Incremental Design Techniques with Non-Preemptive Refinement for Million-Gate FPGAs

Jing Ma

Dissertation submitted to the Faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
In
Electrical Engineering

Dr. Peter M. Athanas, Co-chair
Dr. Mark T. Jones, Co-chair
Dr. Scott F. Midkiff
Dr. Srinidhi Varadarajan
Dr. Brian D. Woerner

January 13th, 2003
Bradley Department of Electrical
and Computer Engineering
Blacksburg, Virginia

Keywords: FPGA, Incremental Design, Placement, Design Tool

Incremental Design Techniques with Non-Preemptive Refinement for Million-Gate FPGAs

Jing Ma

ABSTRACT

This dissertation presents a Field Programmable Gate Array (FPGA) design methodology that can be used to shorten the FPGA design-and-debug cycle, especially as gate counts increase to many millions. Core-based incremental placement algorithms, in conjunction with fast interactive routing, are investigated to reduce the design processing time by distinguishing the changes between design iterations and reprocessing only the changed blocks without affecting the remaining part of the design. Different from other incremental placement algorithms, this tool provides the function not only to handle small modifications; it can also incrementally place a large design from scratch at a rapid rate. Incremental approaches are inherently greedy techniques, but when combined with a background refinement thread, the incremental approach offers the instant gratification that designers expect, while preserving the fidelity attained through batch-oriented programs. An incremental FPGA design tool has been developed, based on the incremental placement algorithm and its background refiner.

Design applications with logical gate sizes varying from tens of thousands to approximately one million are built to evaluate the execution of the algorithms and the design tool. The results show that this incremental design tool is two orders of magnitude faster than the competing approaches such as the Xilinx M3 tools without sacrificing much quality. The tool presented places designs at the speed of 700,000 system gates per second. The fast processing speed and user-interactive property make the incremental design tool potentially useful for prototype developing, system debugging and modular testing in million-gate FPGA designs.

To my husband Xinming

And my parents

Acknowledgements

First of all, I would like to express my deepest appreciation to my advisors, Professor Peter M. Athanas and Professor Mark T. Jones, for their endless encouragement and support throughout my Ph.D. study at Virginia Tech. Their motivation, creativity, and rich knowledge always inspire me to move forward in my dissertation research. Their strong support and invaluable inspiration were always there for me whenever I met problem. Greatest thanks give to their enduring patience. From their answers to each of my question to their suggestions to every writing improvement in my dissertation, what I learned is not just a technical concept or an English word; I learned important personal characters as an excellent professor, and that would benefit me a lot in my future career.

I also would like to thank all my committee members, Dr. Midkiff, Dr. Woerner, and Dr. Varadarajan, for their insightful technical guidance and strong support. I am grateful for the support and help from everyone in the Loki team at Virginia Tech and Xilinx. It has been a great pleasure working in this wonderful team in the past three years. I would especially thank Dr. Steven Guccione and Dr. Cameron Patterson for their support of answering all my questions and providing all the necessary JBits tools. A special thank to Dr. Phil James-Roxby for sharing his brilliant ideas in placement algorithms and testing methods.

My sincere thanks to all my friends and colleagues in the Configurable Computer Lab who helped me in my research. Special thanks to Yanhua Yi, Joanthan Ballagh, Neil Steiner, Dennis Goetz, Jon Scalera, and Jae Park. I wish everybody in the CCM lab great achievements in the future.

I am extremely grateful for the love and support from my family. To my father Shaohua Ma, this is for you. I knew you could see me today, and I knew you would be so proud of me. No words can express my deepest thanks to my mother, Guiyu Ma, my brother and all the relatives from my big warm family in China. Without your love and support, I will not be here today.

Lastly, my deep thanks to my beloved husband, Dr. Xinming Huang. For his love, his understanding and his support at any time, from anywhere I needed. I wish both of us succeeded in our academic career.

Jing Ma

Virginia Polytechnic Institute and State University
January 2003

Contents

Abstract ..... **ii**

List of Figures ..... **ix**

List of Tables ..... **xiii**

Chapter 1 Introduction **1**

Chapter 2 Prior Work **7**

2.1 FPGA Design Tools 7

2.1.1 Current FPGA design tools and traditional design flow 7

2.1.2 Review of placement algorithms 9

2.1.3 Problems in the current design flow 11

2.2 Incremental Compilation 15

2.3 Garbage Collection 18

2.4 JBits and JBits tools 20

2.4.1 JBits APIs 20

2.4.2 RTPCores 21

2.4.3 JRoute 22

2.4.4 BoardScope 22

2.4.5 A simple example of JBits-based FPGA design 23

2.5 Summary 26

Chapter 3 Core-Based Incremental Placement Algorithms **27**

3.1 Incremental Placement Algorithm 27

3.1.1 Basic implementations 28

3.1.2 Three methods 36

3.1.2.1 Method 1: Nearest position 36

3.1.2.2 Method 2: Recursive search 39

3.1.2.3 Method 3: Force-direct 41

3.1.3 Terminating condition 42
3.2 Guided Placement 46
3.2.1 Implementation 46
3.2.2 Exception handling in guided placement methodology 50
3.3 Cluster Merge 51
3.3.1 Implementation in incremental placement algorithms 52
3.3.2 Implementation for a changed design 56
3.4 Summary 57

Chapter 4 An Incremental FPGA Integrated Design Environment 58
4.1 Requirement Analysis 58
4.2 Program Structures and Implementations 60
4.3 Dynamic Linking Techniques 68
4.4 Data Structure 73
4.5 A Simple Design Example 75
4.6 Summary 77

Chapter 5 Garbage Collection Mechanisms 78
5.1 The Importance of the Garbage Collection Techniques 78
5.2 Simulated Annealing Algorithms 79
5.3 Implementation of the Garbage Collection Techniques 84
5.3.1 Parameters 84
5.3.2 Integration with the incremental design tool 87
5.4 Performance Analysis 89

Chapter 6 Application Development 95
6.1 Random Circuit Generator 96
6.2 Polynomial Computation Example 100
6.3 Just Another Hardware Description Language 103
6.4 Xilinx Tools 106
6.5 Summary 107

Chapter 7 Experimental Results 108
7.1 Features of the Incremental Design IDE 109
7.2 Performance Analysis of the Incremental Design Tool 112
7.2.1 Performance of the incremental placer 113
7.2.2 Comparison of the three methods 121
7.2.3 Exception handling 129
7.2.4 Performance of the guided placement methodology 133
7.2.4.1 Performance of the direct copying method 133
7.2.4.2 Performance of handling the minor change in a design 135
7.2.4.3 Performance of handling the major change in a design 138
7.2.4.4 Performance of the exception handling strategy 141
7.2.4.5 Summary 143
7.2.5 Cluster merge 144
7.3 Comparison with Traditional Placers 148
7.4 Advantages of the Background Refiner 157
7.5 Summary 164

Chapter 8 Conclusion 166
8.1 Goals Reexamined 166
8.1.1 Incremental placement algorithms 166
8.1.2 Incremental design IDE 167
8.1.3 Garbage collection mechanism 168
8.1.4 Design evaluation 169
8.2 Future Directions 172
8.2.1 Using standard design circuits 172
8.2.2 More investigation in resource utilization comparison 173
8.2.3 Improving the core-based simulated annealing placer 173
8.2.4 Future expectations 173

Reference and Bibliography 175

List of Figures

2.1 Traditional FPGA design flow 9
2.2 Static JBits design flow 21
2.3 Block diagram of a simple design 23
2.4 Core view of the example design 25
2.5 State view of the example design 25
3.1 Illustration of the clustering method 29
3.2 Example of counting the wire length 31
3.3 Searching the desired position for an added core 33
3.4 A placement example for searching the desired position (a) Place Cores A and B (b) Adding Core C 33
3.5 Program flow chart of the incremental placement algorithm 35
3.6 Placing a shift core—Method 1 37
3.7 Placement example using Method 1 (a) Finding nearest empty position (b) Moving blocked cores if nearest empty position not found 38
3.8 Placement example using Method 2—Example 1 39
3.9 Placement example using Method 2 –Example2 40
3.10 A placement example that cannot find a desired position for an added core 43
3.11 Place the design in Figure3.10 using the refined terminating condition (a) Remove all the block cores (b) Place these cores as the descending order of their height 44
3.12 Method to place "homeless " cores 45
3.13 Program flow chart of the guided placement methodology 49
3.14 Illustration of the cluster merge 52
3.15 Flow chart for the cluster merge strategy 55
4.1 Use case model of the incremental FPGA IDE 60
4.2 Design flow of the JBits-based incremental FPGA design tool 61
4.3 System components and interaction 62
4.4 Structure diagram of the DesignTool package 65
4.5 Sequence diagram of the load class scenario 67

4.6 Dynamically link a user's design 69
4.7 Information flow during the design 73
4.8 Data structure of design registration 74
4.9 Block Diagram of a simple design 75
5.1 Flowchart of a typical simulated annealing algorithm 81
5.2 Controlling thread using tokens 89
5.3 Cooling schedule of the simulated annealing placer 90
5.4 Cost- searching curves (a) Polynomial degree=15 (b) Polynomial degree =20 91
5.5 A Placement example using Incremental Placement (Wire length =5) 92
5.6 A placement example using Simulated Annealing (Wire length=3) (a) Random initial configuration (b) Final placement 93
6.1 Circuit schematic of a polynomial with degree 3 101
6.2 Interface of the Slaac1v Platform in Jab 103
6.3 The circuit browser window 105
6.4 The schematic view window 105
6.5 Browsing a placed design using FPGA Editor 106
7.1 Main interface of the incremental FPGA design IDE 109
7.2 File chooser interface 110
7.3 Guide file option panel 110
7.4 A placed random circuit in XCV1000 in placement view 111
7.5 A placed polynomial computation design in XCV1000 in connection view 111
7.6 Bitstream execution of the design from Figure 7.5 using BoardScope simulator 112
7.7 Place-and-route success rates of the one-by-one connected random circuits 116
7.8 Placement speed of the incremental placer evaluated using the one-by-one connected random circuits 116
7.9 Place-and-route success rates of the partially randomly connected random circuits 119
7.10 The placement speed of the incremental placer evaluated using the partially randomly connected random circuits 119
7.11 Place-and-route success rates of the fully randomly connected random circuits 120
7.12 Placement speed of the incremental placer evaluated using the fully randomly connected random circuits 120

7.13 Comparing three methods in place-and-route success rate with testing circuits in one-by-one connection pattern 122
7.14 Comparing three methods in wire length with testing circuits in one-by-one connection pattern 122
7.15 Comparing three methods in placement time with testing circuits in one-by-one connection pattern 123
7.16 Comparing three methods in place-and-route success rate with the testing circuits in fully random connection pattern 126
7.17 Comparing three methods in wire length with the testing circuits in fully random connection pattern 126
7.18 Comparing three methods in Placement time with testing circuits in fully random connection pattern 127
7.19 Comparing three methods in place-and-route success rate with the testing circuits in partially random connection pattern 127
7.20 Comparing three methods in wire length with testing circuits in partially random connection pattern 128
7.21 Comparing three methods in Placement time with testing circuits in partially random connection pattern 128
7.22 A failed placement without the terminating condition refinement mechanism 130
7.23 A successful placement with the terminating condition refinement mechanism 131
7.24 Processing log file of the placement in Figure 7.23 132
7.25 Placement time comparison in guided placement methodology 136
7.26 Wire length comparison in guided placement methodology 138
7.27 An unsuccessful placement of a design with 90% change using the guided placement methodology without setting threshold 139
7.28 A successful placement of a design with 90% change using the guided placement methodology by setting threshold 140
7.29 Processing log of the design in Figure 7.28 140
7.30 Placement of a polynomial design with degree 5 guided by a design with degree 11 on Virtex XCV300 without the exception handling strategy 142
7.31 Placement of a polynomial design with degree 5 guided by a design with degree 11 on Virtex XCV1000 with the exception handling strategy 142

7.32 Processing log of the design in Figure 7.31 143
7.33 Comparison of the placement of a polynomial with degree 11 by changing the order cores are added without employing the cluster merge mechanism (a) place in different clusters (b) place in same cluster 146
7.34 Placement of a polynomial design with degree 11 by adding more cores in the design in Figure7.33a without employing cluster merge technique 146
7.35 Placement of a polynomial design with degree 11 by adding more cores in the design in Figure7.33b via employing cluster merge mechanism 147
7.36 Placement speed comparison on Xilinx Virtex XCV1000 150
7.37 Placement speed comparison on Xilinx Virtex XCV300 150
7.38 Wire length comparison with design implemented on Xilinx Xirtex XCV1000 153
7.39 Wire length comparison with design implemented on Xilinx Xirtex XCV300 153
7.40 Placement of a polynomial with degree 23 using Xilinx Tools 155
7.41 A placement solution of a polynomial with degree 13 using the incremental placement algorithm (Wire length=1269) 158
7.42 A placement solution of a polynomial with degree 13 using the background refiner (Wire length=988) 158
7.43 A placement solution of a polynomial with degree 16 using the design in Figure 7.41 as the guide design (wire length=2090) 159
7.44 A placement solution of a polynomial with degree 16 using the design in Figure 7.42 as the guide design (wire length=1571) 160
7.45 An unsuccessful placement of a polynomial with degree 16 from the incremental placement tool 161
7.46 A successful placement of a polynomial with degree 16 from the background refiner 162
7.47 A successful placement guided by the design in Figure 7.16 (Wire length =2022, device utilization is 73.6%) 162

Lists of Tables

6.1 Random circuit design description 98

6.2 Design statistics for random circuits with mean height 5 rows and mean width 2.5 columns on Virtex XCV300 99

6.3 Design statistics for random circuits with mean height 5 rows and mean width 2.5 columns on Virtex XCV1000 99

6.4 Design statistics for random circuits with mean height 10 rows and mean width 5 columns on Virtex XCV1000 100

6.5 Design Statistics for polynomial computation design on Virtex XCV300 101

6.6 Design Statistics for polynomial computation design on Virtex XCV1000 102

7.1 Implementation of the polynomial computation circuits on Virtex XCV300 113

7.2 Implementation of the polynomial computation circuits on Virtex XCV1000 114

7.3 Performance of the refined terminating condition 132

7.4 Placement time comparisons with polynomial designs on Xirtex XCV300 (seconds) 134

7.5 Placement time comparisons with polynomial designs on Xirtex XCV1000 (seconds) 134

7.6 Comparison of the placement performance with/o employing the cluster merge mechanism using polynomial designs in Virtex XCV1000 (P-Placed, F-Failed) 148

7.7 Comparison of the placement performance with/o employing the cluster merge mechanism using polynomial designs in Virtex XCV300 (P-Placed, F-Failed) 148

7.8 Processing speed comparisons for the entire design cycle (seconds) 152

7.9 Device utilization comparisons 154

7.10 Comparison of the computation time in guided placement methodology (seconds). 156

7.11 The performance of the background refiner 163