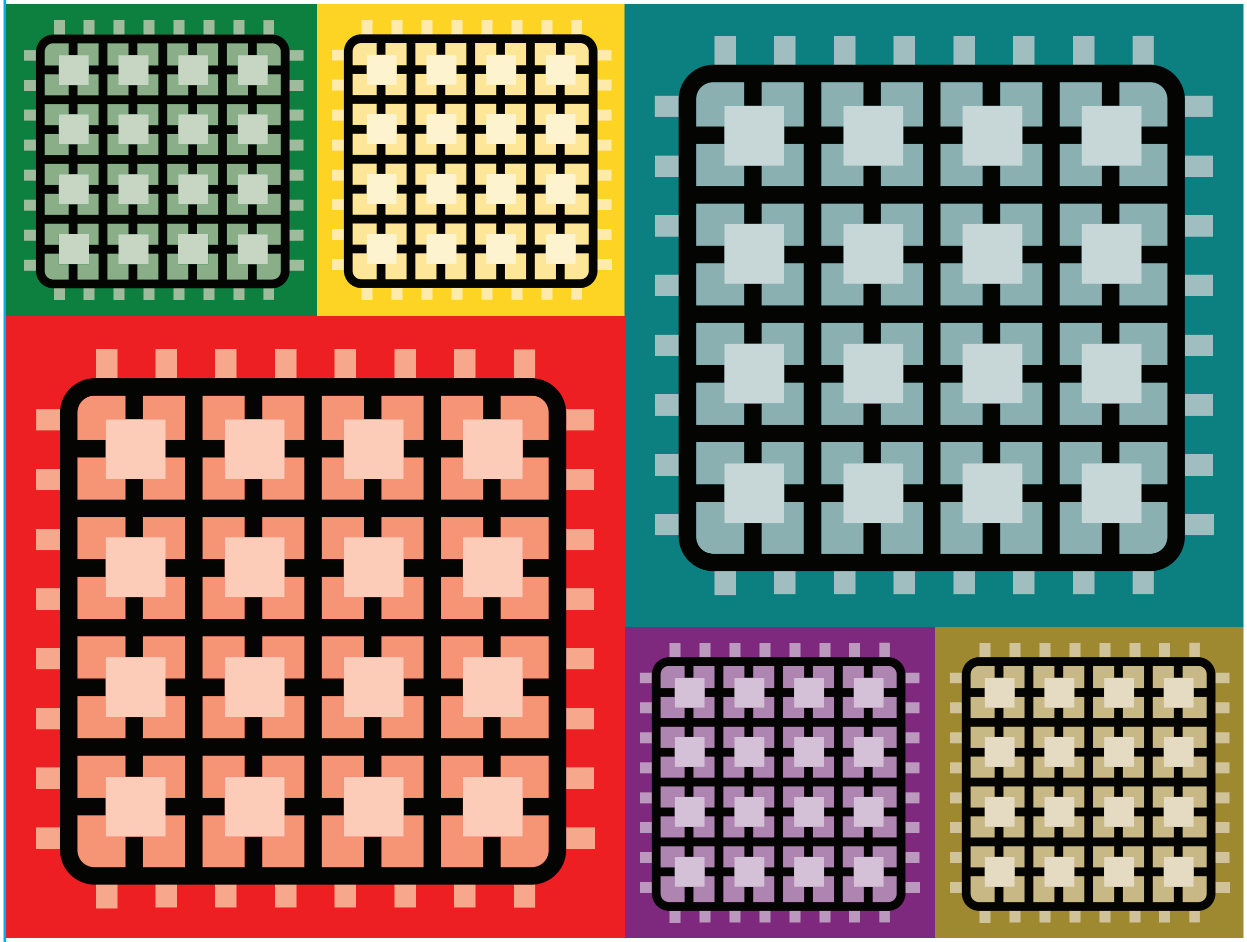
Make: FPGAs



Turning Software into Hardware with Eight Fun & Easy DIY Projects

David Romano

Make: FPGAs

Turning Software into Hardware with Eight Fun and Easy DIY Projects

David Romano

MAKER MEDIA™
SAN FRANCISCO, CA

Make: FPGAs

by David Romano

Printed in the United States of America.

Published by Maker Media, Inc., 1160 Battery Street East, Suite 125, San Francisco, CA 94111.

Maker Media books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://safaribooksonline.com*). For more information, contact O'Reilly Media's institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Editor: Roger Stewart
Production Editor: Melanie Yarbrough
Copyeditor: Rachel Head
Proofreader: Christina Edwards
Indexer: Judy McConville
Interior Designer: David Futato
Cover Designer: Brian Jepson
Cover Art: Shawn Wallace
Illustrator: Rebecca Demarest

February 2016: First Edition

Revision History for the First Edition

2016-02-26: First Release

See *http://oreilly.com/catalog/errata.csp?isbn=9781457187858* for release details.

978-1-457-18785-8

[LSI]

Table of Contents

Preface ix

1. **Overview** 1

Papilio 2

Opal Kelly 4

Red Pitaya 6

Numato Lab 8

Design Flow 11

  Concept Phase 11

  Design Phase 12

  Test Phase 15

  Synthesize Phase 17

  Build Phase 18

  Run Phase 18

Takeaways 19

2. **Count on It!** 21

Blink LEDs Concept 22

  How It Works 22

  Xilinx ISE WebPACK Installation 23

Design . . . 25
Peripheral Breadboard . . . 25
FPGA Circuit Schematic Design Entry . . . 27
FPGA Circuit HDL Design Entry . . . 45
Simulation . . . 60
Build . . . 65
Creating the Constraints File . . . 68
Open Kelly Setup . . . 72
Setup Test . . . 73
Clock Frequency Experimentation . . . 75
Takeaways . . . 76

3. That's Refreshing . . . 79

Stopwatch Concept . . . 79
How It Works . . . 80
Design . . . 82
Peripheral Breadboard . . . 83
FPGA Circuit . . . 84
Digital Clock Manager (DCM) . . . 87
Verilog Code and Concurrency . . . 90
Simulation . . . 98
Build . . . 100
Assigning Physical I/O . . . 100
Takeaways . . . 101

4. Testing 1, 2, 3, 4 . . . 103

The Test Bench . . . 103
Test Bench Anatomy . . . 104
Reuse . . . 105
Running the Test Bench Project . . . 105
Step 1: Selection and Download of Core . . . 105
Step 2: Documentation . . . 107

Step 3: RTL 109

Step 4: Adding Test Bench Files and Running the Simulation 112

Exploring the Test Bench Project 116

Overview 116

Takeaways 124

5. **It Does Not Compute 125**

The CARDIAC Computer Model 125

Getting Started with VTACH 128

Numato Elbert V2 Setup 130

Modifications 134

Step 1: Device Section 134

Step 2: Pin Assignments 135

Step 3: Clocking 137

Step 4: I/O Polarity 138

Step 5: Memory Block Update 141

Design, Build, and Simulation 145

Simulation 145

Building and Running 150

Programing and Assembler 151

Takeaways 153

6. **It's a Small World! 155**

System on Chip 155

SoC Architecture 157

DesignLab 160

Installation 160

Papilio DUO Setup 161

Step 1: Power Up 162

Step 2: Select COM Port 162

Step 3: Create Project 164

Step 4: Associate Circuit 164

Step 5: Load FPGA Bit File 166
Step 6: Compile and Upload Sketch 168
Getting Started with the DesignLab Video-Audio Player 169
How It Works 169
Design 171
Step 1: Create New DesignLab Project 171
Step 2: Edit Your Design in Xilinx ISE 172
Step 3: Add VGA Adapter Block 175
Step 4: Add Audio Blocks 175
Step 5: Implement and Generate Bit File 177
Step 6: Create Sketch, Load, and Run 178
Experiments 179
Source Code 181
Takeaways 181

7. Just for the Fun of It 183
Getting Started with VGA-Displayed Arcade Games 183
How It Works 184
Loading a Game 185
Source Code and ROM Files 189
Getting Started with LED Dot Matrix–Displayed Arcade Games 192
How It Works 193
Design 195
Experiments 197
Source Code 198
Takeaways 199

8. Cha-Ching! 201
Getting Started with the Bitcoin Miner 202
How It Works 203
Design 204
Source Code 207

Takeaways 207

9. **I Hear You!** **209**

Getting Started with the SDR Receiver 210

How It Works 212

Red Pitaya Setup 213

Loading the SDR 213

Step 1: Copy Red Pitaya SD Card Image 213

Step 2: Install SDR Applications on PC 214

Step 3: Connect Red Pitaya to the Network 214

Step 4: Run SDR Applications 214

Source Code 216

Takeaways 217

Appendix A. **FPGA Boards** **219**

Appendix B. **Papilio AVR Loading** **223**

Appendix C. **Text and Code Editor** **229**

Index **231**

Preface

Mention FPGAs to most people, and they will either give you a blank stare or think you are talking about some kind of golf league. To most of us Makers, the term conjures up thoughts of hardware creativity, exploration, and discovery, but many of you may have written it off as being way too complicated to even consider for your next project. This book is for you! It's all about learning what amazing, easy, and affordable projects you can construct with field programmable gate array (FPGA) technology. We will be doing this with hands-on experiments, in a fun and practical way.

This book is not a university textbook providing in-depth studies on hardware description languages (HDLs), HDL coding techniques, digital logic design theory, or validation methods. There are many very good resources both online and in textbook form that accomplish this goal. This will be more a learn-as-you-go experience. You can think of the book as a road map to a journey of design discovery, and I'll be your guide. But before we jump in, I want to give you a little background on the history of FPGAs.

Most of the technical community was first introduced to this exciting technology back in the '80s. I was a recent college graduate (RCG) with a degree in electrical engineering and had just been hired by a small telecommunications company that designed and manufactured modems and TI multiplexers. The company was developing a product that was, at the time, implemented using many 7400 series integrated circuits (ICs) and programmable array logic (PAL) devices. For many of you, this is probably like talking about funny animal paintings on cave walls. It wasn't the Stone Age of electronic design, but 7400 ICs and PALs are primitive compared to today's state-of-the-art systems-on-a-chip (SoCs). My new manager came to me one day with a data book in hand that he had just received from a company called Xilinx. He dropped the book on my desk and said, "Well, since you're the college kid, I want you to convert all of this logic design to an FPGA device," pointing to a large and very complex circuit board. Being a little naive and always eager to work on the latest cutting-edge technology, I said, "No problem, should be a piece of cake." It wasn't a piece of cake, but it was my introduction into the incredible world of field programmable array logic—sort of a baptism by fire.

A lot has happened in the past three decades in the world of digital design, and programmable logic devices (PLDs) are a big part of it. A PLD is an electronic component used to build reconfigurable digital circuits. Unlike logic gates, like those in the 7400 series ICs, which have fixed logic functions, a PLD has an undefined function at the time of manufacture. Before the PLD can be used in a circuit, it must be programmed (i.e., reconfigured). Before PLDs were invented, read-only memory (ROM) chips were used to create arbitrary combinational logic functions—talk about the Stone Age!

Today, there are many reasons a design team will consider FPGA technology in industry. For example, in many silicon IC design companies, FPGA-based platforms are used for what is called "shift left," where a new SoC device is mapped to FPGAs early in the design phase, in order to begin software integration long before the actual silicon device is manufactured. This is called "emulation" of the design. The big advantage here is that emulation runs orders of magnitude faster than simulation, so you can get some real-world hardware/software interactions very early in the validation phase (mostly on a functional level). The FPGA system typically operates at only a fraction of the silicon operating frequency, but the time saved in integration is tremendous.

Another example of where FPGAs are considered a viable solution in industry is where the design requires having multiple hardware personalities in the same footprint. For example, this was the case for a portable test and measurement instrument that I architected when I was a design engineer. By using an FPGA in the design, the customer was able to download different test instruments to the same hardware, essentially having multiple instruments in one hardware device.

The real question, then, for this book is: why would you, the do-it-yourself hobbyist or student, even consider experimenting with FPGAs? For students, it exposes you to contemporary digital logic design methods and practices in a fun, practical, affordable way. For the hobbyist, my goal is to show you how an affordable, off-the-shelf FPGA platform can be used in some very interesting and fun DIY projects. Many of you have had experience with Arduino or similar small microcontroller projects. With these types of projects, you usually breadboard up a small circuit, connect it to your Arduino, and write some code in the C programming language (which Arduino is based on) to perform the task at hand. Typically your breadboard can hold at best a few discrete components and one or two small ICs. Then you always need to go through the pain of wiring up the circuit and connecting it to your Arduino with a rat's nest of jumper wires. Instead, imagine having a breadboard the size of a basketball court or football field to play with and, best of all, no jumper wires. Imagine you can connect everything virtually. You don't even need to buy a separate microcontroller board; you can just drop different processors into your design as you choose. Now that's what I'm talking about! Welcome to the world of FPGAs.

FPGA History

Xilinx Inc. was founded in 1984, and as the result of numerous patents and technology breakthroughs, the company produced the first family of general-purpose, user-programmable logic devices based on an array architecture. It called this technology breakthrough the Logic Cell Array (LCA), and with this the Xilinx XC 2000 family of FPGAs was born.

You can think of an LCA as being made up of three types of configurable elements: input/output (I/O) blocks, logic blocks, and an interconnect matrix (see Figure P-1). From these, a designer can define individual I/O blocks that interface to external circuitry. You can think of these as configurable pins of ports. The designer can also use logic blocks, connected together through the interconnect matrix, to implement logic functions. These functions can be as simple as a counter or as complex as a microcontroller core. In a way the interconnect matrix is like the wires on a breadboard that connect everything together, but completely programmable.

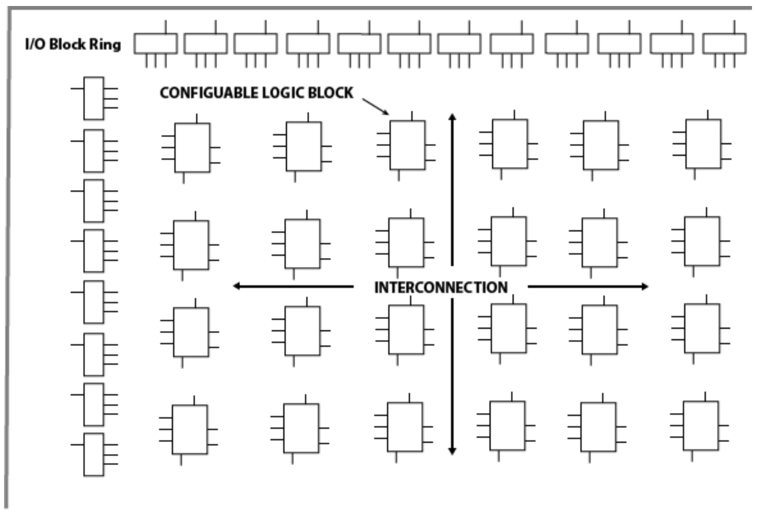


Figure P-1 *Xilinx LCA architecture*

Before there were FPGAs, you needed to use dozens of discrete ICs on a circuit board, or sometimes even hundreds of ICs on multiple circuit boards, to accomplish the hardware functionality you can achieve today with one FPGA device. For example, today you can create the entire Pac-Man game on a single FPGA device, including the game software. Now that's fun!

Xilinx and Altera are the two major players in the FPGA product space. Each of them provides a complete solution including a design tool suite. Altera came on the scene in 1992 when it introduced its first FPGA device family, the FLEX 10K line. There are pros and cons for each manufacturer, and many design wins come down to preference and price. For the purposes of this book, we will be focusing mainly on Xilinx, but the designs and experiments that follow will easily map to comparable Altera FPGAs, if you so desire.

The configuration of an FPGA device is accomplished through programming the memory cells, which determine the logic functions and interconnections. In the early days, the program (or what has become known as the *bit file*) was loaded at power-up from EEPROM, EPROM, or ROM on the circuit board, or loaded from a PC through a serial connection on the board from the FPGA programming tool. Since the underlying technology is volatile static RAM (SRAM), the bit file must be reloaded with every power cycle of the device. Today, SD flash memory replaces the EPROM and USB or JTAG replaces the serial connection, but the programming function remains much the same as it was in the beginning.

About the Book

This book is made up of eight interesting FPGA projects that will help you develop some of the skills you will need to really begin exploring this exciting world of turning software into hardware through FPGA technology. The projects will show you how to select an FPGA development board and set it up, and then give you the applications you will need to start making. The first chapter provides an overview of the boards and workflow we will be using in the book. The next two chapters walk you through a couple of simple projects, providing you with a hands-on look at the basics of the FPGA design flow. You can find the example code for these projects at my GitHub repository. The rest of the book concentrates on fun FPGA SoC projects. You can also check out my website for more information on learning with FPGAs. Always remember that learning is a lifelong adventure. I hope you enjoy the journey.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

`Constant width italic`
: Shows text that should be replaced with user-supplied values or by values determined by context.

This element signifies a tip, suggestion, or general note.

This element indicates a warning or caution.

Using Code Examples

Supplemental material (code examples, exercises, etc.) is available for download at *https://github.com/tritechpw/Make-FPGA*.

This book is here to help you get your job done. In general, if example code is offered with this book, you may use it in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from Make: books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Make: FPGAs* by David Romano (Maker Media). Copyright 2016 David Romano, 978-1-457-18785-8."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *bookpermissions@makermedia.com*.

Safari® Books Online

Safari Books Online is an on-demand digital library that delivers expert content in both book and video form from the world's leading authors in technology and business.

Technology professionals, software developers, web designers, and business and creative professionals use Safari Books Online as their primary resource for research, problem solving, learning, and certification training.

Safari Books Online offers a range of plans and pricing for enterprise, government, education, and individuals.

Members have access to thousands of books, training videos, and prepublication manuscripts in one fully searchable database from publishers like O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology, and hundreds more. For more information about Safari Books Online, please visit us online.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

> Make:
> 1160 Battery Street East, Suite 125
> San Francisco, CA 94111
> 877-306-6253 (in the United States or Canada)
> 707-639-1355 (international or local)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *http://bit.ly/make-fpgas*.

Make: unites, inspires, informs, and entertains a growing community of resourceful people who undertake amazing projects in their backyards, basements, and garages. Make: celebrates your right to tweak, hack, and bend any technology to your will. The Make: audience continues to be a growing culture and community that believes in bettering ourselves, our environment, our educational system—our entire world. This is much more than an audience, it's a worldwide movement that Make: is leading—we call it the Maker Movement.

For more information about Make:, visit us online:

> Make: magazine: *http://makezine.com/magazine*
> Maker Faire: *http://makerfaire.com*
> Makezine.com: *http://makezine.com*
> Maker Shed: *http://makershed.com*

To comment or ask technical questions about this book, send email to *bookquestions@oreilly.com*.

Acknowledgments

Many experiences led me to ultimately writing this book. Some were years in the making as I made my way through a very exciting and rewarding engineering career. Along the way, many people helped me learn and gave me great opportunities to explore and invent. I can't list them all here, but I would like to thank them. I also want to thank my publisher, Brian Jepson, for offering me this chance, and my editor, Roger Stewart, who

was so understanding throughout the writing process. I'd also like to thank Jack Gassett for all his help and support. Most of all I want to thank my wife, Elaine, for always believing in me and my Lord Jesus Christ for blessing me each step of the way.

Oh, and I can't forget Gracie the high-tech cat. She worked tirelessly by my side every day, overseeing the entire project.

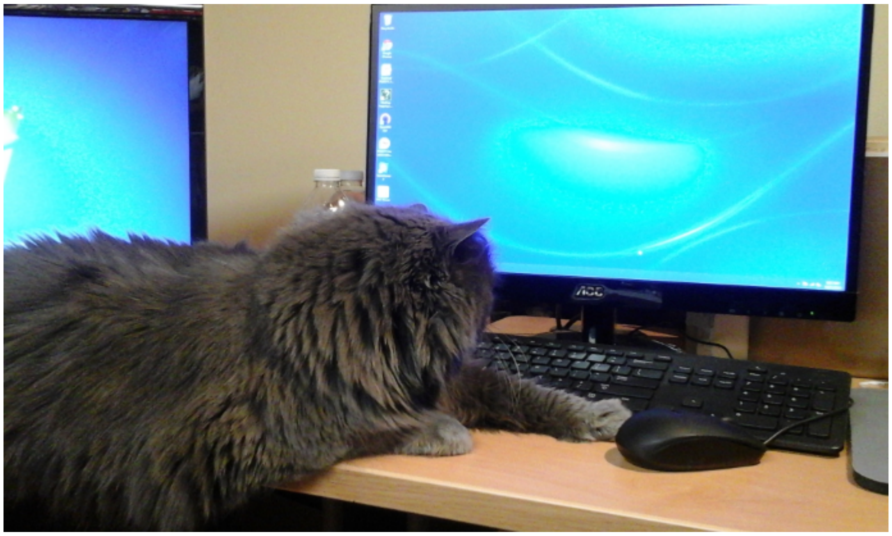


Figure P-2 *Gracie, the high-tech cat, hard at work!*

Overview 1

When it comes to off-the-shelf FPGA development boards, the sky is the limit. There are many options to choose from. Prices can range from under $50 to thousands of dollars. For this book, I chose to keep the price window between $50 and $200. After price, one of the primary things to consider when choosing an FPGA board is what you will be using it for. For most of us it will be general experimentation, but some of you may already have a specific project in mind.

When looking at FPGA boards we obviously want to know the vendor, family, and size of the FPGA device. A good rule of thumb is the larger the device, the more it will cost. The next thing to look at is what features the platform supports. The most important of these is external interface connectors. Obviously, if you can't connect anything to the board easily, you are very limited in what you can do with it. It's good to note the size (number of pins) and frequency rating (this is less important for us) of the interfaces provided. Also note if the interfaces allow for connection to standard-form-factor add-on modules like the popular Ardunio shields, Digilent's Pmods, and the Papilio Wings. Other interface options to consider are the type and number of onboard standard communication interfaces that are provided, such as USB, Gigabit Ethernet, HDMI/DVI, PCI/PCI Express, external nonserial memory (DDR/flash, etc.), SD card, I2C/SPI, VGA, UART, etc.

FYI

Xilinx is standardized on the FPGA Mezzanine Connect (FMC), which is an industry standard. There are a lot of companies that make FMC-based plug-in cards.

I also like to have some number of onboard LED, switches, buttons, or even a 7-segment display, which are all very handy to have on the baseboard. These features can help you with the initial bring-up of the module, as we shall see in the next chapters, and also with the learning curve of the platform. They can also provide a convenient and cost-effective way to build some very basic experiments without the hassle of ordering additional boards or breadboarding your own circuits.

The other big thing to consider is the development tools that you will need to design and program your FPGA with. For our purposes, the tool license should be free of charge and provide a robust design entry method that supports both schematic capture and HDL (VHDL and Verilog) input. The tool should provide some basic simulation capability and a solid synthesis engine with good output reporting. Connection to the FPGA device should be easy with a standard interface like USB, and uploading to the FPGA should be done easily through a PC. OS support should include Windows and Linux with Mac as an option.

One other thing to consider on the FPGA platform is if there is some type of microcontroller integrated onboard or onchip. The Papilio DUO and Xilinx Zynq are examples of these. Having a local microcontroller available opens up another great dimension for creativity and exploration. In Appendix A, you will find a list of many low-cost FPGA boards that you can choose from.

In the rest of this chapter, I will review a few boards that I think have some unique features. The features that I'll be highlighting may provide you with some interesting opportunities for innovation.

Papilio

For the FPGA hobbyist and DIYers, Papilio by Gadget Factory is second to none. Everything about these products, from the array of affordable hardware modules to the innovative design environment and abundant learning material, was created with you in mind. As you can tell, I'm over the top with these guys. I felt like a kid on Christmas morning when I opened the box the day my shipment of Papilio products arrived. There were so many options to play with, I really didn't know where to begin. Now that's impressive!

So what is Papilio?

Papilio is a series of FPGA development boards and add-on hardware application modules called "Wings" that plug into the main board—sort of like Arduino shields.

As the Gadget Factory website points out, Papilio is Latin for butterfly:

> Papilio conveys the ability of an FPGA to undergo "digital metamorphosis." FPGA technology allows the Papilio to become any type of digital circuit including microcontrollers and custom chips such as the Commodore 64 audio (SID) chip.

You clearly get a sense of the creativity level that we are dealing with from that. This is going to be fun!

Papilio FPGA boards come in a range of affordable options, from the $37.99 Papilio One to the $87.99 Papilio DUO. You read that right: you can get into experimenting with FPGAs for as little as $37.99.

Did You Know?

With the $37.99 Papilio One 250K, you can start experimenting with FPGAs using the Xilinx XC3S250E, a 5.5K logic cell device. You also get onboard Papilio Wings connectors, a two-channel USB connection, and 48 general-purpose input/output (GPIO) pins.

I will start at the other end of spectrum, with the $87.99 Papilio DUO. This module is unique because it combines an Arduino-compatible microcontroller with a Xilinx FPGA on the same board. Not only does the board provide the ability to connect to the optional Papilio Wings modules, it also provides full compatibility with the Arduino shield ecosystem through the standard onboard Arduino connectors. There's even support for a "Pmod" (Peripheral Module interface, an open standard defined by Digilent Inc. in the Digilent Pmod™ Interface Specification for peripherals used with FPGAs or microcontrollers) connector on the board. Talk about add-on options!

The Papilio DUO uses the Xilinx Spartan-6 XC6SLX9 FPGA as its core. Table 1-1 describes the features of this FPGA device.

Table 1-1 *Spartan-6 XC6SLX9 features*

XC6SLX9 feature	Description
9K logic cells	Logic cell ratings are intended to show the logic density of one Xilinx device as compared to another device. Like a logic block, the typical cell includes a couple of flip-flops, multiplier, logic gates, and a small amount of RAM for a configurable lookup table (LUT). The logic cell is normalized to a 4-input lookup table (LUT).
576 Kb of block RAM	A block RAM (BRAM) is a dedicated two-port memory block containing several kilobits of RAM. The FPGA contains several (or many) of these blocks. In the Spartan-6 family, the block size is 18 Kb and the 6SLX9 has 32 of these, so the total size is 576 Kb.
2 CMTRs	There are two clock management tiles (CMTs) in the 6SLX9. Each CMT contains two digital clock managers (DCMs) and one phase-locked loop (PLL). The DCM core is a very versatile and complex piece of Xilinx intellectual property (IP). It can be used to implement a delay locked loop, digital frequency synthesizer, a digital phase shifter, or a digital spread spectrum.

XC6SLX9 feature	Description
TMDS I/O	Transition minimized differential signaling (TMDS) I/O support means that DVI and HDMI interfaces can be implemented directly with the FPGA I/O pins without any extra chips.
16 DSP slices	There are 16 digital signal processor (DSP) slices in the 6SLX9.

You can check out the full data sheet for the Xilinx Spartan 6-LX9 on the Xilinx website.

Opal Kelly

I chose to feature Opal Kelly in this book because of their unique approach to bridging the gap between FPGA design/development and interconnection with a PC or other computing device using USB or PCI Express. For the hobbyist, this opens some very interesting opportunities for innovation. Opal Kelly offers a full product line of FPGA modules ranging in price from the very affordable XEM6001-6002 at $174.95 to the $1399.95 XEM5010-50M256. We will be using the XEM6002 for our experiments. This module uses the same Xilinx Spartan-6 XC6SLX9 FPGA as the Papilio DUO for its core. This FPGA contains 9K Logic cells, 576Kb of Block RAM, 16 DSP slices, and two CMTs. For details on this device, see the previous section.

The Opal Kelly XEM6002 module is also equipped with four Digilent Pmod-compatible connectors, as seen in Figure 1-1, which allow for interfacing to a wide variety of low-bandwidth peripheral modules available from several semiconductor manufacturers. Think of the possibilities!

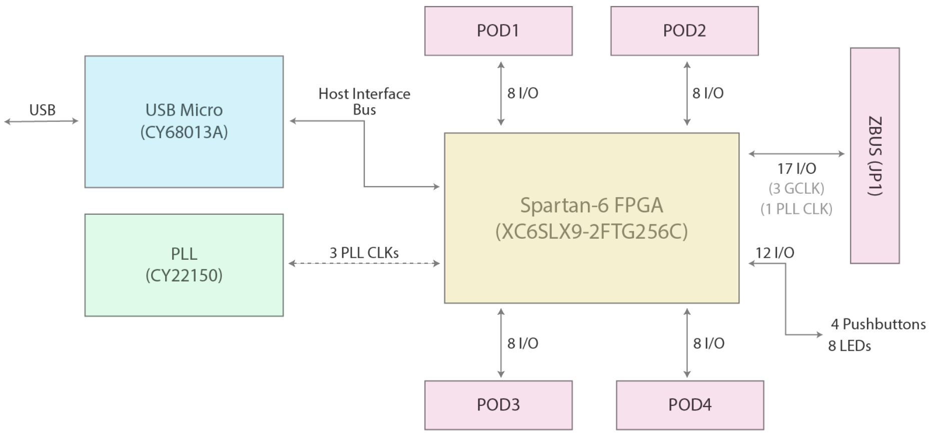


Figure 1-1 *XEM6002 module block diagram*

Check out the Digilent Peripheral Modules page for a look at some of the fun and interesting plug-in modules you'll have access to.

Did You Know?

Pmods are small I/O interface boards that offer a great way to extend the capabilities of an FPGA or embedded controller board. Pmods communicate with system boards using standard 6- or 12-pin connectors. In addition to various sensors and connectors, there are Pmods for I/O, data acquisition and conversion, external memory, and much more.

The key to the Opal Kelly innovation is the FrontPanel SDK. The software development kit, or SDK, provides a small FPGA library block that integrates with your FPGA design to make USB (or PCI Express) host communication simple and easy. It also includes a software API, simplifying the programming development of the communication interface, and a robust driver to communicate with your device over USB or PCI Express (see Figure 1-2). The USB driver and FrontPanel API work together to provide an easy-to-use software interface to your hardware that is consistent across the Windows (32-/64-bit), Linux (32-/64-bit), and Mac OS X development environments.

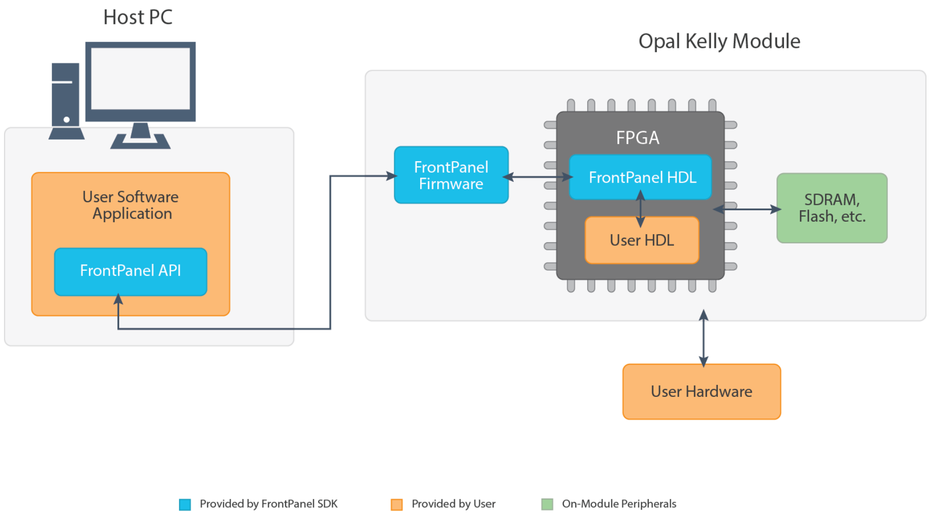


Figure 1-2 *Opal Kelly FrontPanel SDK*

Opal Kelly also provides prebuilt wrappers to the FrontPanel API for C, C#, C++, Python, and Java, and the DLL can be used from any software that allows external calling, such as MATLAB or LabVIEW.

The standalone FrontPanel application lets you quickly and easily define your own graphical user interface (GUI) that communicates with your hardware. It supports a variety of user interface elements, including LEDs, hexadecimal displays, sliders, push buttons, checkboxes, toggle buttons, and numerical entry. You can think of the FrontPanel as a virtual breadboard environment.

Wow, A Virtual Breadboard Sandbox!

With the Opal Kelly FrontPanel virtual interface you can quickly and easily create a GUI of an instrument right on your PC that interfaces with your FPGA design! How cool is that?

Red Pitaya

The Red Pitaya platform, according to its developers, is an "open-source measurement and control tool replacing many expensive laboratory instruments." It is based on the Xilinx Zynq-7010 programmable SoC device, which combines a dual-core ARM® Cortex™-A9 MPCore processor with a programmable logic array that contains 28K logic cells, 240 Kb of block RAM, and 80 DSP slices. The processing subsystem operates autonomously from the programmable logic, "booting on reset like any other processing device." It acts as the "system master" and controls the configuration of the programmable logic, enabling full or partial reconfiguration during operation.

The retail cost of the Red Pitaya platform at the time of writing is $238.80, which makes it one of the higher-priced platforms of the group being used for this book—and you will need more than just the board to get going. Figure 1-3 shows the Red Pitaya v1.1 board and required accessories. If you are planning on using scope probes, you will need to purchase a couple of those and a couple of SMA male to BNC female adapters (the adapters are required to connect standard BNC-type oscilloscope probes because Red Pitaya uses SMA connectors on the board). You will also need a 5V, 2A, USB micro, power supply, and a 4–8 GB micro SD card. All of this is conveniently offered on the Red Pitaya website as a Red Pitaya website as a Diagnostic Kit for a grand total of $310.80, not including any taxes or shipping.

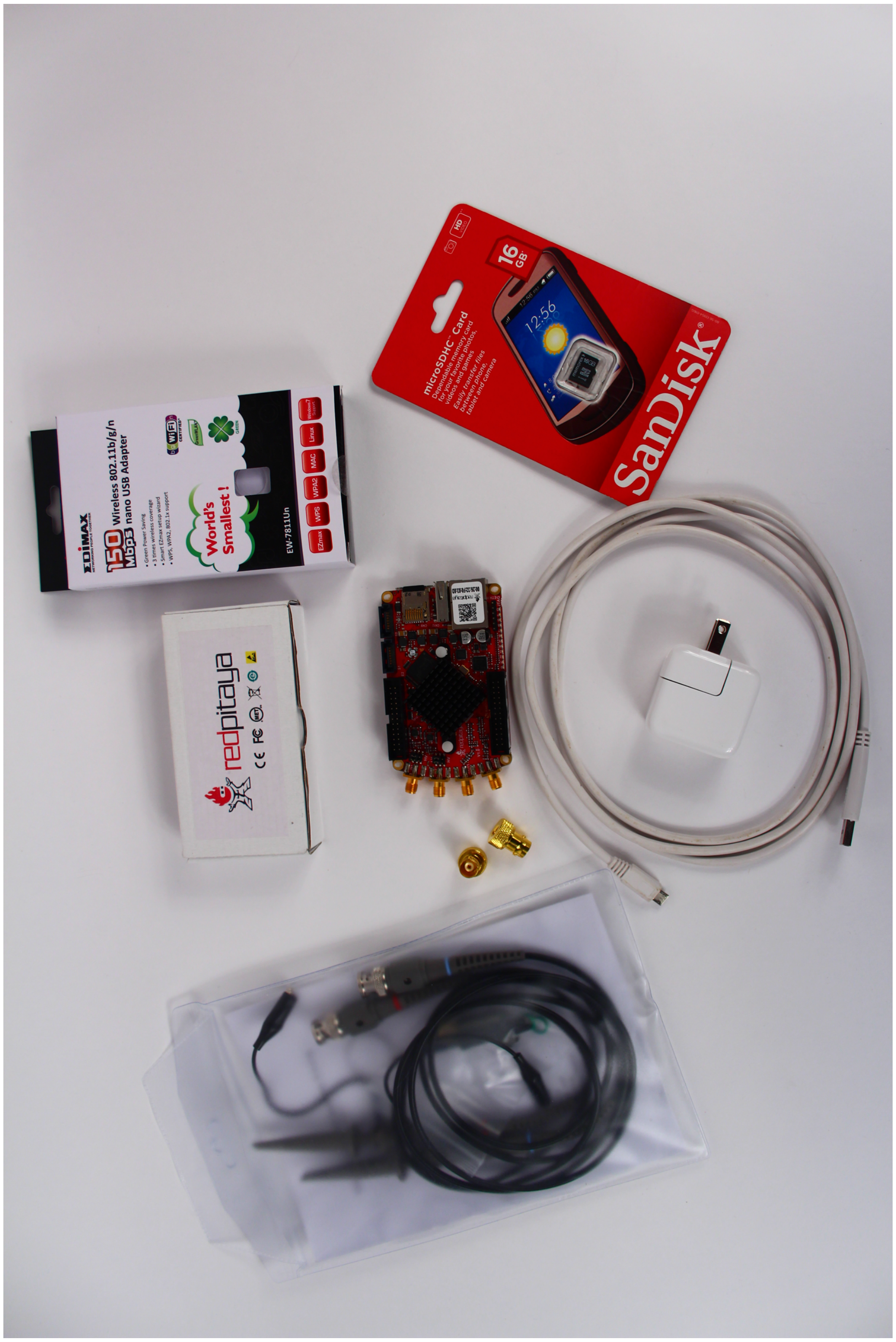


Figure 1-3 *Red Pitaya board and accessories*

Did You Know?

The ARM A9 core is the application processor used in many tablets and cell phones. It is a high-performance, low-power microprocessor core.

What is really impressive about this platform are the three major technology subsystems that you have access to. The first is the RF/analog frontend, which gives you the ability to instantiate a true oscilloscope, spectrum analyzer, or signal generator. The second is the ARM CPU core, and the third is the FPGA itself. You can purchase premade instrument files (Oscilloscope + Signal Generator or Spectrum Analyzer) from the Red Pitaya online store. Check out the Detail specification page on the Red Pitaya wiki for a full description of the frontend hardware.

That being said, let's crack open the box and see what's inside (Figure 1-4). Like on a first date, first impressions are lasting ones. I must admit I was impressed by the small size of the platform, measuring only about 4 × 2.5 inches (the website calls it a "credit card footprint").

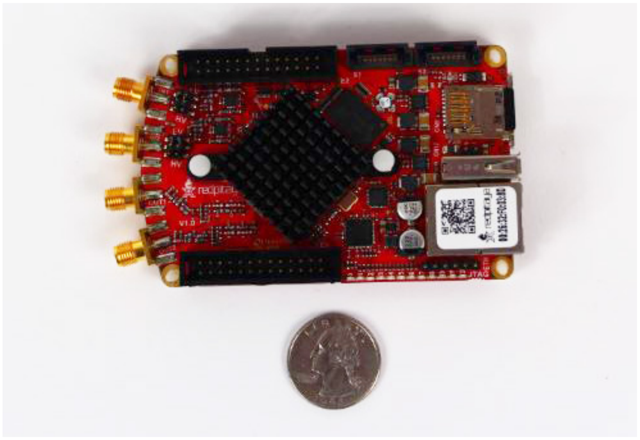


Figure 1-4 *The impressive small size of the Red Pitaya platform*

Numato Lab

Numato Lab provides a complete product line of low-priced FPGA development boards and accessories. But just because the prices are low doesn't mean that they are short on features—on the contrary, the bang for your buck you get with these products is amazing. I was able to take a look at three of the Numato FPGA platforms, pictured in Figure 1-5:

- The $29.95 Elbert V2 - Spartan 3A FPGA Development Board
- The $49.95 Mimas V2 Spartan 6 FPGA Development Board with DDR-SDRAM

- The $199.95 Waxwing Spartan 6 FPGA Development Board

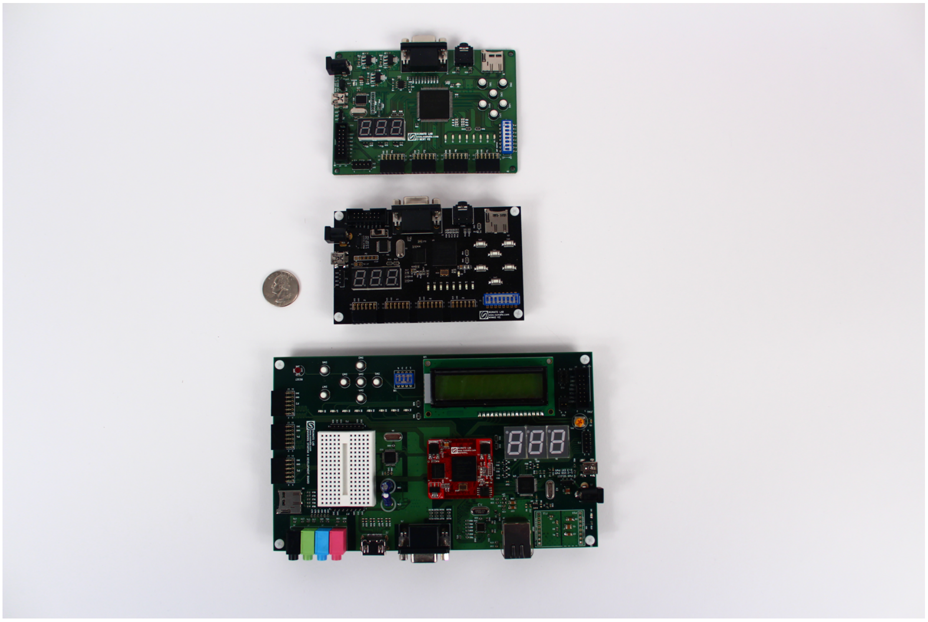


Figure 1-5 *Numato Lab Elbert V2 (top), Mimas V2 (middle), and Waxwing (bottom)*

That's right—for just $29.95 you can jump into working with FPGA technology! I may sound like a used car salesman, but this is not a barebones FPGA board. Take a look at the Elbert V2 feature list:

- FPGA: Spartan XC3S50A in TQG144 package
- 16 Mb SPI flash memory (M25P16)
- USB 2.0 interface for onboard flash programming
- FPGA configuration via JTAG and USB
- 8 LEDs, six push buttons, and an 8-way DIP switch for user-defined applications
- VGA output
- Stereo audio out
- Micro SD card adapter
- Three 7-segment displays
- 39 I/Os for user defined-purposes
- Onboard voltage regulators for single power rail operation

This is a remarkable value, as you get almost everything you need: LEDs, push buttons, DIP switches, and even three 7-segment displays. The only downside is that the Spartan XC3S50A is a 1.5K logic cells device, compared to the Spartan-6 LX9 FPGA, which is a 9K logic cell device.

If FPGA size is an issue, for another $20.00 you can jump up to the Mimas V2, which has a Spartan-6 XC6SLX9 and also includes 512 Mb of DDR memory along with all the other features of the Elbert.

For $199.95, the Waxwing development board should really be called a development laboratory. This board has everything you can imagine on it:

- Spartan-6 FPGA (XC6SLX45 in CSG324 package on Waxwing Mini Module)
- 166 MHz 512 Mb LPDDR
- 100M Ethernet (LAN8710A)
- National Semiconductor LM4550 AC '97 audio codec
- 128 Mb SPI flash memory (W25Q128FV)
- 100 MHz CMOS oscillator
- DVI-D connector for video (compatible with HDMI)
- 8-bit VGA output connector
- 2 channel audio out through 3.5 mm audio jack
- 16 × 2 character LCD display
- Micro-SD adapter
- 3 common anode 7-segment LED displays
- 7 onboard push button switches
- 44.5 × 35.1 mm mini breadboard for easy prototyping
- High-speed USB 2.0 interface for onboard flash programming
- FT2232H channel A dedicated for SPI flash programming; channel B can be used for custom applications
- FPGA configuration via JTAG and USB
- Onboard voltage regulators for single power rail operation

They even provide you with a mini-solderless breadboard, right on the board. This is really an experimenter's dream—you don't need to buy any additional add-on modules to do some really cool experiments.