Welcome to CSE467!

- Course Staff: Bruce Hemingway and Ryan McElroy
- My office: CSE 464 Allen Center, 206 543-6274
- Today: Course overview
  - What is computer engineering?
  - What we will cover in this class
  - What is “design”, and how do we do it?
  - Basis for FPGAs
  - The project- develop 5 megapixel digital camera processor

Highlights:

- We'll be reading hand-outs and papers from various sources.
- The course work will be built around an FPGA in the Altera DE-1 board.
- Tools are contained in Altera Quartus.
- Language is verilog.
- Applications in the FPGA will include image processing.
What is computer engineering?

- CE is not PC design
  - It includes PC design
- CE is not necessarily digital design
  - Analog computers
  - Real-world (analog) interfaces
- CE is about designing information-processing systems
  - Computers
  - Networks and networking HW
  - Automation/controllers (smart appliances, etc.)
  - Medical/test equipment (CT scanners, etc.)
  - Much, much more

What we will cover in CSE467

- Basic digital design (much of it review)
  - Combinational logic
    - Truth tables & logic gates
    - Logic minimization
    - Special functions (muxes, decoders, ROMs, etc.)
  - Sequential logic
    - Flip-flops and registers
    - Clocking
    - Synchronization and timing
  - State machines
    - Counters
    - State minimization and encoding
    - Moore vs Mealy
What we will cover (con’t)

- Advanced topics
  - Field-programmable gate arrays (FPGAs)
  - RTL design
  - High-speed DSP design
- HDLs and synthesis
  - Verilog
  - Synthesizing to an FPGA
  - Re-usable code modules-- IP
- An imaging design project

CSE467

CSE467 is about design

- Design is an art
  - You learn by doing
  - Takes years of practice
  - This class is a starting point
- Engineering is learning how to solve problems
  - Independent of any specific technology
  - Independent of any specific tools
- *Imagination is more important than knowledge* – Einstein

CSE467
Example: Design an encryptor

- **DES**
  - Input is clear-text (or plain-text)
  - Output is cipher-text
  - Same algorithm encrypts and decrypts

- **Details**
  - IP is initial permutation
  - L is left-half of message
  - R is right half of message
  - 16 iterations on the message
  - K are keys
  - f is a function (next slide)
  - IP\(^{-1}\) is an inverse permutation

---

One round (iteration) of DES

- This block represents one of the 16 iteration steps
- Standard DES operates on 64-bit blocks (eight 8-bit bytes)
- The key is 56 bits
  - \(K_1 \sim K_{16}\) are 16 permutations on the master key
- Use lookup tables for the permutations and substitutions
DES in hardware

- Standard DES algorithm
  - Turns a 64-bit message block into a 64-bit cipher block
  - Electronic Code Book (ECB) mode
    - Each 64-bit block is encrypted independently

- Imagine you have to design a fast encryptor
  - For a continuous data stream
    - e.g. satellite communications or wireless LAN or ...
  - How would you implement DES?

Hardware DES

- Options are
  - Discrete logic (fixed-function ICs)
  - PLDs & discrete logic
  - Microprocessor / microcontroller / DSP
  - FPGA
  - Semicustom ASIC
  - Custom IC

- How do you choose?
DES with discrete logic? PLDs?

- How?
  - Large PCB
  - Discrete FSM
  - Leverage PLDs

- Advantages
  - None

- Disadvantages
  - High parts cost
    - Many ICs
    - Complex PCB
  - High design cost
  - Slow throughput
  - Hard to change algorithm or fix design errors

DES using a \(\mu\)P

- How?
  - Use COTS development system
  - Or a custom PCB with \(\mu\)P, memory, firmware, glue logic

- Advantages
  - Easy to change algorithm or fix design errors (rewrite firmware)
  - Low design cost

- Disadvantages
  - High parts cost
  - Slow throughput
  - Serial execution
DES using an FPGA

- **How?**
  - PCB with one or several FPGAs
  - Include glue logic in FPGAs

- **Advantages**
  - Easy to change algorithm or fix design errors (resynthesize logic)
  - Moderate throughput
    - Pipelined logic; 25 – 75 MHz clock

- **Disadvantages**
  - Moderate parts cost
    - Okay for small production runs
  - Moderate design cost
DES using an ASIC

How?
- A single ASIC--Include glue logic

Advantages
- Fast throughput
  - Pipelined logic
  - 50 – 300 MHz clock
- Low parts cost for volume production
- Moderate design cost
  - Debug using FPGAs
  - Synthesize to an ASIC

Disadvantages
- Difficult to change algorithm or fix design errors
  - Redesign chip

DES as a custom IC

How?
- A single custom IC
  - Include glue logic in IC

Advantages
- Ultra-fast throughput
  - Pipelined, tuned logic
  - 200 – 1000 MHz clock
- Low cost for volume production

Disadvantages
- Difficult to change algorithm or fix design errors
  - Redesign chip
  - Long, slow, expensive design process
FPGA Basics

Field **PROGRAMMABLE** Gate Array

Adapted from:
The Design Warrior’s Guide to FPGAs
Devices, Tools, and Flows. ISBN 0750676043
Copyright © 2004 Mentor Graphics Corp.
(www.mentor.com)

A Simple Programmable Function

![Diagram of a simple programmable function with AND gate and pull-up resistors.]

Potential links

Logic 1

Pull-up resistors

\( y = 1 \) (N/A)
Unprogrammed Fusible Links

Logic 1

Fuses

Pull-up resistors

y = 0 (N/A)

AND

Programmed Fusible Links

Logic 1

Pull-up resistors

y = a & !b

AND
Unprogrammed antifuse links

Unprogrammed antifuses

Logic 1

Pull-up resistors

y = 1 (N/A)

Programmed antifuse links

Programmed antifuses

Logic 1

Pull-up resistors

y = !a & b

CSE467
(a) Before programming

Growing an anti-fuse

Amorphous silicon column

Metal

Oxide

Metal

Substrate

Polysilicon via

(b) After programming

A Transistor-based mask-programmed ROM cell

Row (word) line

Transistor

Logic 0

Column (data) line

Mask-programmed connection

Logic 1

Pull-up resistor
A Transistor-and-fusible-link-based PROM cell

Fusible link

Row (word) line

Transistor

Logic 0

Column (data) line

Logic 1

Pull-up resistor

Standard MOS versus EPROM transistors

(a) Standard MOS transistor

(b) EPROM transistor
An EPROM transistor-based memory cell

- Row (word) line
- EPROM Transistor
- Logic 0
- Logic 1
- Pull-up resistor
- Column (data) line

Intel 1702A EPROM die, circa 1971- 256x8
An SRAM-based programmable cell
Summary of Programming Technologies

<table>
<thead>
<tr>
<th>Technology</th>
<th>Symbol</th>
<th>Predominantly associated with ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fusible-link</td>
<td>![Symbol]</td>
<td>SPLDs</td>
</tr>
<tr>
<td>Antifuse</td>
<td>![Symbol]</td>
<td>FPGAs</td>
</tr>
<tr>
<td>EPROM</td>
<td>![Symbol]</td>
<td>SPLDs and CPLDs</td>
</tr>
<tr>
<td>E²PROM/FLASH</td>
<td>![Symbol]</td>
<td>SPLDs and CPLDs (some FPGAs)</td>
</tr>
<tr>
<td>SRAM</td>
<td>![Symbol]</td>
<td>FPGAs (some CPLDs)</td>
</tr>
</tbody>
</table>

What you will do

- Implement an imaging processor in an Altera FPGA
  - Altera DE-1 development board
  - Verilog and Altera Quartus toolset
- You won’t learn end-to-end design
  - Starting with problem statement
  - Ending with product
- You will experience design
  - You will implement an algorithm in hardware
  - You will create your own design implementation
  - You will analyze existing code and modify it