

CS/EE 3710

Computer Design Lab
Fall 2008

CS/EE 3710

- ◆ Computer Design Lab
 - T Th 3:40pm-5:00pm
Lectures in WEB 110, Labs in MEB 3133 (DSL)
- ◆ Instructor: Erik Brunvand
 - MEB 3142
 - Office Hours: After class, when my door is open, or by appointment.
- ◆ TA: Eli Ribble
 - Office hours to be determined

CS/EE 3710

- ◆ Web Page - all sorts of information!
- ◆ <http://www.eng.utah.edu/~cs3710>
- ◆ Contact:
 - 3710@list.eng.utah.edu
 - Goes to everyone in the class
 - teach-3710@list.eng.utah.edu
 - Goes to instructor and Tas
- ◆ No textbook – I'll hand out stuff.
 - There's lots of good stuff linked to the web page

Prerequisites

- ◆ Digital Logic
 - CS/EE 3700 or equivalent
- ◆ Computer Architecture
 - CS/EE 3810 or equivalent
- ◆ First assignment is a review of these subjects!
 - It's on the web page now!
 - It's due on Thursday, September 4 at 5:00pm

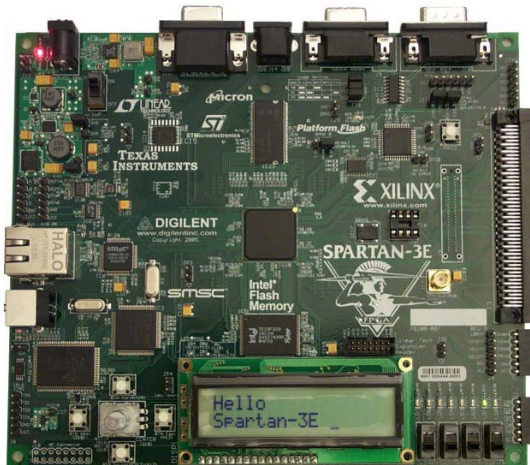
Class Goal

- ◆ Use skills from both 3700 and 3810 to build a moderately sized project
 - Specifically, a computer processor!
 - Based on a commercial RISC core
- ◆ Team projects – groups of ~3-4
 - Each group will have to customize the processor for a particular application
 - You choose the application!
 - You choose the customizations!

University of Utah

CS/EE 3710

Hardware Infrastructure



FPGA: Spartan-3E FPGA
500,000 gate equivalents,
plus 40Kbytes of onboard RAM
Clock: 50 MHz crystal clock oscillator
Memory: 128 Mbit Parallel Flash
16 Mbit SPI Flash
64 MByte DDR SDRAM
Connectors and Interfaces:
Ethernet 10/100 Phy
JTAG USB download
Two 9-pin RS-232 serial port
VGA output connector
PS/2- style mouse/keyboard port,
rotary encoder with push button
Four slide switches
Eight individual LED outputs
Four momentary-contact push buttons
100-Pin expansion connection ports
Three 6-pin expansion connectors
Display: 16 character - 2 Line LCD

- ◆ Spartan-3E “starter” Board from Xilinx

CS/EE 3710

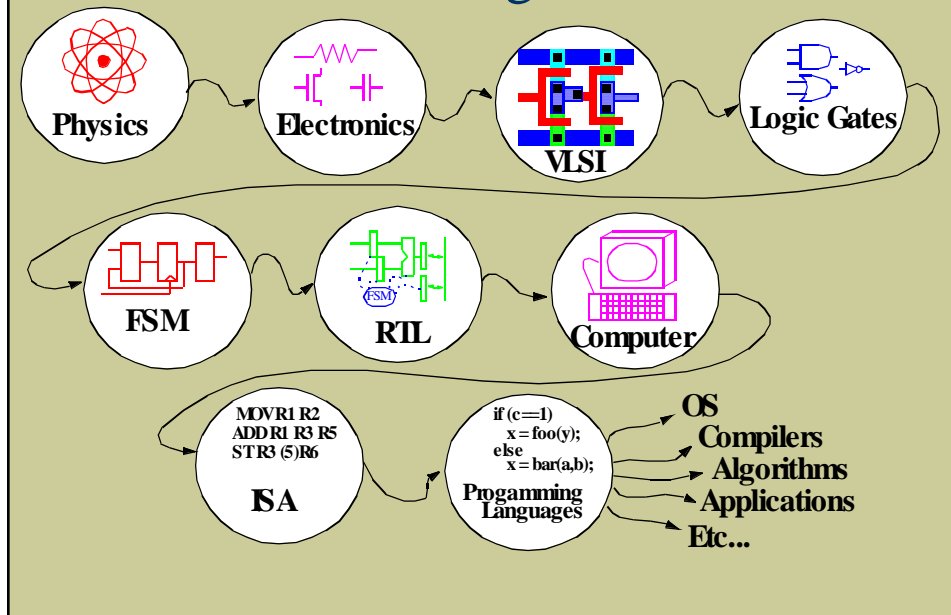
CAD Software

- ◆ **Xilinx ISE WebPACK 10.1**
 - Verilog system definition
 - Schematic capture
 - Verilog/Schematic simulation
 - Synthesis to the Spartan-3E
 - Mapping to the Spartan-3E

University of Utah

CS/EE 3710

The Big Picture



The Big Picture

- ◆ I'll hand out a **Baseline ISA** (it's on the web site)
 - Every group must implement these instructions
- ◆ There will be labs that require you to design and demonstrate steps along the way
- ◆ Each group will customize their processor
 - New instructions
 - New I/O
 - Other features
- ◆ End up demonstrating code running on your processor!

The Big Picture

- ◆ Design the **datapath** using schematics/Verilog
 - ALU, register file, shifter, misc. registers, etc.
- ◆ Design the **control FSM** using Verilog
 - Remember Verilog state machine design from 3700?
- ◆ Use ISE for simulation/synthesis
- ◆ Processor runs on the Spartan-3E board
 - Memory-mapped I/O
 - Keyboard? UART? VGA? Other?

The Short-Term Picture

- Start with a review assignment
- Next assignment is a Finite State Machine mapped to the Spartan-3E board
 - Tbird tail lights...
- Next assignment will be a very small processor
 - I'll hand out mips.v code from Weste/Harris
 - I'll hand out Verlog code for block RAMs
 - I'll hand out sample Fibonacci assembly code
 - You'll augment the processor with ADDI
 - You'll augment the processor with very simple I/O
 - You'll augment the Fibonacci code

The Medium Term Picture

- ◆ We'll hand out lab kits on Tuesday next week during class
 - We'll meet in the DSL, MEB 3133
- ◆ Be thinking about who to team up with
 - Teams will be 3-4 people
 - Good teams have a mix of skills
- ◆ Mid-term presentations
 - Present your plans and your design so far
 - All team members must participate

The Long Term Picture

- ◆ Once teams are formed (Late September)
 - Start working on your project
 - Start with baseline, augment for your application
 - Think about memory and I/O
 - Think about support software (assemblers, compilers, etc.)
 - Think about application software
- ◆ Whole thing due at the end of class
 - Demo day at the end of the semester

Design

- ◆ What is design?
 - Design is the progression from the abstract to the concrete
 - From the idea for the *SuperGizmoWidget* until you've actually got the real live chip in your grubby little paws
 - How does one go from an idea to a product?
 - How does one go from a specification to a piece of hardware?

Exploit Abstraction

- ◆ Design from the top down!
- ◆ Start with an understanding of the complete system
 - The Big Picture!
- ◆ Break it into more manageable chunks
- ◆ Describe the chunks in more detail
- ◆ Continue until the chunks are easy enough that you can build them!

Actually...

- ◆ You can't really do things totally top-down or totally bottom-up
 - Top-down is usually the best place to start though
 - At some point you'll need to look at the details
- ◆ Learning when to switch views is important!
 - When do you switch between levels of abstraction?
 - Learn by doing and with practice

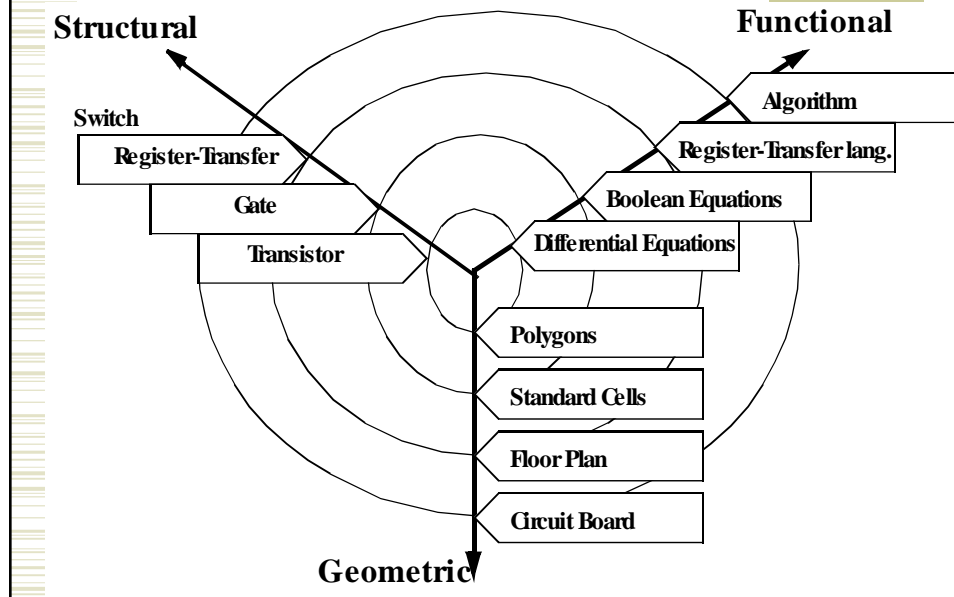
A Couple of Rules

- ◆ Don't build complex systems, build simple ones!
 - Use appropriate abstractions
 - Use hierarchy in your designs
- ◆ Beg, Borrow, and Steal
 - Find tools that will help you
 - Reuse modules when it makes sense
 - Avoid NIH syndrome!

Digital Design Abstractions

- ◆ System Architecture
- ◆ Instruction Set Architecture (ISA)
- ◆ Register-Transfer Level
- ◆ Gates
 - Boolean logic, FPGAs, gate-arrays, etc...
- ◆ Circuits – transistors
- ◆ Silicon – mask data, VLSI

Another Look at Abstraction



When to Switch Levels?

- ◆ When do you switch to a new level in the abstraction hierarchy?
 - When does a collection of transistors look like a gate?
 - When does a collection of gates look like a register-transfer level module?
- ◆ Engineering judgement! One mark of a good engineer is one who breaks things up at the appropriate level of abstraction!

Problems With Abstraction

- ◆ You may abstract away something important!
 - When you jump up a level you lose some info
 - When you jump down a level you may get swamped in the details
- ◆ Example: An appropriate collection of transistors doesn't always behave like a logic gate!
 - Slowly changing signals
 - Metastability
 - Other electrical effects
- ◆ You may also miss some possible optimizations

Design Validation

- ◆ It's hard to make sure that different models are describing the same thing!
- ◆ Write a behavioral model in C, then create a gate-level model in ISE. How do you know they're the same?
 - Simulation?
 - Correct-by-construction techniques?
 - Formal proofs?
 - Cross your fingers?

CAD Tools

- ◆ Mask Level
 - Magic, Mentor, Cadence, Spice, Spectre, etc.
- ◆ Gate Level
 - ISE, Mentor, Cadence, cosmos, irsim, espresso, MisII, etc.
- ◆ RT and up – “High-level” descriptions start to look a lot like software...
 - Verilog, VHDL, HardwareC,
 - ISE-XST, Synopsys, Ambit, Leonardo

High Level Synthesis

- ◆ Allows behavioral descriptions
- ◆ Larger and more complex systems can be designed
- ◆ Abstracts away low-level details
- ◆ Design cycle is shortened
- ◆ Correct by construction
(if you believe in the tools!)

Synthesis Drawbacks

- ◆ Larger circuits
- ◆ Slower circuits
- ◆ No innovative circuits
 - Of course, you can make counter-arguments to each of these drawbacks...

Synthesis Tools

- ◆ A whole bunch of different CAD tools
 - Quite complex
- ◆ ISE-XST from Xilinx
 - Targets Xilinx FPGAs in particular
- ◆ Synopsys is an industry leader in general synthesis
 - Can also target Xilinx FPGAs
 - Try it on your own if you like...