

# FPGA Design



Philip Simpson

# FPGA Design

Best Practices for Team-based Design



Springer

Philip Simpson  
Altera Corporation  
San Jose, CA 95134  
USA  
Feilmidh@sbcglobal.net

ISBN 978-1-4419-6338-3 e-ISBN 978-1-4419-6339-0  
DOI 10.1007/978-1-4419-6339-0  
Springer New York Dordrecht Heidelberg London

Library of Congress Control Number: 2010930598

© Springer Science+Business Media, LLC 2010

All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer Science+Business Media, LLC, 233 Spring Street, New York, NY 10013, USA), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed is forbidden.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

© 2010 Altera Corporation

ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS & STRATIX are Reg. U.S. Pat & Tm. Off. and Altera marks in and outside the U.S.

Printed on acid-free paper

Springer is part of Springer Science+Business Media ([www.springer.com](http://www.springer.com))

# Preface

In August of 2006, an engineering VP from one of Altera's customers approached Misha Burich, VP of Engineering at Altera, asking for help in reliably being able to predict the cost, schedule and quality of system designs reliant on FPGA designs.

At this time, I was responsible for defining the design flow requirements for the Altera design software and was tasked with investigating this further.

As I worked with the customer to understand what worked and what did not work reliably in their FPGA design process, I noted that this problem was not unique to this one customer. The characteristics of the problem are shared by many Corporations that implement designs in FPGAs. The Corporation has many design teams at different locations and the success of the FPGA projects vary between the teams. There is a wide range of design experience across the teams. There is no working process for sharing design blocks between engineering teams.

As I analyzed the data that I had received from hundreds of customer visits in the past, I noticed that design reuse among engineering teams was a challenge. I also noticed that many of the design teams at the same Companies and even within the same design team used different design methodologies.

Altera had recently solved this problem as part of its own FPGA design software and IP development process.

I worked with the top talent in Altera Engineering to develop a Best Practices Design methodology based upon Altera's experience and the techniques used by many customers successfully in FPGA design. The resulting methodology was presented and implemented at the customer, with great success.

Through the analysis of past customer data and feedback from customers over the last 3 years, it has become clear that this challenge exists broadly in the industry. The challenge is not specific to one specific FPGA vendor; it is an industry wide challenge.

As such, I have tuned the Best practices FPGA design methodology over the last 3 years and deployed it at several customers with great success.

This book captures the Best Practices FPGA design methodology and now makes it available to all design teams implementing system designs in FPGA devices.

San Jose, CA

Philip Simpson



# Contents

<b>1</b>	<b>Best Practices for Successful FPGA Design</b> .....	1
1.1	Introduction.....	1
<b>2</b>	<b>Project Management</b> .....	5
2.1	The Role of Project Management.....	5
2.1.1	Project Management Phases.....	5
2.1.2	Estimating a Project Duration.....	6
2.1.3	Schedule.....	6
<b>3</b>	<b>Design Specification</b> .....	9
3.1	Design Specification: Communication Is Key to Success.....	9
3.1.1	High Level Functional Specification.....	9
3.1.2	Functional Design Specification.....	10
<b>4</b>	<b>Resource Scoping</b> .....	15
4.1	Introduction.....	15
4.2	Engineering Resources.....	15
4.3	Third Party IP.....	16
4.4	Device Selection.....	16
4.4.1	Silicon Specialty Features.....	17
4.4.2	Density.....	18
4.4.3	Speed Requirements.....	19
4.4.4	Pin-Out.....	19
4.4.5	Power.....	20
4.4.6	Availability of IP.....	20
4.4.7	Availability of Silicon.....	20
4.4.8	Summary.....	21
<b>5</b>	<b>Design Environment</b> .....	23
5.1	Introduction.....	23
5.2	Scripting Environment.....	23
5.3	Interaction with Version Control Software.....	24
5.4	Use of a Problem Tracking System.....	25

- 5.5 A Regression Test System..... 26
- 5.6 When to Upgrade the Versions of the FPGA Design Tools..... 26
- 5.7 Common Tools in the FPGA Design Environment..... 27
- 6 Board Design ..... 29**
  - 6.1 Challenges that FPGAs Create for Board Design..... 29
  - 6.2 Engineering Roles and Responsibilities..... 30
    - 6.2.1 FPGA Engineers ..... 30
    - 6.2.2 PCB Design Engineer ..... 31
    - 6.2.3 Signal Integrity Engineer ..... 32
  - 6.3 Power and Thermal Considerations ..... 33
    - 6.3.1 Filtering Power Supply Noise ..... 33
    - 6.3.2 Power Distribution ..... 33
  - 6.4 Signal Integrity..... 34
    - 6.4.1 Types of Signal Integrity Problems..... 34
    - 6.4.2 Electromagnetic Interference ..... 35
  - 6.5 Design Flows for Creating the FPGA Pinout..... 36
    - 6.5.1 User Flow 1: FPGA Designer Driven ..... 36
    - 6.5.2 User Flow 2 ..... 38
    - 6.5.3 How Do FPGA and Board Engineers Communicate  
Pin Changes?..... 40
  - 6.6 Board Design Check List for a Successful FPGA Pin-Out..... 40
- 7 Power and Thermal Analysis ..... 41**
  - 7.1 Introduction..... 41
  - 7.2 Power Basics ..... 42
    - 7.2.1 Static Power ..... 42
    - 7.2.2 Dynamic Power ..... 42
    - 7.2.3 I/O power ..... 42
    - 7.2.4 Inrush Current ..... 43
    - 7.2.5 Configuration Power ..... 43
  - 7.3 Key Factors in Accurate Power Estimation ..... 43
    - 7.3.1 Accurate Power Models of the FPGA Circuitry ..... 44
    - 7.3.2 Accurate Toggle Rate Data on Each Signal ..... 44
    - 7.3.3 Accurate Operating Conditions..... 45
    - 7.3.4 Resource Utilization..... 46
  - 7.4 Power Estimation Early in the Design Cycle  
(Power Supply Planning) ..... 46
  - 7.5 Simulation Based Power Estimation  
(Design Power Verification)..... 47
    - 7.5.1 Partial Simulations ..... 50
  - 7.6 Best Practices for Power Estimation..... 50



- 8 RTL Design** ..... 51
  - 8.1 Introduction ..... 51
  - 8.2 Common Terms and Terminology ..... 51
  - 8.3 Recommendations for Engineers with an ASIC  
Design Background..... 53
  - 8.4 Recommended FPGA Design Guidelines..... 54
    - 8.4.1 Synchronous Versus Asynchronous ..... 54
    - 8.4.2 Global Signals ..... 54
    - 8.4.3 Dedicated Hardware Blocks ..... 55
    - 8.4.4 Use of Low-Level Design Primitives ..... 56
    - 8.4.5 Managing Metastability ..... 57
  - 8.5 Writing Effective HDL ..... 57
    - 8.5.1 What’s the Best Language ..... 58
    - 8.5.2 Good Design Practices ..... 59
    - 8.5.3 HDL for Synthesis ..... 65
  - 8.6 Analyzing the RTL Design ..... 75
    - 8.6.1 Synthesis Reports..... 75
    - 8.6.2 Messages ..... 76
    - 8.6.3 Block Diagram View..... 77
  - 8.7 Recommended Best Practices for RTL Design..... 78
  
- 9 IP and Design Reuse** ..... 79
  - 9.1 Introduction..... 79
  - 9.2 The Need for IP Reuse..... 79
    - 9.2.1 Benefits of IP Reuse..... 80
    - 9.2.2 Challenges in Developing a Design  
Reuse Methodology ..... 80
  - 9.3 Make Versus Buy ..... 82
  - 9.4 Architecting Reusable IP ..... 83
    - 9.4.1 Specification ..... 83
    - 9.4.2 Implementation Methods ..... 83
    - 9.4.3 Use of Standard Interfaces ..... 85
  - 9.5 Packaging of IP ..... 86
    - 9.5.1 Documentation ..... 87
    - 9.5.2 User Interface..... 87
    - 9.5.3 Compatibility with System Integration Tools ..... 88
    - 9.5.4 IP Security..... 89
  - 9.6 IP Reuse Checklist..... 90
  
- 10 The Hardware to Software Interface** ..... 91
  - 10.1 Software Interface..... 91
  - 10.2 Definition of Register Address Map ..... 91
  - 10.3 Use of the Register Address Map ..... 91
    - 10.3.1 IP Selection ..... 92
    - 10.3.2 Software Engineers Interface..... 92

10.3.3	RTL Engineers Interface .....	92
10.3.4	Verification Interface.....	93
10.3.5	Documentation.....	93
10.4	Summary.....	94
<b>11</b>	<b>Functional Verification .....</b>	<b>95</b>
11.1	Introduction.....	95
11.2	Challenges of Functional Verification .....	95
11.3	Glossary of Verification Concepts .....	96
11.4	RTL Versus Gate Level Simulation .....	97
11.5	Verification Methodology .....	97
11.6	Attack Complexity .....	98
11.6.1	Modularize Your Design and Your Tests .....	98
11.6.2	Plan for Expected Operation.....	98
11.6.3	Plan for the Unexpected.....	98
11.7	Functional Coverage .....	99
11.7.1	Directed Testing.....	100
11.7.2	Random Dynamic Simulation.....	100
11.7.3	Constrained Random Tests .....	100
11.7.4	Use of System Verilog for Design and Verification.....	100
11.7.5	General Testbench Methods.....	101
11.7.6	Self Verifying Testbenches .....	102
11.7.7	Formal Equivalency Checking.....	103
11.8	Code Coverage.....	104
11.9	QA Testing .....	104
11.9.1	Functional Regression Testing.....	104
11.9.2	GUI Testing for Reusable IP.....	105
11.10	Hardware Interoperability Tests.....	105
11.11	Hardware/Software Co-Verification .....	106
11.11.1	Getting to Silicon Fast .....	106
11.12	Functional Verification Checklist.....	106
<b>12</b>	<b>Timing Closure.....</b>	<b>107</b>
12.1	Timing Closure Challenges.....	107
12.2	The Importance of Timing Assignments and Timing Analysis ...	108
12.2.1	Background.....	108
12.2.2	Basics of Timing Analysis .....	109
12.3	A Methodology for Successful Timing Closure .....	115
12.3.1	Family and Device Assignments.....	115
12.3.2	Design Planning.....	116
12.3.3	Early Timing Estimation.....	121
12.3.4	CAD Tool Settings.....	122
12.4	Common Timing Closure Issues.....	129
12.4.1	Missing Timing Constraints.....	130
12.4.2	Conflicting Timing Constraints .....	130

- 12.4.3 High Fan-Out Registers ..... 130
- 12.4.4 Missing Timing by a Small Margin ..... 131
- 12.4.5 Restrictive Location Constraints ..... 131
- 12.4.6 Long Compile Times ..... 131
- 12.5 Design Planning, Implementation, Optimization and  
Timing Closure Checklist ..... 132
- 13 In-System Debug** ..... 133
  - 13.1 In-System Debug Challenges ..... 133
  - 13.2 Planning ..... 134
  - 13.3 Techniques ..... 134
    - 13.3.1 Use of Pins for Debug ..... 134
    - 13.3.2 Internal Logic Analyzer ..... 135
    - 13.3.3 Use of Debug Logic ..... 138
    - 13.3.4 External Logic Analyzer ..... 139
    - 13.3.5 Editing Memory Contents ..... 139
    - 13.3.6 Use of a Soft Processor for Debug ..... 140
  - 13.4 Use Scenarios ..... 140
    - 13.4.1 Power-Up Debug ..... 140
    - 13.4.2 Debug of Transceiver Interfaces ..... 141
    - 13.4.3 Reporting of System Performance ..... 141
    - 13.4.4 Debug of Soft Processors ..... 142
    - 13.4.5 Device Programming Issues ..... 143
  - 13.5 In-System Debug Checklist ..... 144
- 14 Design Sign-Off** ..... 145
  - 14.1 Sign-Off Process ..... 145
  - 14.2 After Sign-Off ..... 145
- Bibliography** ..... 147
- Index** ..... 149



# List of Figures

<b>Fig. 1.1</b>	Three steps to successful FPGA development.....	2
<b>Fig. 1.2</b>	Recommended best practices design methodology for successful FPGA design .....	3
<b>Fig. 2.1</b>	Percentage complete dilemma.....	7
<b>Fig. 3.1</b>	Sample revision control page .....	11
<b>Fig. 6.1</b>	Example .csv file that interfaces between board design SW and FPGA SW .....	32
<b>Fig. 6.2</b>	Design cycle diagram detailing engineering discipline involvement.....	33
<b>Fig. 6.3</b>	FPGA designer driven flow for creating the FPGA pin-out.....	37
<b>Fig. 6.4</b>	Board designer driven flow.....	39
<b>Fig. 7.1</b>	Key elements in accurate power estimation .....	43
<b>Fig. 7.2</b>	Graph of standby current versus temperature.....	45
<b>Fig. 7.3</b>	Sample power estimation spreadsheet for the Altera Stratix IV GX family .....	47
<b>Fig. 7.4</b>	Probability of nodes toggling .....	49
<b>Fig. 7.5</b>	Sample power estimation report from Quartus II PowerPlay Estimator .....	49
<b>Fig. 7.6</b>	Best practices for power estimation.....	50
<b>Fig. 8.1</b>	Behavioral modeling.....	52
<b>Fig. 8.2</b>	Structural modeling .....	52
<b>Fig. 8.3</b>	Synthesis.....	53
<b>Fig. 8.4</b>	Synchronizer for an asynchronous reset.....	55
<b>Fig. 8.5</b>	Instantiation versus inferencing.....	56
<b>Fig. 8.6</b>	Two-register synchronizer .....	57
<b>Fig. 8.7</b>	Good design partitioning .....	61
<b>Fig. 8.8</b>	Example bad and good partition.....	62
<b>Fig. 8.9</b>	Sample code for dealing with tristates at partition boundaries.....	62

<b>Fig. 8.10</b>	Divide and conquer approach to RTL design .....	62
<b>Fig. 8.11</b>	Combine sub-blocks to create an optimized design block .....	63
<b>Fig. 8.12</b>	Two-LUT levels between registers .....	67
<b>Fig. 8.13</b>	Use of pipeline stages to break up routing delays .....	67
<b>Fig. 8.14</b>	New data on simultaneous read/write.....	69
<b>Fig. 8.15</b>	Coding style that will infer a RAM that returns the OLD data on a simultaneous read/write .....	69
<b>Fig. 8.16</b>	Initialize the RAM contents to all 1 s.....	70
<b>Fig. 8.17</b>	Inferencing of a ROM.....	70
<b>Fig. 8.18</b>	Finite state machine .....	70
<b>Fig. 8.19</b>	Use of enumerated types in VHDL for state machine inferencing .....	71
<b>Fig. 8.20</b>	Verilog FSM .....	71
<b>Fig. 8.21</b>	State machine encoding styles .....	71
<b>Fig. 8.22</b>	Multiply-accumulate operation .....	73
<b>Fig. 8.23</b>	Verilog example of a register.....	73
<b>Fig. 8.24</b>	Register in VHDL.....	73
<b>Fig. 8.25</b>	Synthesis priority of secondary control signals for registers.....	74
<b>Fig. 8.26</b>	Multiplexer tree .....	74
<b>Fig. 8.27</b>	N:1 multiplexer.....	75
<b>Fig. 8.28</b>	Quartus II RTL viewer.....	77
<b>Fig. 8.29</b>	Quartus II state machine viewer .....	78
<b>Fig. 9.1</b>	Example detailing the use of parameters in a Verilog source file.....	84
<b>Fig. 9.2</b>	Sample GUI for IP demonstrated by the Quartus II Component Editor .....	88
<b>Fig. 10.1</b>	Sample from Header file generated by the Altera SOPC Builder tool.....	92
<b>Fig. 11.1</b>	Constrained random test flow.....	101
<b>Fig. 11.2</b>	Simple testbench that requires manual checking.....	102
<b>Fig. 11.3</b>	Example diagram of a self-checking testbench .....	103
<b>Fig. 11.4</b>	Verification system architecture .....	103
<b>Fig. 12.1</b>	Launch and latch edge diagram .....	110
<b>Fig. 12.2</b>	tsu and th diagram .....	111
<b>Fig. 12.3</b>	Clock arrival and data arrival diagram.....	111
<b>Fig. 12.4</b>	Multi-cycle path.....	112
<b>Fig. 12.5</b>	Input delay .....	113
<b>Fig. 12.6</b>	Output delay .....	113
<b>Fig. 12.7</b>	Clock uncertainty.....	115
<b>Fig. 12.8</b>	Bottom-up design flow .....	118

**Fig. 12.9** Integration of modules in the top-level design ..... 119

**Fig. 12.10** Example design partitioned for incremental compilation ..... 119

**Fig. 12.11** Example of the RTL viewer in the Quartus II software..... 125

**Fig. 12.12** Example view of a FSM from the Quartus II RTL viewer ..... 126

**Fig. 12.13** Critical Path View in Quartus II  
technology map viewer..... 127

**Fig. 12.14** The Quartus II chip planner detailing the  
Stratix IV ALM architecture ..... 127





# Chapter 1

## Best Practices for Successful FPGA Design

### 1.1 Introduction

This book which describes the Best Practices for successful FPGA design is the result of meetings with hundreds of customers on the challenges facing each of their FPGA design teams. By gaining an understanding into their design environments, processes, what works, what does not work, I have been able to identify the areas of concern in implementing System designs. More importantly, it has enabled me to document a recommended methodology that provides guidance in applying a best practices design methodology to overcome the challenges.

This material has a strong focus on design teams that are across sites. The goal being to increase the productivity of FPGA design teams by establishing a common methodology across design teams; enabling the exchange of design blocks across teams.

Best Practices establishes a roadmap to predictability for implementing system designs in a FPGA.

The three steps (Fig. 1.1) to predictable results are:

1. Proper project planning and scoping
2. Choosing the right FPGA device to ensure that the right technology is available for today's and tomorrow's projects
3. Following the best practices for FPGA design development in order to shorten the design cycle and to ensure that your designs are complete on schedule and that the design blocks can be re-used on future projects with minimal effort

All three elements need work together smoothly to guarantee a successful FPGA design.

The choice of vendor should be a long-term partnership between the Companies. By sharing roadmaps and jointly managing existing projects, you can ensure that not only is the current project a success but provide the right solutions on time for future projects. A process of fine tuning based on experience working together to guarantee success on projects.

These two topics are touched upon briefly in the Best Practices for Successful FPGA Design methodology.