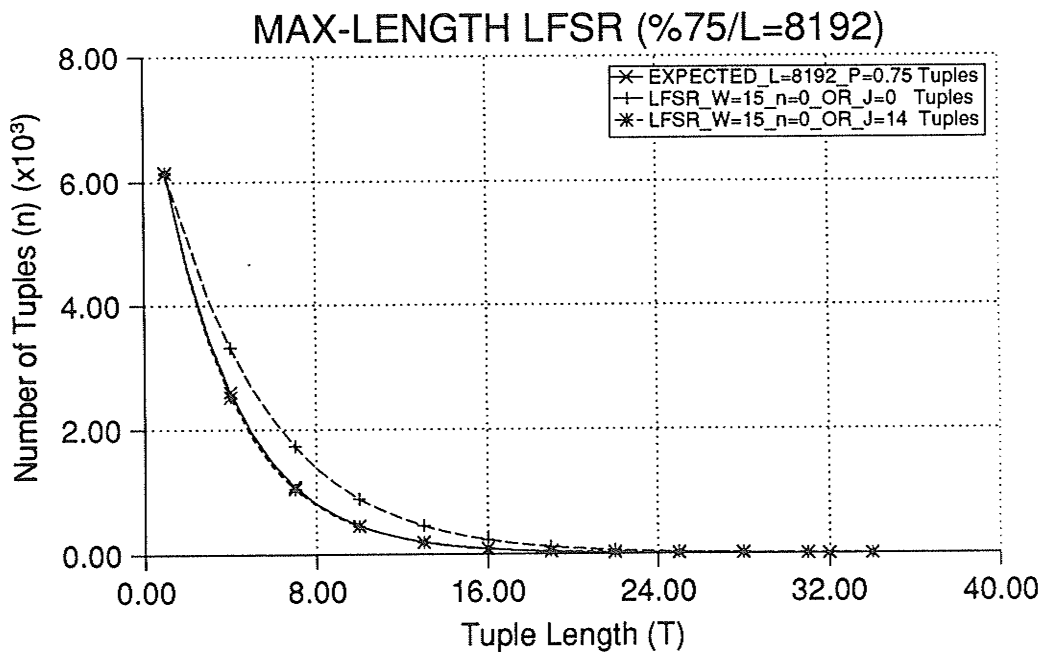

Figure I.2: *Bit Sequence Tuple Profile of LFSR weighted to 75% (OR-gate bank): [J=0], [J=14]; W=15; L=8,192; Reference n=0.*

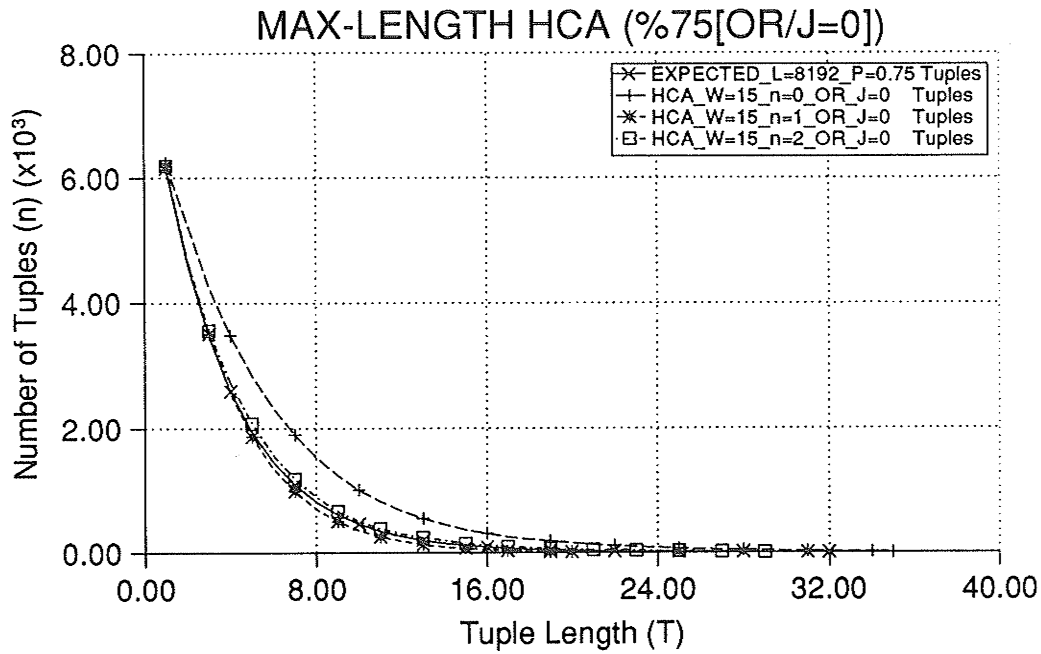

Figure I.3a): *Bit Sequence Tuple Profile of Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=0]; W=15; L=8,192; Reference n=(0,1,2).*

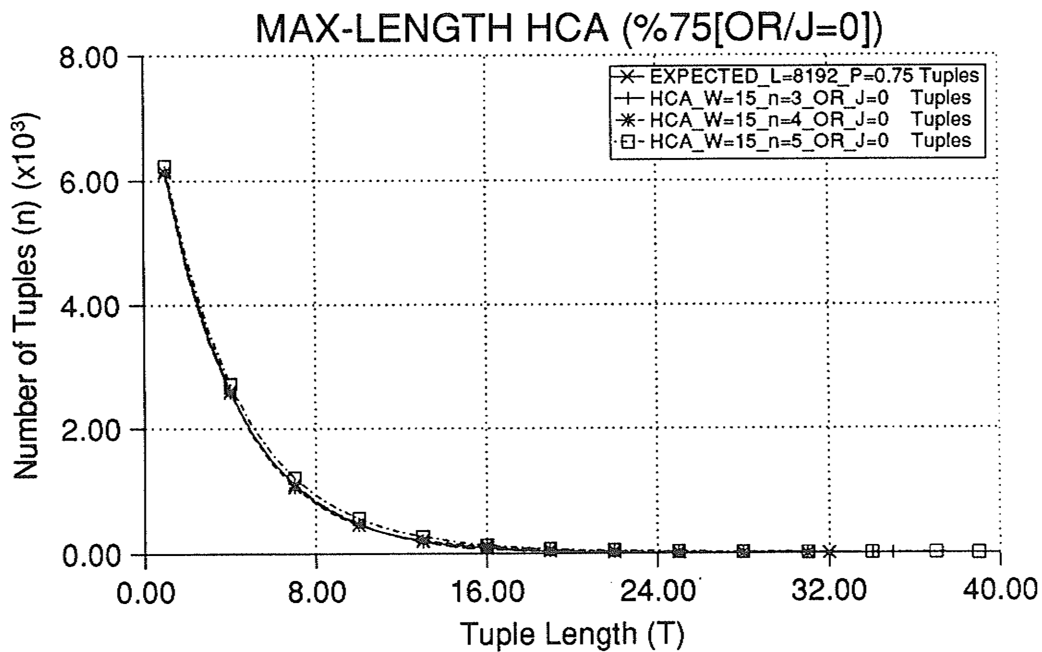

Figure I.3b): *Bit Sequence Tuple Profile of Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=0]; W=15; L=8,192; Reference n=(3,4,5).*

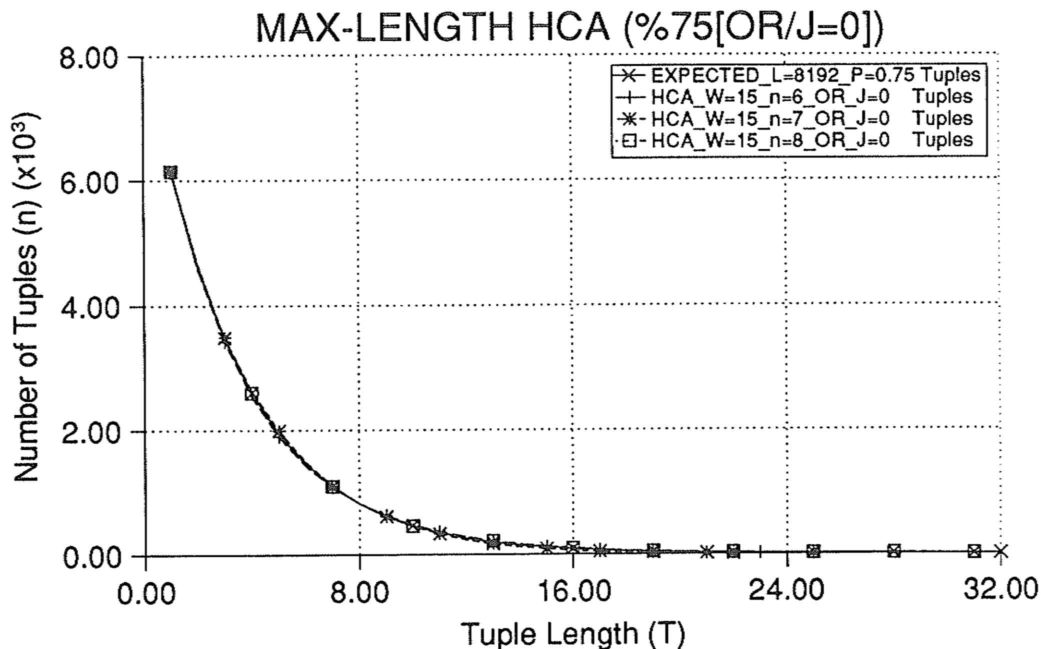

Figure I.3c): *Bit Sequence Tuple Profile of Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=0]; W=15; L=8,192; Reference n=(6,7,8).*

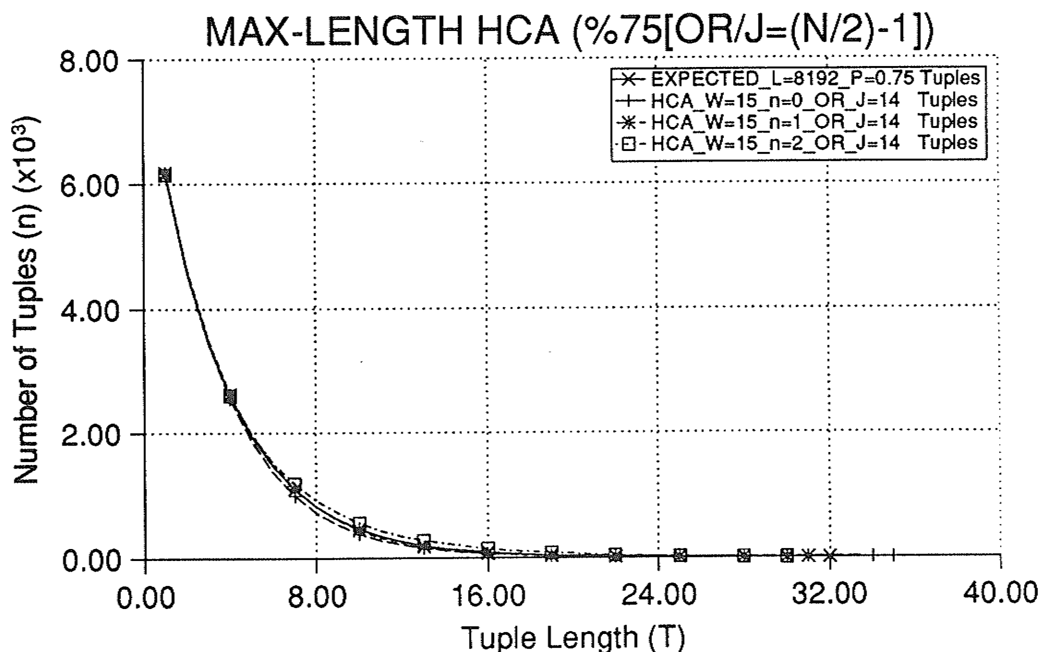

Figure I.4a): *Bit Sequence Tuple Profile of Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=14]; W=15; L=8,192; Reference n=(0,1,2).*

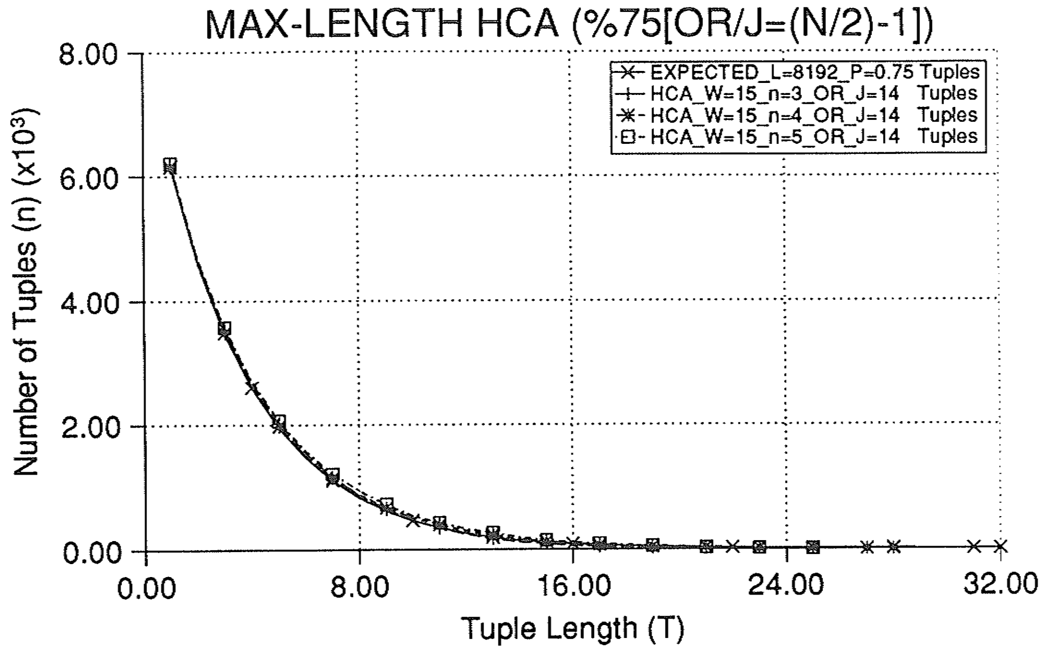

Figure I.4b): *Bit Sequence Tuple Profile of Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=14]; W=15; L=8,192; Reference n=(3,4,5).*

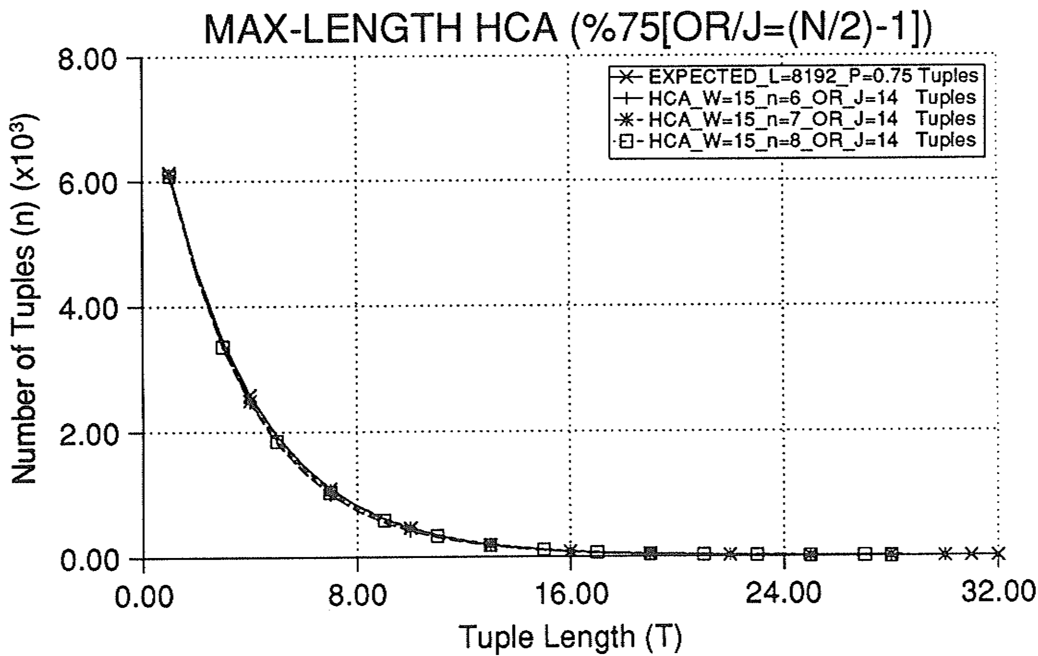

Figure I.4c): *Bit Sequence Tuple Profile of Rule 90/150 HCA weighted to 75% (OR-gate bank): [J=14]; W=15; L=8,192; Reference n=(6,7,8).*

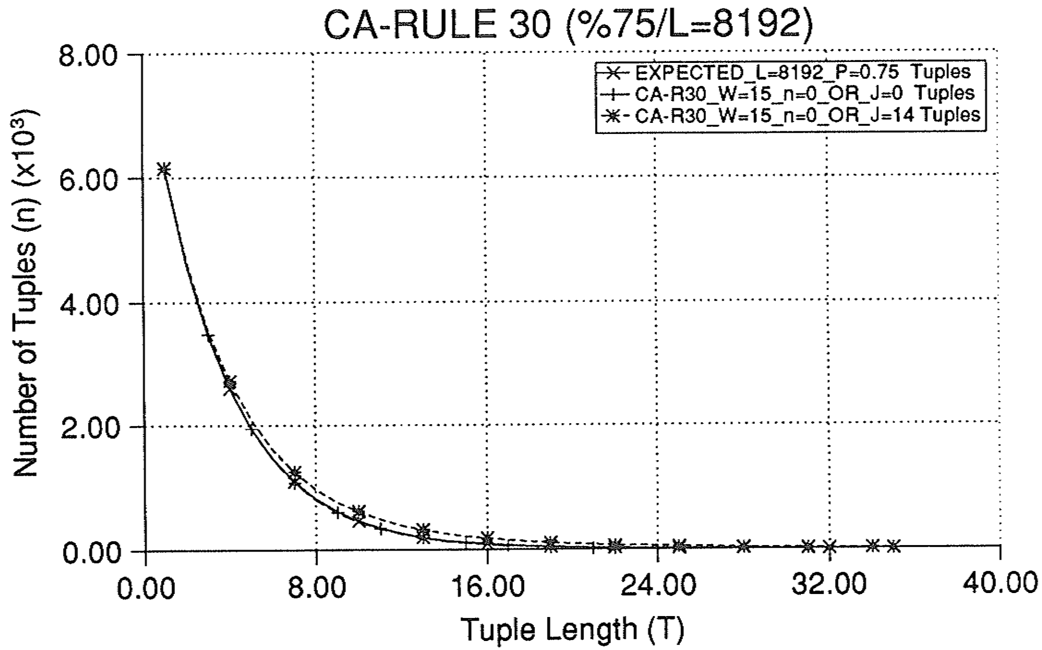

Figure I.5: *Bit Sequence Tuple Profile of Rule 30 CA weighted to 75% (OR-gate bank): [J=0], [J=14]; W=15; L=8,192; Reference n=0.*

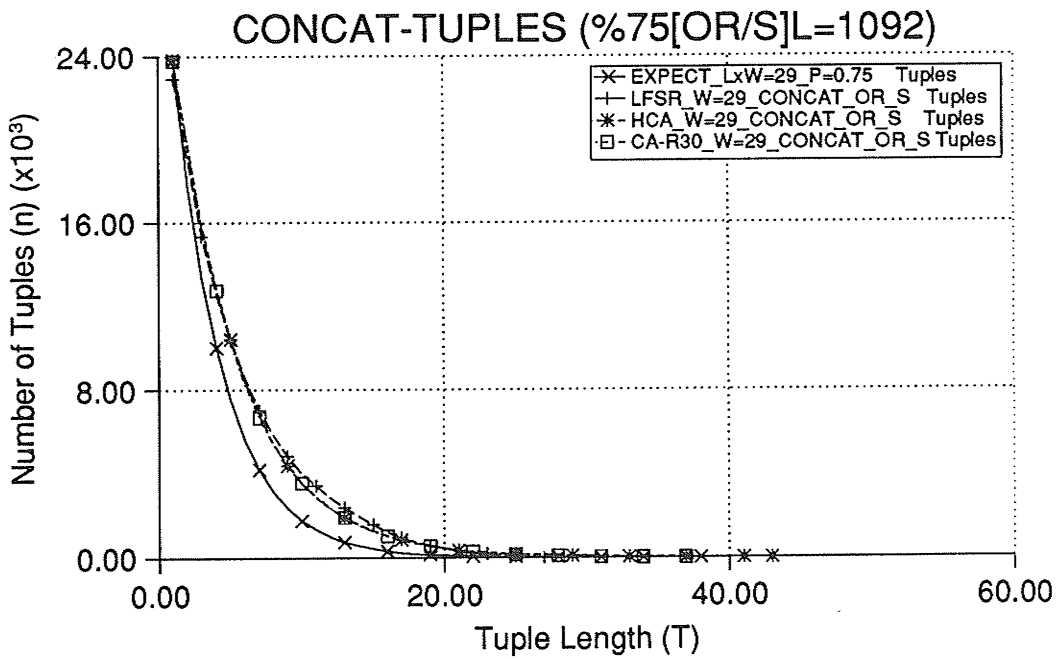

Figure I.6: *Concatenated Bit Sequence Tuple Profiles for finite state machines weighted to 75% (OR-gate bank): [S]; W=29; L=1,092.*

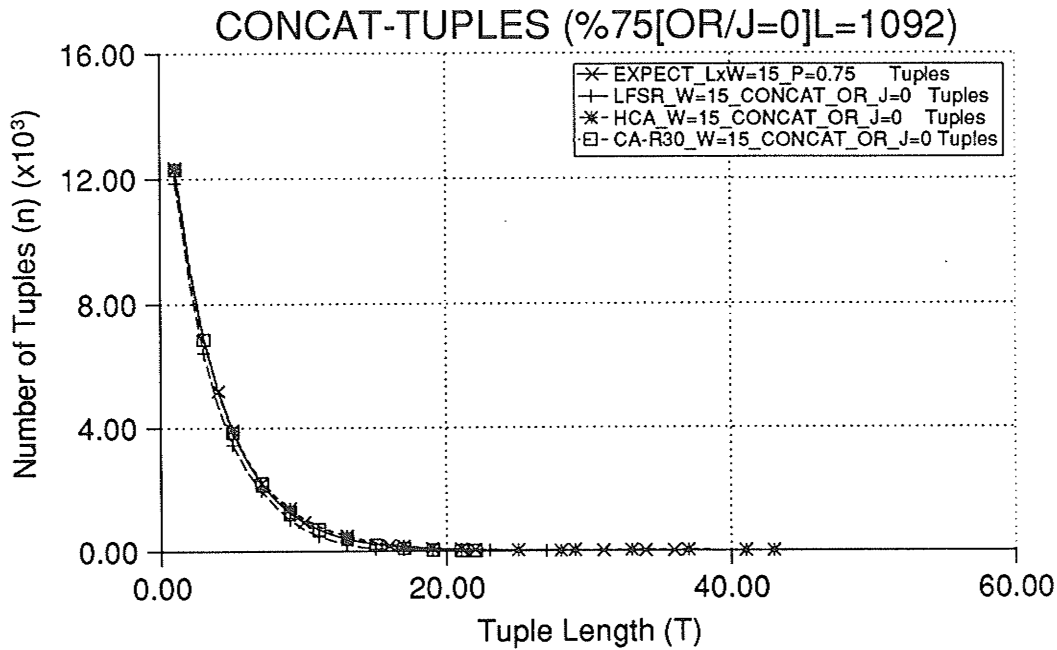

Figure I.7: *Concatenated Bit Sequence Tuple Profiles for finite state machines weighted to 75% (OR-gate bank): [J=0]; W=15; L=1,092.*

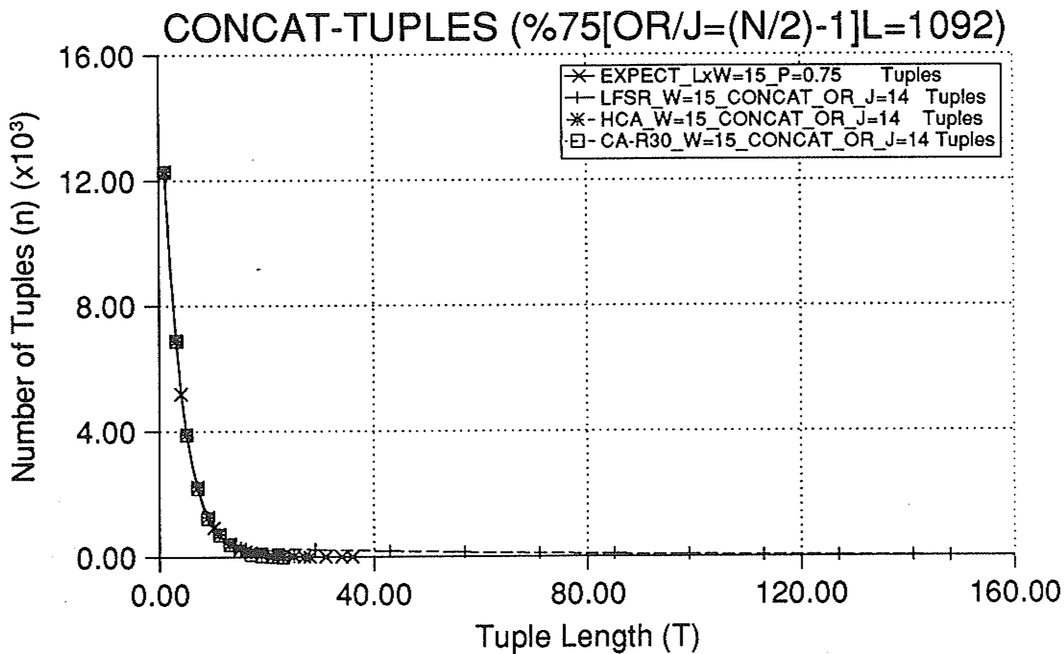

Figure I.8: *Concatenated Bit Sequence Tuple Profiles for finite state machines weighted to 75% (OR-gate bank): [J=14]; W=15; L=1,092.*

Appendix J

Glossary

AT: *Area-Time*
BILBO: *Built In Logic Block Observer/Observation*
BIST: *Built-In Self-Test*
CA: *Cellular Automata/Automaton*
CALBO: *Cellular Automaton Logic Block Observer/Observation*
DFT: *Design For Testability*
HCA: *Hybrid Cellular Automata/Automaton*
LFSR: *Linear Feedback Shift Register*
LSSD: *Level Sensitive Scan Design*
MISR: *Multiple Input Signature Analyzer*
NLFSR: *Non-Linear Feedback Shift Register*
PMF: *Probability Mass Function*
PTPG: *Pseudorandom Test Pattern Generation*
RTPG: *Random Test Pattern Generation*
SRL: *Shift Register Latch*
UPTPG: *Unbiased Pseudorandom Test Pattern Generation*
VLSI: *Very Large Scale Integration*
WCALBO: *Weighted Cellular Automaton Logic Block Observer/Observation*
WTPG: *Weighted Test Pattern Generator/Generation*