Embedded Systems Design
with Platform FPGAs
Principles & Practices

Ron Sass and Andrew G. Schmidt
http://www.rcs.uncc.edu/~rsass
University of North Carolina at Charlotte
Spring 2011
Chapter 1 — Introduction
Chapter 1 Learning Objectives

Topics

- embedded systems concepts
- programming hardware and software
- challenges that embedded system designers face
- FPGA characteristics
a Field Programmable Gate Array, or FPGA, is

in other words, it is a device whose function is not fixed at manufacture but rather can be configured (and often repeatedly reconfigured) after it has been installed.
FPGA — a “Blank Slate”

Blank “slate” (when powered on) → After configuration
a Platform FPGA is physically identical to an FPGA.

we use Platform FPGA to characterize a device.

the aim of this class is to provide a foundation for building embedded systems on a Platform FPGA device.
Embedded Systems

- What is a computing machine?
- How is embedded different from general-purpose?
- Mixing hardware and software — different execution models
Abstract Computing Machine

- first of all, it is a machine
  - first were mechanical
  - later electro-mechanical
  - nowadays, electronic

- environment provides
  - inputs
  - energy

- machine produces
  - outputs
  - heat
Abstract Computing Machine Illustrated

![Diagram of an abstract computing machine showing inputs, outputs, energy, and heat. The environment is depicted as a box with arrows indicating energy input and heat output.](Image-Link)
Computing Machine

- *computing machine*

- implicit in the machine is an encoding that gives meaning to the inputs and outputs
- the modern definition includes the ability to be controlled by a stored program
Introduction

Hardware versus Software

- **hardware**
  - the stuff that exists in the physical world
  - you can touch it — it is concrete

- **software**
  - it is a set of rules, directives, commands that operates the machine
  - as such it is abstract and doesn’t exist in the physical world

\[1\] although physical representations of software, such as print-outs, do exist
Introduction

Software Terms

• a **program** —

  • for example, an assignment might be to create a program
  • the language (C or MATLAB, for example) defines the syntax and semantics of valid programs

• as a verb (in the software world) **programming** refers to act of creating software; i.e.
  • “I will be programming until lunch.”
  • or “He was programming the computer.”
FPGAs are part of a family of devices that are collectively referred to as *programmable logic*. It makes sense as a noun because these devices leave the factory with no fixed function. However, the early devices didn’t use a language — the engineer simply picked which fuses this can lead to some confusion because...

In the hardware world, *programming* as a verb usually refers to the act of storing data in a non-volatile device or transferring a configuration.

- “Use this device to program your EEPROM.”
- “Wait until the FPGA has been completely programmed before starting the motor.”

To avoid confusion, we prefer to use the verb *configure* to refer to the act of programming an FPGA.
Modern Field-Programmable Gate Arrays further muddy the waters...

- modern FPGAs have *millions* of configuration bits
- simply not practical to manually set the configuration bits
- instead, designers use a Hardware Description Languages (HDL)

  - an **HDL** is

  - as such, it shares the same characteristics as *software* (it is abstract, has syntax, and semantics)
  - but designers still typically *call it* hardware!
Hardware Design Terms

- In the case of FPGAs, “hardware” is often used as shorthand for “hardware design”

- A (hardware) **design** is

- A **core** is
  - Could be as simple as a “multiplier core”
  - Or as complex as a “processor core”

- Organizations that sell cores will call them IP cores where IP stands for Intellectual Property
FPGA-specific Design Terms

- A **hard core** in FPGA lingo refers to a core that has been implemented in CMOS transistors and provides a single fixed-function resource on an FPGA.

- A **soft core** is a core that has been implemented in the programmable logic of an FPGA.

**NOTE:**
- Note that this is different from the ASIC world where they mean something else.
- At one time, a vendor referred to hard core as a **diffused IP core**.
- Sometimes, **block** is synonymous with core (as in, “your design requires a multiplier block”).
- **Hard macros** are something completely different (we’ll introduce them later).
Embedded versus General-Purpose

- an **embedded computing system** (or just *embedded system*) is

- in contrast, a **general-purpose computing system** is a computer that is an end-product of itself

<table>
<thead>
<tr>
<th>General Characteristics</th>
<th>General-Purpose</th>
<th>Embedded</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>standard peripherals</td>
<td>specialized peripherals</td>
</tr>
<tr>
<td></td>
<td>3rd party software</td>
<td>single vendor software</td>
</tr>
<tr>
<td></td>
<td>purchased as a ‘computer’</td>
<td>product has different name (i.e., “phone”)</td>
</tr>
</tbody>
</table>
Embedded Systems Examples

Mobile Phone

Security Camera

Payment Capture
since embedded systems are part of a larger product,
they interact with wide range of specialized peripherals
(application-specific actuators, sensors)
often require the functionality to be split between hardware
and software components
however, hardware and software use different execution
models
Sequential Model

Introduction
Custom R0 + R1 + R2 Unit

Sequencer

4-bit adder

4-bit adder

Acc
Execution Models Explained

- **sequential**
  - uniform time steps
  - (typically) one instruction completes before next is started
  - data movement is general

- **dataflow**
  - data may not move uniformly through system
  - data flows along pre-assigned (or restricted) paths between computation units
Sequential v. Dataflow Timing

**Sequential**

- **fetch** Instruction CLR Acc
- **execute**
  - assert CLR on Acc register;
  - advance PC

- **fetch** Instruction ADD R0
- **execute**
  - assert RD, select 00 on reg. file;
  - assert read;
  - select function ADD on ALU
  - latch result on Acc

- **fetch** Instruction ADD R1

**Data-flow**

- latch register to

- latch register Acc

---

we will return to this concept multiple times
designing embedded systems is more challenging than programming ordinary applications
- software starts at power up and must run until powered off
- products usually have a hardware design component
- an additional hardware/software integration step
- debugging can be more difficult

moreover, the embedded systems are evaluated over a larger range of metrics than ordinary applications

and, finally, the costs of completing an embedded systems project tend to be more complex

Before we show how Platform FPGAs can be useful to embedded systems designers, we will review the challenges that embedded systems designers face.
Project Management

- often embedded systems designers are involved with all aspects of the project
- this can include several non-technical tasks such as
  - project management
  - melding different approaches to development
  - finance
- as we close the first chapter, we briefly visit some of topics
there many different development models; the “waterfall” model is probably the simplest

basic ideas:

- complete each stage before advancing
- the cost of going backwards (up the waterfall) to fix a — for example, a requirements mistake — is very steep

mostly, the model helps a designer consciously think about organizing the project’s activities
Hardware and Software Revision Rates

- **Initial SW**
- **Initial HW**
- **Effort**
- **Major revision**
- **Minor revisions**
- **Final revision**
- **2nd revision**
- **1st revision**
- **Time**

![Diagram showing the progression of hardware and software revisions over time](image-url)
Introduction

Development Philosophies

- embedded system projects tend to bring different groups of people together with different approaches
- philosophies
  - hardware designers usually have very expensive non-recurring costs; they tend to test extensively and revise cautiously
  - software developers have the advantage of fast turn around (i.e. compilation time is fast); they tend revise quickly
- on traditional embedded systems projects, these philosophies can clash but the two groups have to be tightly coupled (for example, some software development can’t begin until the hardware board is delivered)
- Platform FPGA projects have the opportunity to blend these approaches
programmers and engineers that let deadlines slip are sometimes ignorant of the costs associated with it

- some products (such as consumer electronics) have annual cycles that are critical to sales
  - i.e., bring a product to market before the end of year Holidays is extremely important
- many projects are financed which means that additional work days are incurring additional finance charges
- some products have a peak demand — missing the window means that some potential sales are lost forever

at the least, designers should be cognizant of the issues if not actively involved in these aspects of the project
Costs Illustrated

Profit ($)

On-time

Delayed

Time

Labor

A

B

C

W

2W
Chapter 1 addressed:

- *What is an embedded system?*
- *Why are embedded systems different?*
- *How does Platform FPGAs help?*

Specifically, the topics covered:

- embedded systems concepts
- programming hardware and software
- challenges that embedded system designers face
- FPGA characteristics
Chapter 1 Terms

FPGA

Field Programmable Gate Array

Platform FPGA

An FPGA device that includes sufficient resources and functionality to host an entire system on a single device.

Computing machine

A device consisting of a control or processing mechanism that responds to inputs by signaling its outputs; implicit in the machine is an encoding that gives meaning to the inputs and outputs.

Embedded computing system

An computing system that is integral to a larger, enclosing product; in contrast to general-purpose computing, the embedded computing system is not intended to be an end-product of itself.

Embedded Systems Design with Platform FPGAs
Chapter 1 Terms

**General-Purpose Computing System**
A product itself and, as such, the end-user directly interacts with it.

**Hardware**
Refers to the physical implementation of a computing machine.

**Software**
A specification that describes the behavior of the machine, generally written in a programming language (such as C, MATLAB, Java, etc.).

**Program**
Software written in a specific programming language that is the representation of desired machine behavior.

**Processor**
Hardware that implements the sequential execution model.

**Design**
The digital circuit that is programmed (or configured) into an FPGA.
Chapter 1 Terms

hardware description language (HDL) is a programming language used to describe the behavior of hardware and is the most common form of design entry today.

IP core: intellectual property that refers to a hardware specification which, depending on how it is expressed, can be used to manufacture an integrated circuit (hard core) or configure the resources of an FPGA (soft core); diffused IP core is a hard core embedded in an FPGA integrated circuit.

hard core: an FPGA-specific term for an IP core where the logical operations and interconnection have been specified and mapped to the components of particular device; thus the core has a fixed shape and location on the chip.
Chapter 1 Terms

**soft core** is an FPGA-specific term for an IP core where the logical operations and interconnection is specified but these operations have not been mapped to a particular device module.

**hard block** is a core that has been implemented in CMOS transistors and **soft block** is a core that has been implemented in the function generators and memories of an FPGA device.