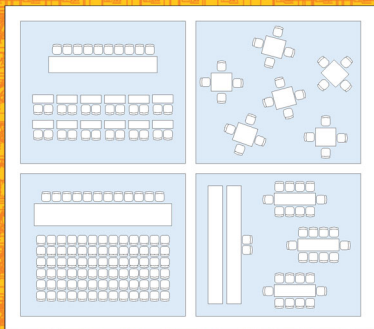


Christophe Bobda

# Introduction to Reconfigurable Computing

*Architectures, algorithms and applications*



*Foreword by Reiner Hartenstein*



Springer

# Introduction to Reconfigurable Computing

# Introduction to Reconfigurable Computing

Architectures, Algorithms,  
and Applications

*by*

Christophe Bobda

*University of Kaiserslautern, Germany*

 Springer

A C.I.P. Catalogue record for this book is available from the Library of Congress.

ISBN 978-1-4020-6088-5 (HB)

ISBN 978-1-4020-6100-4 (e-book)

---

Published by Springer,  
P.O. Box 17, 3300 AA Dordrecht, The Netherlands.  
[www.springer.com](http://www.springer.com)

*Printed on acid-free paper*

All Rights Reserved

© 2007 Springer

No part of this work may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, microfilming, recording or otherwise, without written permission from the Publisher, with the exception of any material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work.

To Lewin, Jan and Huguette for being so patient

# Foreword

"Christophe Bobda's book is an all-embracing introduction to the fundamentals of the entire discipline of Reconfigurable Computing, also seen with the eyes of a software developer and including a taxonomy of application areas.

Reconfigurable Computing is a disruptive innovation currently going to complete the most important breakthrough after introduction of the von Neumann paradigm. On software to FPGA migrations a dazzling array of publications from a wide variety application areas reports speed-up factors between 1 and 4 orders of magnitude and promises to reduce the electricity bill by at least an order of magnitude. Facing the cyberinfrastructure's growing electricity consumption (predicted to reach 35–50% of the total electricity production by the year 2020 in the USA) also this energy aspect is a strategic issue.

The focal point of worldwide almost 15 million software developers will shift toward new solutions of the productivity problems which stem from programming the coming many-core microprocessors and from educational deficits. Currently Reconfigurable Computing for High Performance computing is even disorienting the supercomputing scene. Their tremulous question is: Do we need to learn hardware design?

In the past, when students asked for a text book, we had to refer to a collection of specialized books and review articles focused on individual topics or special application areas, where Reconfigurable Computing takes only a corner or a chapter, sometimes even treating FPGAs as exotic technology (although in important areas it is mainstream for a decade). The typical style of those books or articles assumes that the reader has a hardware background: a leap too far for the existing software developers community.

The book by Christophe Bobda, however, has also been written for people with a software background, substantially reducing the educational leap hybridizing the gap. His book has the potential to become a best-seller and to

stimulate the urgently needed transformation of the software developer population's mindset, by playing a similar role as known from the famous historic Mead-&-Conway textbook for the VLSI design revolution.

Reiner Hartenstein, IEEE fellow,  
Professor, TU Kaiserslautern "

# Contents

Foreword	vii
Preface	xiii
About the Author	xv
List of Figures	xvii
List of Tables	xxv
1. INTRODUCTION	1
1 General Purpose Computing	2
2 Domain-Specific Processors	5
3 Application-Specific Processors	6
4 Reconfigurable Computing	8
5 Fields of Application	9
6 Organization of the Book	11
2. RECONFIGURABLE ARCHITECTURES	15
1 Early Work	15
2 Simple Programmable Logic Devices	26
3 Complex Programmable Logic Device	28
4 Field Programmable Gate Arrays	28
5 Coarse-Grained Reconfigurable Devices	49
6 Conclusion	65
3. IMPLEMENTATION	67
1 Integration	68
2 FPGA Design Flow	72
3 Logic Synthesis	75
4 Conclusion	98



4. HIGH-LEVEL SYNTHESIS FOR RECONFIGURABLE DEVICES	99
1 Modelling	100
2 Temporal Partitioning Algorithms	120
3 Conclusion	148
5. TEMPORAL PLACEMENT	149
1 Offline Temporal Placement	151
2 Online Temporal Placement	160
3 Managing the Device's Free Space with Empty Rectangles	161
4 Managing the Device's Occupied Space	165
5 Conclusion	179
6. ONLINE COMMUNICATION	181
1 Direct Communication	181
2 Communication Over Third Party	182
3 Bus-based Communication	183
4 Circuit Switching	183
5 Network on Chip	188
6 The Dynamic Network on Chip (DyNoC)	199
7 Conclusion	212
7. PARTIAL RECONFIGURATION DESIGN	213
1 Partial Reconfiguration on Virtex Devices	214
2 Bitstream Manipulation with <i>JBits</i>	216
3 The Modular Design Flow	217
4 The Early Access Design Flow	225
5 Creating Partially Reconfigurable Designs	234
6 Partial Reconfiguration using Handel-C Designs	244
7 Platform design	246
8 Enhancement in the Platform Design	256
9 Conclusion	257
8. SYSTEM ON A PROGRAMMABLE CHIP	259
1 Introduction to SoPC	259
2 Adaptive Multiprocessing on Chip	268
3 Conclusion	284

9. APPLICATIONS	285
1 Pattern Matching	286
2 Video Streaming	294
3 Distributed Arithmetic	298
4 Adaptive Controller	307
5 Adaptive Cryptographic Systems	310
6 Software Defined Radio	313
7 High-Performance Computing	315
8 Conclusion	317
References	319
Appendices	336
A Hints to Labs	337
1 Prerequisites	338
2 Reorganization of the Project Video8_non_pr	338
B Party	345
C Quick Part-Y Tutorial	349

# Preface

One indicator of the growing importance of Reconfigurable Computing is the large number of events (conferences, workshops, meetings) organized and devoted to this topic in the last couple of years. Also, the growth observed in the market share of programmable logic devices, particularly the FPGAs, is an indicator of the strong interest in reconfigurable logic.

Following this development, teaching reconfigurable computing, which was initiated in many Universities a couple of years before, has gained more importance. The curricula in reconfigurable computing varies from simple seminars to more heavy syllabus including lectures, exercises and labs.

Many people among whom Reiner Hartenstein have been advocating years ago in favour of a normalized reconfigurable computing syllabus. Aware of the importance of a teaching book in this normalization, during a bi-annual meeting on reconfigurable computing held in Dagstuhl in 2003 [11], Hartenstein coined the importance of a text book in reconfigurable computing and proposed the attendees to write one. He suggested to have several people writing to minimize the work and have the book published as faster as possible. Unfortunately, this initiative was not pursued.

A couple of months after this initiative, in the summer term 2004, I started teaching a course in reconfigurable computing at the university of Erlangen-Nuremberg. With the difficulties of acquiring teaching materials and labs, I started writing a script to ease the learning process of students. The positive feedback gained from the student encouraged me to continue writing. Further repetitions of the course in winter term 2004 and in winter term 2006 were used to improve the course contents. It should be mentioned that in a couple of books [153] [220] [134], reconfigurable computing were published in between. However, none of them were found to cover the complete aspects of reconfigurable computing as I use to teach in my course.

My goal in writing this book was to provide a strong theoretical and practical background, as a contribution for a syllabus in reconfigurable computing.

A short overview on the content of each chapter is provided in Section 6 of Chapter 1.

This book targets graduate students and lecturers in computer engineering, computer science and electrical engineering. Also professional in the aforementioned field can use the book as well. We supply the book with teaching materials (slides and labs) to ease the course preparation for those willing to introduce a reconfigurable computing curricula. The teaching material as well as the labs can be downloaded from the course Web page at [www.bobda.net/rc-book](http://www.bobda.net/rc-book).

Finally, despite all the effort place in the review, we cannot be sure that all the mistakes were filtered out. We will therefore be grateful to receive your comments and feedback on possible errors.

Kaiserslautern, June 2007  
Christophe Bobda

## About the Author

Dr. Bobda received the Licence degree in mathematics from the University of Yaounde, Cameroon, in 1992, the diploma of computer science and the Ph.D. degree (with honors) in computer science from the University of Paderborn in Germany in 1999 and 2003, respectively. In June 2003, he joined the department of computer science at the University of Erlangen-Nuremberg in Germany as post doc. In October 2005, he moved to the University of Kaiserslautern as Junior Professor, where he leads the working group Self-Organizing Embedded Systems in the department of computer science. His research interests include reconfigurable computing, self-organization in embedded systems, multiprocessor on chip and adaptive image processing.

Dr. Bobda received the Best Dissertation Award 2003 from the University of Paderborn for his work on synthesis of reconfigurable systems using temporal partitioning and temporal placement.

Dr. Bobda is member of The IEEE Computer Society, the ACM and the GI. He has also served in the program committee of several conferences (FPL, FPT, RAW, RSP, ERSA, DRS) and in the DATE executive committee as proceedings chair (2004, 2005, 2006, 2007). He served as reviewer of several journals (*IEEE TC*, *IEEE TVLSI*, *Elsevier Journal of Microprocessor and Microsystems*, *Integration the VLSI Journal*) and conferences (DAC, DATE, FPL, FPT, SBCCI, RAW, RSP, ERSA).

# List of Figures

1.1	The Von Neumann Computer architecture	2
1.2	Sequential and pipelined execution of instructions on a Von Neumann Computer	4
1.3	ASIP implementation of Algorithm 1	7
1.4	Flexibility vs performance of processor classes	8
2.1	Structure of the Estrin Fix-Plus Machine	17
2.2	The basic building blocks of Fix-Plus Machine	17
2.3	The wiring harness of the Estrin-Machine	18
2.4	The motherboard of the Estrin's Fix-Plus	18
2.5	Estrin at work: Hand-Controlled Reconfiguration	19
2.6	Structure of the Rammig machine	20
2.7	<i>META-46 GOLDLAC</i>	20
2.8	General architecture of the XPuter as implemented in the Map oriented Machine (MOM-3) prototype	21
2.9	Structure of the XPuter's reconfigurable ALU	22
2.10	Programmable Active Memory (PAM) architecture as array of (PABs)	23
2.11	Architecture of the SPLASH II array board	25
2.12	PAL and PLA implementations of the functions $F1 = A \cdot C + A \cdot \overline{B}$ and $F2 = A \cdot B + B \cdot C$	27
2.13	Structure of a CPLD device	28
2.14	Structure of an FPGA	29
2.15	Antifuse FPGA Technology	30
2.16	A Xilinx SRAM cell	31
2.17	Use of SRAM in FPGA-Configuration	31

2.18	EEPROM Technology	32
2.19	Implementation of $f=ab$ in a 2-input MUX	33
2.20	Implementation of a Full adder using two 4-input one output MUX	35
2.21	The Actel basic computing blocks uses multiplexers as function generators	36
2.22	2-input LUT	36
2.23	Implementation of a full adder in two 3-input LUTs	37
2.24	Basic block of the Xilinx FPGAs	38
2.25	CLB in the newer Xilinx FPGAs (Spartan 3, Virtex 4 and Virtex 5)	38
2.26	Logic Element in the Cyclone II	39
2.27	Stratix II Adaptive Logic Module	40
2.28	The four basic FPGA structures	41
2.29	Symmetrical array arrangement in a) the Xilinx and b) the Atmel AT40K FPGAs	42
2.30	Virtex routing resource	42
2.31	Local connection of an Atmel Cell	42
2.32	Row based arrangement on the Actel ACT3 FPGA Family	43
2.33	Actel's ACT3 FPGA horizontal and vertical routing resources	44
2.34	Actel ProASIC local routing resources	45
2.35	Hierarchical arrangement on the Altera Stratix II FPGA	46
2.36	LAB connection on the Altera Stratix devices	46
2.37	General structure of an I/O component	47
2.38	Structure of a Xilinx Virtex II Pro FPGA with two PowerPC 405 Processor blocks	49
2.39	Structure of the PACT XPP device	51
2.40	The XPP ALU Processing Array Element. The structure of the RAM ALU is similar.	52
2.41	Structure of the NEC Dynamically Reconfigurable Processor	53
2.42	The DRP Processing Element	54
2.43	Structure of the picoChip device	55
2.44	The Quicksilver ACM hierarchical structure with 64 nodes	56
2.45	ACM Node and Routing resource	57
2.46	IPflex DAP/DNA reconfigurable processor	59
2.47	The Stretch 5530 configurable processor	59

2.48	Pipeline Reconfiguration: Mapping of a 5 stage virtual pipeline auf eine 3 stage	63
3.1	Architecture of a run-time reconfigurable system	69
3.2	A CPU-RPU configuration and computation step	70
3.3	The FPGA design flow	73
3.4	A structured digital system	75
3.5	Example of a boolean network with: $y_1 = x_1 + x_2$ , $y_2 = x_3 \cdot x_4$ , $y_3 = \overline{x_5 \cdot x_6}$ , $y_4 = \overline{y_1 + y_2}$ , $z_1 = y_1 + y_4$ , and $z_2 = y_2 \oplus y_3$	76
3.6	BDD-representation of the function $f = abc + \bar{b}d + b\bar{c}d$	79
3.7	Example of a $K$ -feasible cone $C_v$ at a node $v$	83
3.8	Example of a graph covering with $K$ -feasible cone and the corresponding covering with LUTs	83
3.9	Chortle two-level decomposition	84
3.10	Example of multi-level decomposition	86
3.11	Exploiting reconvergent paths to reduce the amount of LUTs used	87
3.12	Logic replication at fan-out nodes to reduce the number of LUTs used	88
3.13	Construction of the network $N_t$ from the cone $C_t$ at node $t$	90
3.14	Minimum height 3-feasible cut and node mapping	91
3.15	Illustration of the two cases in the proof of Lemma 3.8	92
3.16	Transforming $N_t$ into $N'_t$ by node collapsing	94
3.17	Transforming the node cut constraints into the edge cut ones	94
3.18	Improvement of the FlowMap algorithm through efficient predecessor packing	97
4.1	Dataflow Graph for Quadratic Root	102
4.2	Sequencing graph with a branching node linking to two different sub graphs	103
4.3	Transformation of a sequential program into a FSM	105
4.4	Transformation of the greatest common divisor program into an FSM	106
4.5	The datapath and the corresponding FSM for the GCD-FSM	107
4.6	Dataflow graph of the functions: $x = ((a \times b) - (c \times d)) + ((c \times d) - (e - f))$ and $y = ((c \times d) - (e - f)) - ((e - f) + (g - h))$	109



4.7	HLS of the graph in figure 4.6 on a an architecture with one instances of the resource types $+$ , $*$ and $-$	110
4.8	HLS of the graph in figure 4.6 on a reconfigurable device	111
4.9	Partitioning of a coarse-grained node in the dataflow graph.	113
4.10	Example of configuration graph	116
4.11	Wasted resources	117
4.12	Partitioning of the graph $G$ with connectivity 0.24 with an algorithm that produces a quality of 0.25	119
4.13	Partitioning of the graph $G$ with connectivity 0.24 with an algorithm that produces a quality of 0.45	119
4.14	Scheduling example with the ASAP-algorithm	122
4.15	ALAP Scheduling example	122
4.16	An example of list scheduling using the depth of a node as priority	124
4.17	Levelizing effect on the list-scheduling on a dataflow graph	128
4.18	Partitioning with a better quality than the list-scheduling	128
4.19	Dataflow graph transformation into a network	134
4.20	Transformation and partitioning steps using the network flow approach	135
4.21	1-D and 2-D spectral-based placement of a graph	136
4.22	Dataflow graph of $f = ((a + b) * c) - ((e + f) + (g * h))$	140
4.23	3-D spectral placement of the DFG of figure 4.22	141
4.24	Derived partitioning from the spectral placement of figure 4.23	141
4.25	Internal and external edges of a given nodes	143
4.26	Partitioning of a graph into two sets with common sets of operators	146
4.27	Logical partitioning of the graph of figure 4.26	147
4.28	Implementation of configuration switching with the partitions of figure 4.27	147
5.1	Temporal placement as 3-D placement of blocks	150
5.2	First-fit temporal placement of a set of clusters	154
5.3	Valid two dimensional packing	157
5.4	A non valid two dimensional packing	158
5.5	3-D placement and corresponding interval graphs, complement graphs and oriented packing	159
5.6	Various types of empty rectangles	162

5.7	Increase of the number of MER through the insertion of a new component	163
5.8	Two different non-overlapping rectangles representations	164
5.9	Splitting alternatives after new insertion	164
5.10	IPR of a new module $v$ relative to a placed module $v'$	167
5.11	Impossible and possible placement region of a component $v$ prior to its insertion	168
5.12	Nearest possible feasible point of an optimal location that falls within the IPR	170
5.13	Moving out of consecutive overlapping IPRs	172
5.14	Expanding existing modules and shrinking chip area and the new	173
5.15	Characterization of IPR of a given component: The set of contours (left), the contour returned by the modified CUR (middle), the contour returned by the CUR (right)	174
5.16	<i>Placement of a component on the chip (left) guided by the communication with its environment (right)</i>	176
5.17	<i>Computation of the union of contours. The point on the boundary represent the potentially moves from the median out of the IPR.</i>	178
6.1	Direct communication between placed modules on a reconfigurable device	182
6.2	Drawback of circuit switching in temporal placement Placing a component using 4 PEs will not be possible, although enough free resources are available	185
6.3	The RMBoc architecture	186
6.4	RMBoc FPGA implementation	187
6.5	Crosspoint architecture	187
6.6	A Network on Chip on a 2-D Mesh	189
6.7	Router Architecture	190
6.8	A general FIFO Implementation	191
6.9	General format of a packet	192
6.10	Arbiter to control the write access at output data lines	193
6.11	A general wrapper architecture	194
6.12	Implementation of a large reconfigurable module on a Network on Chip	199
6.13	The communication infrastructure on a DyNoC	201
6.14	A impossible placement scenario	203

6.15	A strongly connected configuration on a DyNoC	204
6.16	Obstacle avoidance in the horizontal direction	206
6.17	Obstacle avoidance in the vertically direction	207
6.18	Placement that cause an extreme long routing path	208
6.19	Router guiding in a DyNoC	209
6.20	DyNoC implementation of a traffic light controller on a VirtexII-1000	212
7.1	Generation of bitstreams for partial reconfiguration	215
7.2	Routing tools can route the same signals on different paths	217
7.3	The recommended directory structure for the modular design flow	219
7.4	Limitation of a PR area to a block (dark) and the actual dimensions (light)	225
7.5	Scheme of a PR application with a traversing bus that may not be interrupted	226
7.6	Improved directory structure for the Early Access Design Flow	227
7.7	Usage of the new EA bus macros	229
7.8	Narrow (a) and wide (b) bus macro spanning two or four CLBs	230
7.9	Three nested bus macros	230
7.10	Scheme of <i>Animated Patterns</i>	235
7.11	Two patterns that can be created with modules of <i>Animated Patterns</i> . Left: the middle beam is moving. Right: the diagonal stripes are moving	236
7.12	Scheme of <i>Video8</i>	236
7.13	Reconstructing example <i>Video8</i> to place the partially reconfigurable part and connected modules in the top-level	238
7.14	Moving a partially reconfigurable module to the top-level design on the example of <i>Video8</i>	238
7.15	Modules using resources (pins) not available in their placement area (the complete column) must use <i>feed-through</i> signals to access those resources	246
7.16	Illustration of th pin problematique on the RC200-Board	248
7.17	Architecture of the ESM-Baby board	251
7.18	Architecture of the ESM MotherBoard	253
7.19	Intermodule communication possibilities on the ESM	254
7.20	SRAM-based intermodule communication on the ESM	255

7.21	Possible enhancement of the Erlangen Slot Machine on the Xilinx Virtex 4 and Virtex 5 FPGAs	257
8.1	Integration of PCB modules into a single chip: from system on PCB to SoC	260
8.2	Example of system ingration with CoreConnect buses	266
8.3	Implementation of the OPB for two masters and two slaves	268
8.4	General adaptive multiprocessor hardware infrastructure	271
8.5	Structure of the on chip network	273
8.6	Implementation of the transceiver	274
8.7	The communication protocoll	275
8.8	Automatic hardware generation flow	277
8.9	The Platform-independent Hardware generation tool (PinHat)	279
8.10	The PinHaT framework	280
8.11	software configuration flow	282
8.12	4-Processor infrastructure for the SVD on the ML310-Board	283
9.1	Sliding windows for the search of three words in parallel	288
9.2	FSM recognizers for the word 'conte': a) sate diagram, b) transition table, c) basis structure the hardware implementation: 4 flip flops will be need to code a $5 \times 6$ transition table	291
9.3	a) Use of the common prefix to reduce the number of flip flops of the common word detector for 'partir', 'paris', 'avale', 'avant'. b) implementation without use of common prefix and common comparator set	291
9.4	Basic structure of a FSM-based words recognizer that exploits the common prefix and a common set of characters	292
9.5	Processing steps of the FSM for the word 'tictic'	293
9.6	Implementation of a $5 \times 5$ sliding windows	296
9.7	A modular architecture for video streaming on the ESM	297
9.8	Architecture of a distributed arithmetic datapath	300
9.9	k-parallel distributed arithmetic datapath	301
9.10	Datapath of the distributed arithmetic computation for floating-point numbers	304
9.11	An optical multimode waveguide is represented by a multiport with several transfer paths	305
9.12	Screenshot of the 6-parallel DA implementation of the recursive convolution equation on the Celoxica RC100-PP platform	306