ECE/CS 3710

Computer Design Lab

Ken Stevens

Fall 2009

ECE/CS 3710

- Computer Design Lab
  - Tue & Thu 3:40pm – 5:00pm
    Lectures in WEB 110, Labs in MEB 3133 (DSL)
- Instructor: Ken Stevens
  - MEB 4506
  - Office Hours: After class or by appointment.
- TA: Tyler Day
  - Office Hours to be determined

ECE/CS 3710

- Web Page - all sorts of information!
- http://www.eng.utah.edu/∼kstevens/3710/3710.html
- Contacts:
  - 3710@list.eng.utah.edu
    - Goes to **everyone** in class
    - Must be member of list to post
  - teach-3710@list.eng.utah.edu
    - Goes to instructor and TA
- No textbook – You'll receive handouts
  - There's *lots* of good stuff linked to the web page

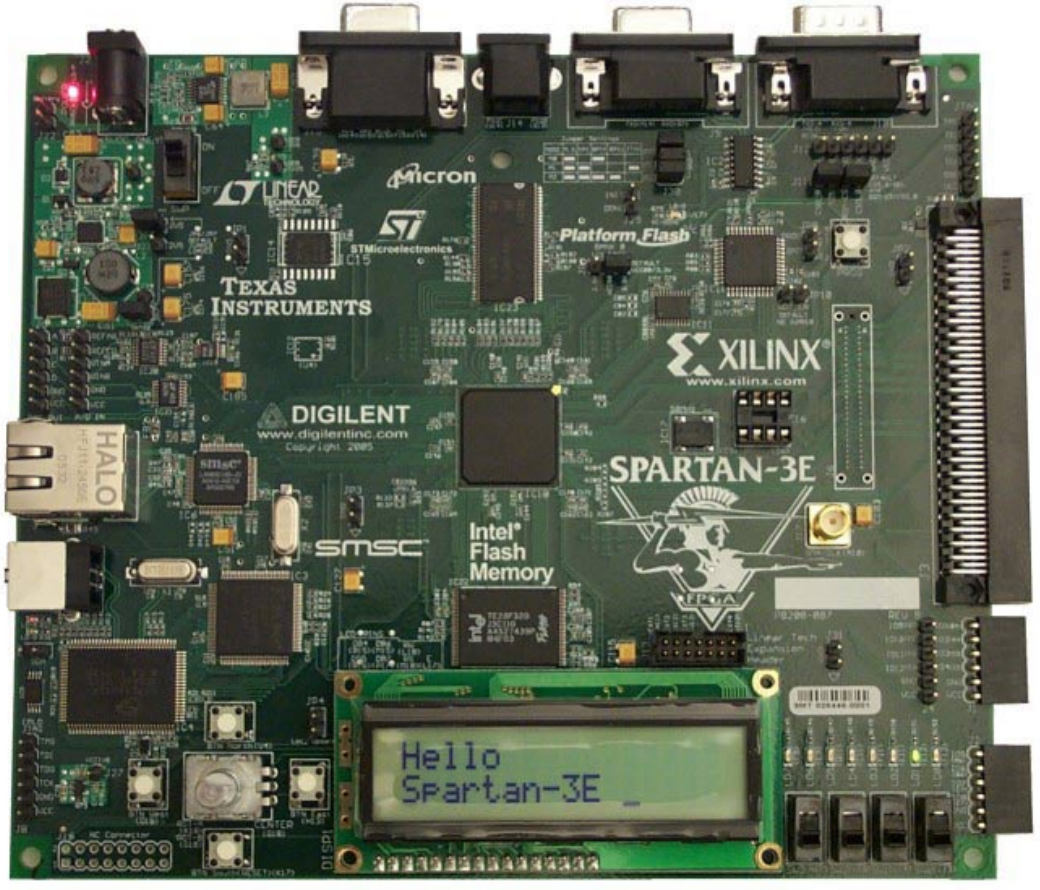
Prerequisites

- Digital Logic
  - ECE/CS 3700 or equivalent
- Computer Architecture
  - ECE/CS 3810 or equivalent
- First assignment is a review of these subjects!
  - It's on the web page now!
  - It's due next Thursday, September 3rd

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 or 4
  - Each group will have to customize the processor for a particular application
    - You choose the application!
    - You choose the customizations!
    - Be creative and have **fun!**

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

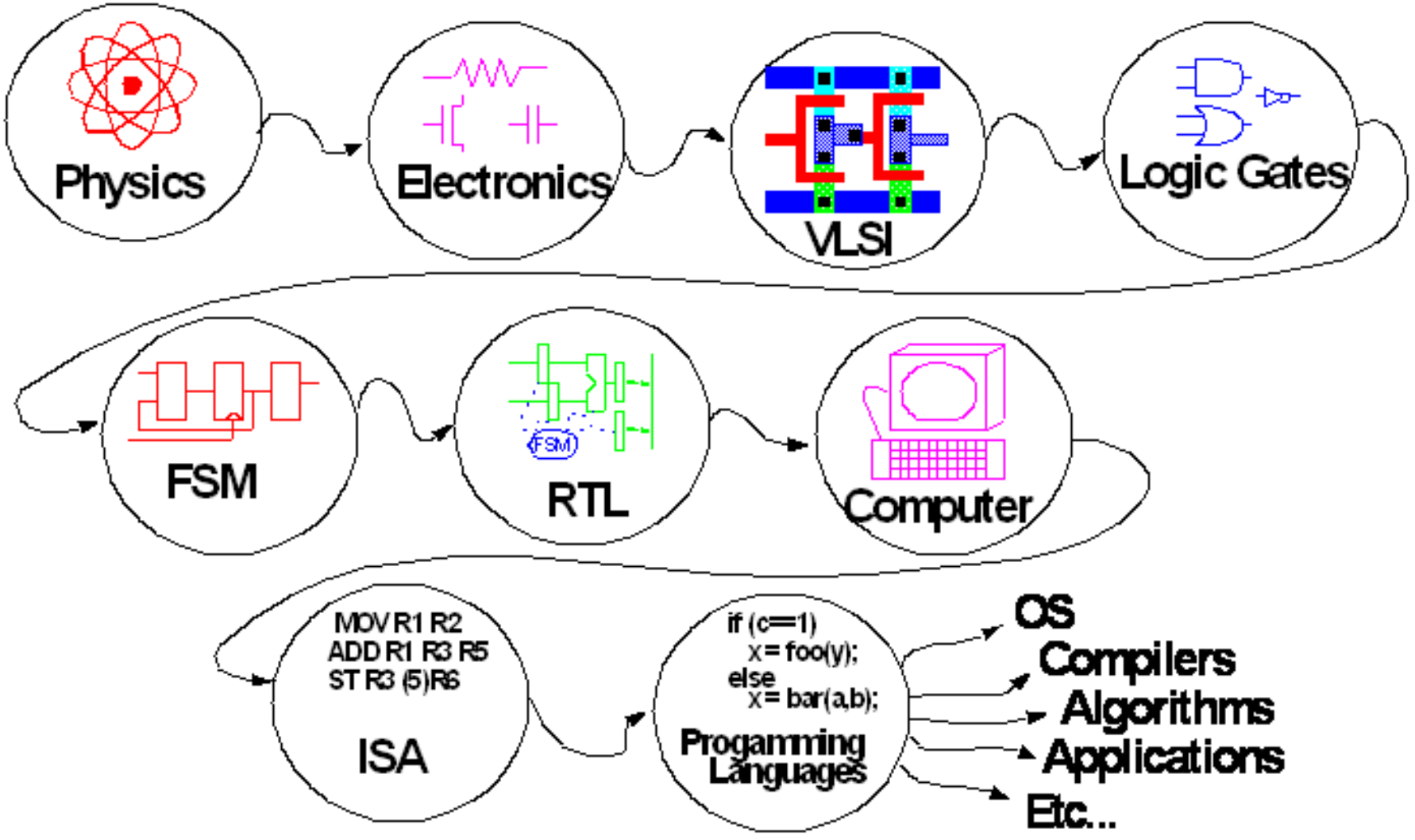
Display: 16 character - 2 Line LCD

- Spartan-3E “starter” Board from Xilinx

CAD Software

- Xilinx ISE WebPACK 10.1 / 11.1
  - Verilog system definition
  - Schematic Capture
  - Verilog/Schematic simulation
  - Synthesis to the Spartan-3E
  - Mapping to the Spartan-3E

The Big Picture



The Big Picture

- You'll get 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 and synthesis
- The Processor you build runs on the Spartan-3E board
  - Memory-mapped I/O
  - Keyboard? UART? VGA? Other?

The Short-Term Picture

- Start with a review assignment
- Nest assignment is a Finite State Machine (FSM) mapped to the Spartan-3E board
  - Cool thunderbird tail lights. . .
- Next assignment will be a very small processor
  - I'll provide the mips.v code from Weste/Harris
  - I'll provide the Verilog code for block RAMs
  - I'll provide sample Fibonacci assembly code
  - You'll augment the processor with the ADDI instruction
  - 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 complementary skills
- Start dreaming about your project
  - Mid-term project proposal presentations
    - Present your plans and your design so far
    - All team members must participate and present

The Long Term Picture

(2010 is an eternity away, eh?)

- 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
- The complete working project due at the end of class
  - Demo day at the end of the semester
    - Same time as CE Senior Project demo day
      - Let's work the seniors!

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 hands
  - 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
    although
  - 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 with practice

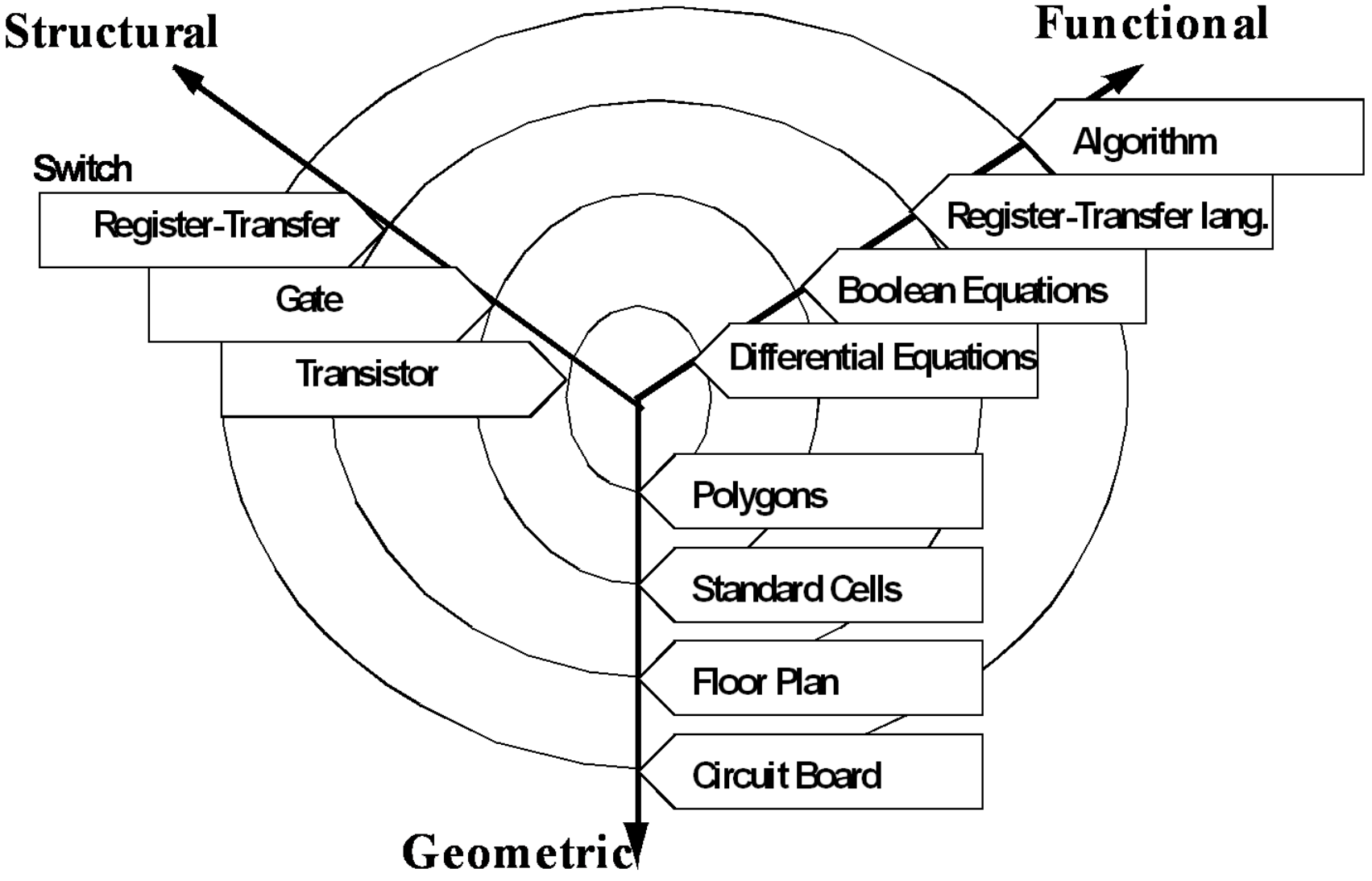
A Couple of Rules

- Don't build complex systems, build compositions of simple ones!
  - Use appropriate abstractions
  - Use hierarchy in your designs
- Don't re-invent the wheel
  - Exploit available resources
  - Find tools that will help you
  - Reuse modules when it makes sense
  - Avoid NIH syndrome (This isn't CalTech. . .)

Digital Design Abstractions

- System Architecture
- Instruction Set Architecture (ISA)
- Register-Transfer Level (RTL)
- 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 judgment! 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!
  - You lose some information when you jump up a level of abstraction
  - When you move down a level you may get swamped in the details and need to modify the higher level
- Example: An appropriate collection of transistors doesn't always behave like a logic gate!
  - Slope (rise or fall time) too great
  - 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 Xilinx ISE
- How do you know they are 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, System-C. . .
  - 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 trust the tools)

Synthesis Drawbacks

- Larger circuits
- Slower circuits
- No innovative circuits
  - You can, of course, make some 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 industry leader in general synthesis
  - Can also target Xilinx FPGAs
  - Try it on your own if you like...
- You will get to know the Xilinx CAD tools well