Welcome to CSE467!

- Course Staff: Bruce Hemingway and Charles Giefer
- Course web: http://www.cs.washington.edu/467/
- 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- audio string model

Highlights:

- We'll be reading hand-outs and papers from various sources.
- The course work will be built around an embedded-core processor in an FPGA.
- Tools are Active-HDL from Aldec, Synplify, and Xilinx ISE.
- Languages are verilog and C.
- Applications in the FPGA will include some audio.

You may do this week's lab at your own time.

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)
  - Multilevel logic
  - High-speed design
- HDLs and synthesis
  - Verilog
  - Synthesizing to an FPGA
  - Re-usable code modules- IP
- Lab equipment
  - Oscilloscopes
  - Logic analyzers
  - An audio design project

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

You may do this week’s lab at your own time.
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
  - $K_1 - K_{16}$ 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 µP

- How?
  - Use COTS development system
  - Or a custom PCB with µP, 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
FPGA primer

Idealized FPGA

CLB Architecture

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

A Simple Programmable Function

Potential links

Logic 1

y = 1 (N/A)
Unprogrammed Fusible Links

Programmed Fusible Links

Unprogrammed antifuse links

Programmed antifuse links

Growing an anti-fuse

A Transistor-based mask-programmed ROM 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><del>v</del></td>
<td>SPLDs</td>
</tr>
<tr>
<td>Antifuse</td>
<td><del>D</del></td>
<td>FPGAs</td>
</tr>
<tr>
<td>EPROM</td>
<td><del>L</del></td>
<td>SPLDs and CPLDs</td>
</tr>
<tr>
<td>EPROM/FLASH</td>
<td><del>L</del></td>
<td>SPLDs and CPLDs (some FPGAs)</td>
</tr>
<tr>
<td>SRAM</td>
<td><del>SRAM</del></td>
<td>FPGAs (some CPLDs)</td>
</tr>
</tbody>
</table>

What you will do

- Implement physical model of a vibrating string in a Xilinx FPGA
- Xilinx development board
- PC design environment—Active-HDL, Synpilly
- Verilog and Xilinx ISP toolset
- Microblaze soft processor—example of IP
- 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