CS/EE 3710

Computer Design Lab

Fall 2019

Introduction

CS/EE 3710

- Computer Design Lab
  - TTh 9:10pm-10:30pm
    Lectures in WEB L102, Labs in MEB 3133 (DSL)
- Instructor: Chris Myers
  - MEB 4112
  - Office Hours: T 12:30-1:30pm or by appointment.
- TA: Harikrishna
  - Initial office hours:
    M 9am – 11:30am, T 8am – 10:30am,
    W 3pm – 5pm, Th 9am – noon, F 11am – 1pm.

Class Meetings

- August 20th – 29th: meet TTh 9:10 to 10:30am in WEB L102 for introductory lectures.
- September 3rd – 19th: meet TTh 9:10 to 10:30am in MEB 3133 for introductory labs.
- September 24th onwards: meet at assigned group meeting time in MEB 3133 for project meetings.
- November 26th: meet in WEB L102 for group presentations.
- December 6th: DEMO DAY

CS/EE 3710

- Canvas Web Page - all sorts of information!
- Contact:
  - teach-3710@googlegroups.com
    - Goes to instructor and TA
- No textbook
  - There's lots of good stuff linked to the web page

Prerequsites

- 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 CANVAS now!
  - Due electronically on CANVAS by midnight on Friday, August 30th

Academic Misconduct

- Note that the SoC and ECE department has adopted a tougher stance on academic misconduct you will need to read the policy, print, and sign the form. At this point in your program, I assume that everybody already has this form on file with your advisor. If you don't, for some reason, you need to it in to Will Turner (the CE academic advisor) by Friday, August 24.

Team Setup

- Send email by Tuesday August 27th to teach-ece3710@googlegroups.com
  - Your name
  - Your major
  - Your preferred teammates (optional)
  - Times you cannot meet on Tuesday and Thursday. There will be a REQUIRED weekly meeting with the TA for each group in addition to the weekly meeting with the instructor during the project.
- Subsets of the teams will work together on the labs to get to know each other.

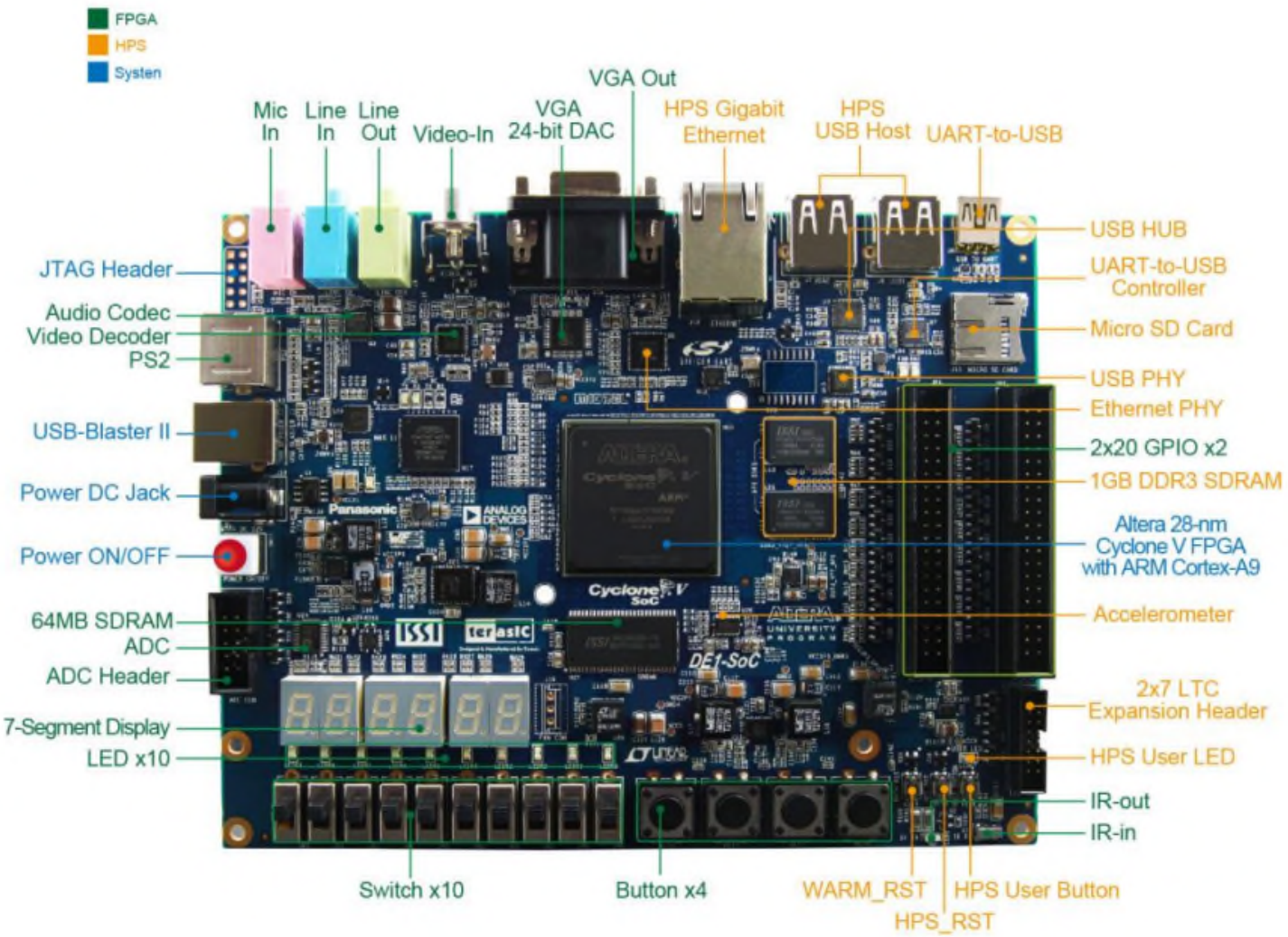
Class Goal

- 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 customize their processor for a particular application
    - You choose the application!
    - You choose the customizations!

Course Deliverables

- Grading breakdown:
  - Labs and checkpoints: 35%
  - Mid-semester presentation: 10%
  - Final Project: 50%
    - Written Documentation: 20%
    - Project details: 30% (project meetings, project functionality, implementation, team member evaluation)
  - Group Evaluations: 5%
- Incomplete grades are only possible if you have a documented medical or legal emergency

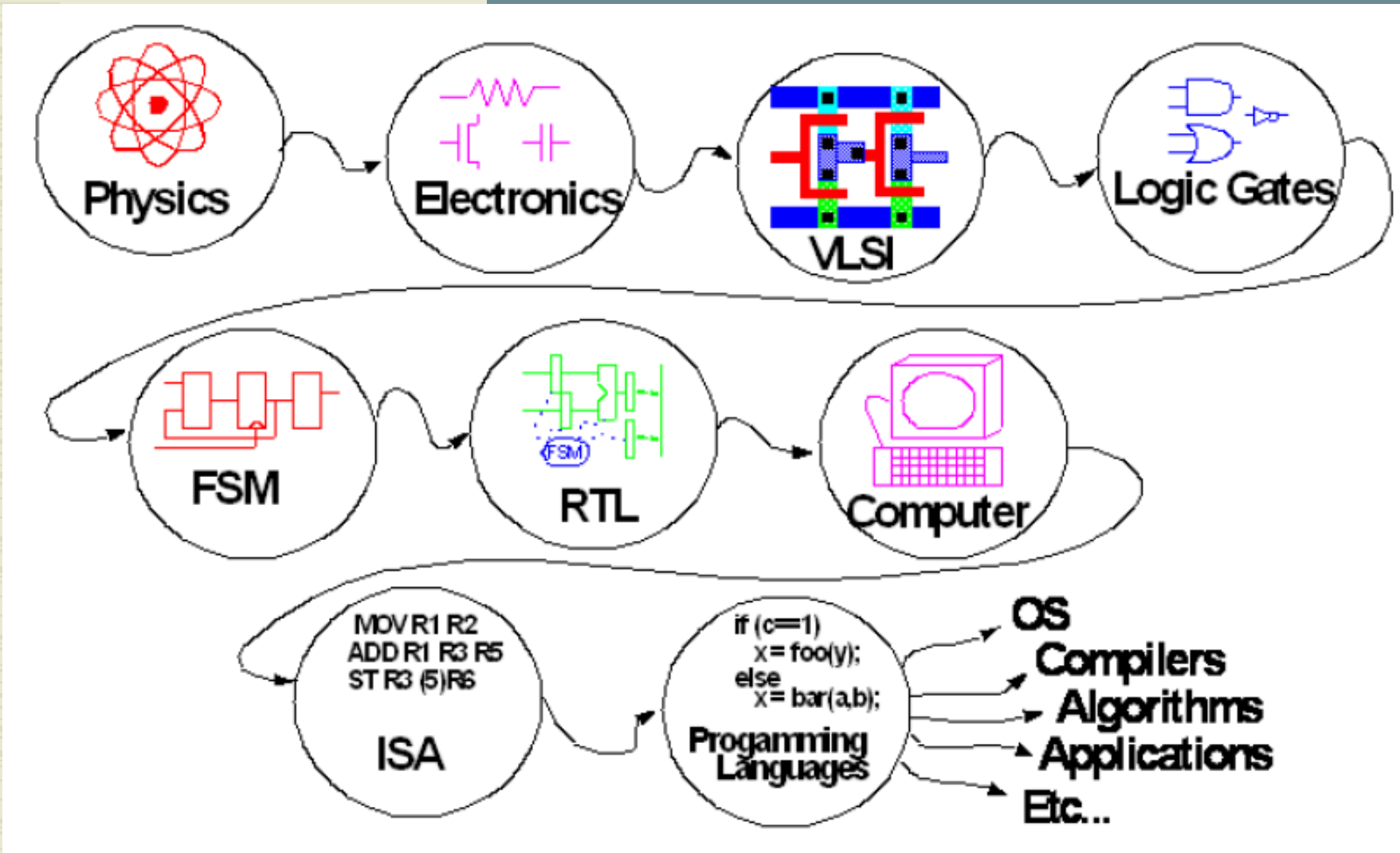
Hardware (terasIC)



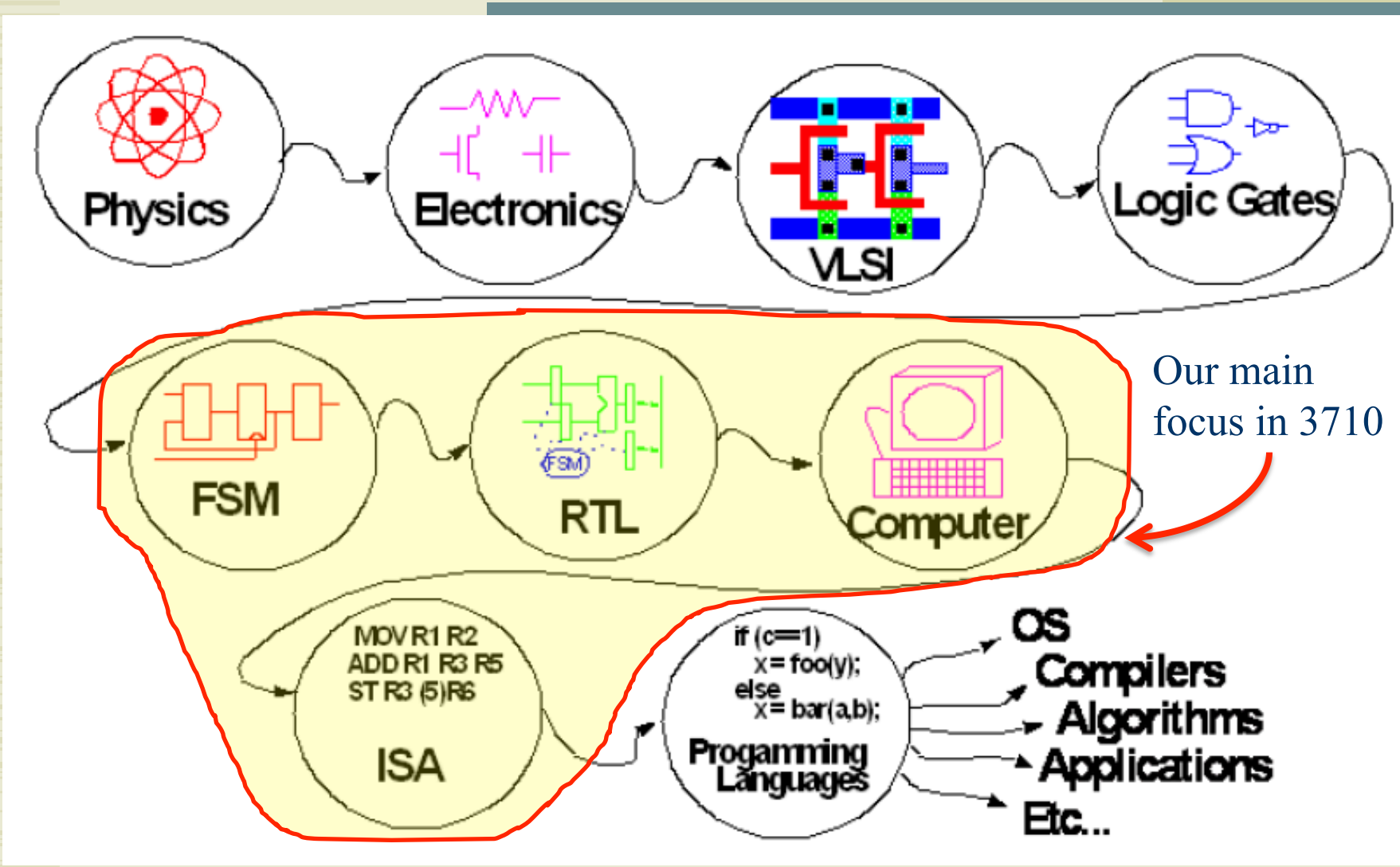
CAD Software

- Quartus Prime
  - Verilog system definition
  - Schematic capture
  - Verilog/Schematic simulation
  - Synthesis to the Altera FPGA
  - Mapping to the Altera FPGA
- This is installed on the DSL machines, in the CADE PC lab, and is free to install on your own machine
  - It's a BIG download though…

The Big Picture



The Big Picture



The Big Picture

- I'll hand out a Baseline ISA on CANVAS
  - Every group must implement these instructions
- There will be checkpoints 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 with a mix of schematics and Verilog
  - Design the datapath
    - ALU, register file, shifter, misc. registers, etc.
  - Design the control FSM
    - Remember Verilog state machine design from 3700?
  - Design the I/O system
    - Memory mapped I/O
    - VGA, PS/2, UART, LCD, etc.
- Use Quartus Prime for simulation/synthesis
- Processor runs on the terasIC FPGA board

Verilog

- Plan on good Verilog coding style this semester!
  - Verilog is NOT a programming language!
  - Verilog is a Hardware Description Language
    - A huge number of Verilog errors are related to confusion between combinational and sequential descriptions
    - Think of the HW first, before coding
  - What is "good" Verilog?
    - I like excessive comments in the code
    - I like clear distinctions between seq. and comb. code
    - I like hierarchy
    - I like using a coding style that makes synthesis easy
    - I like using a purely synchronous clocking style in this class

Remember This?

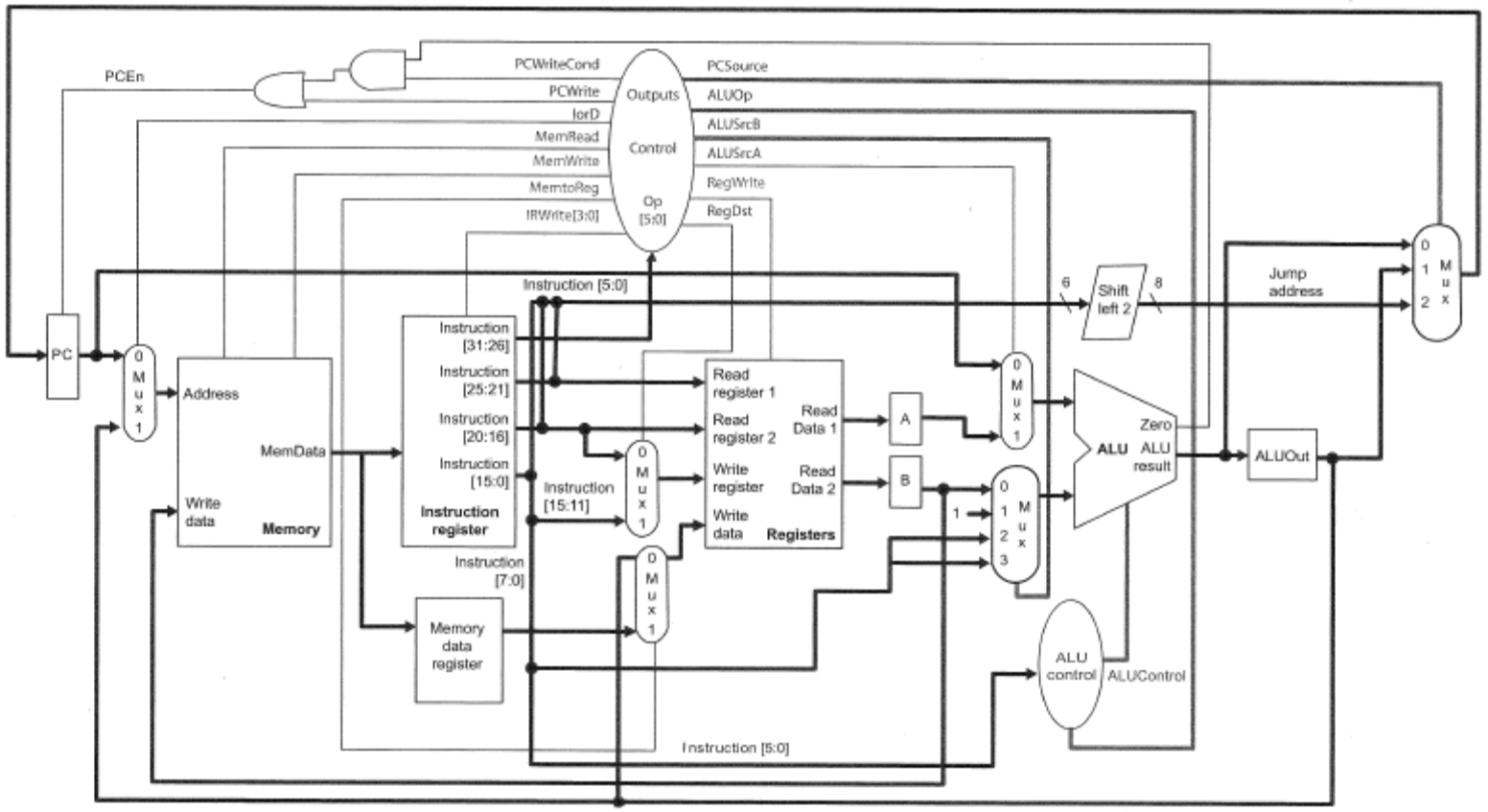
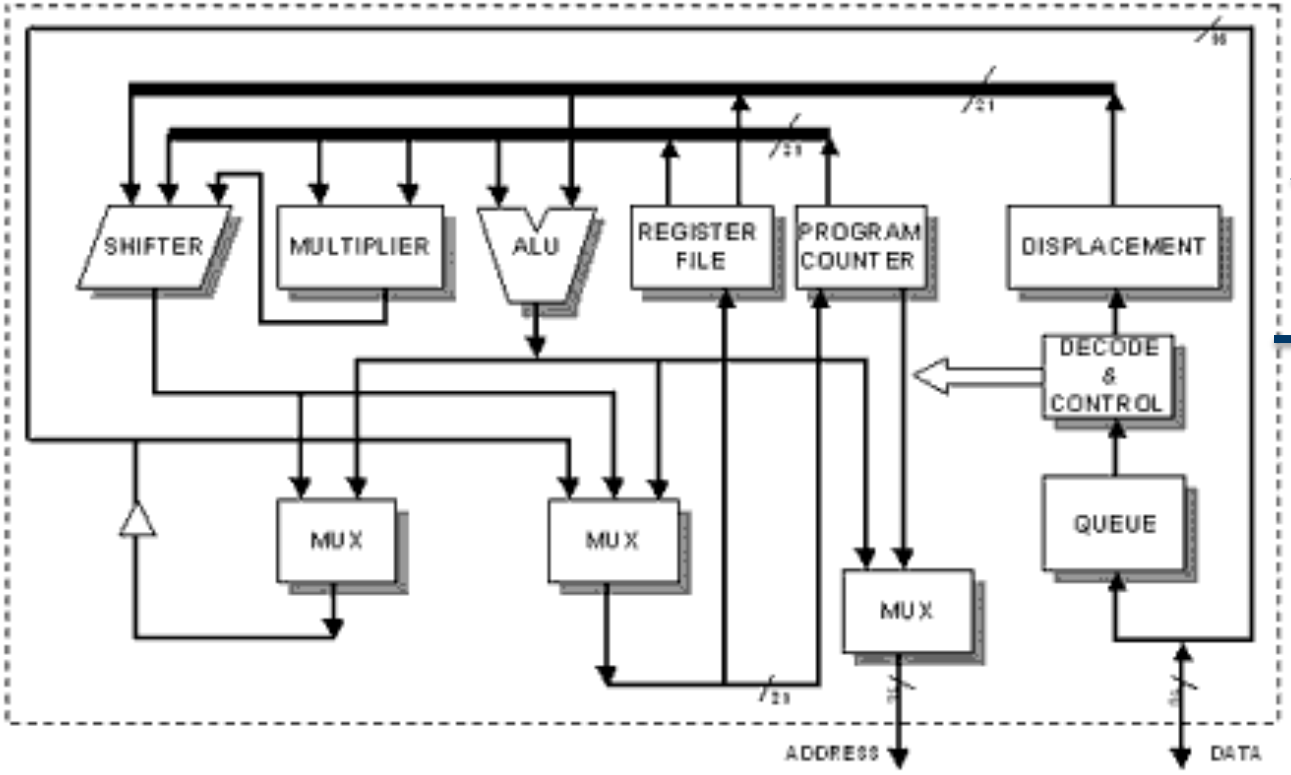


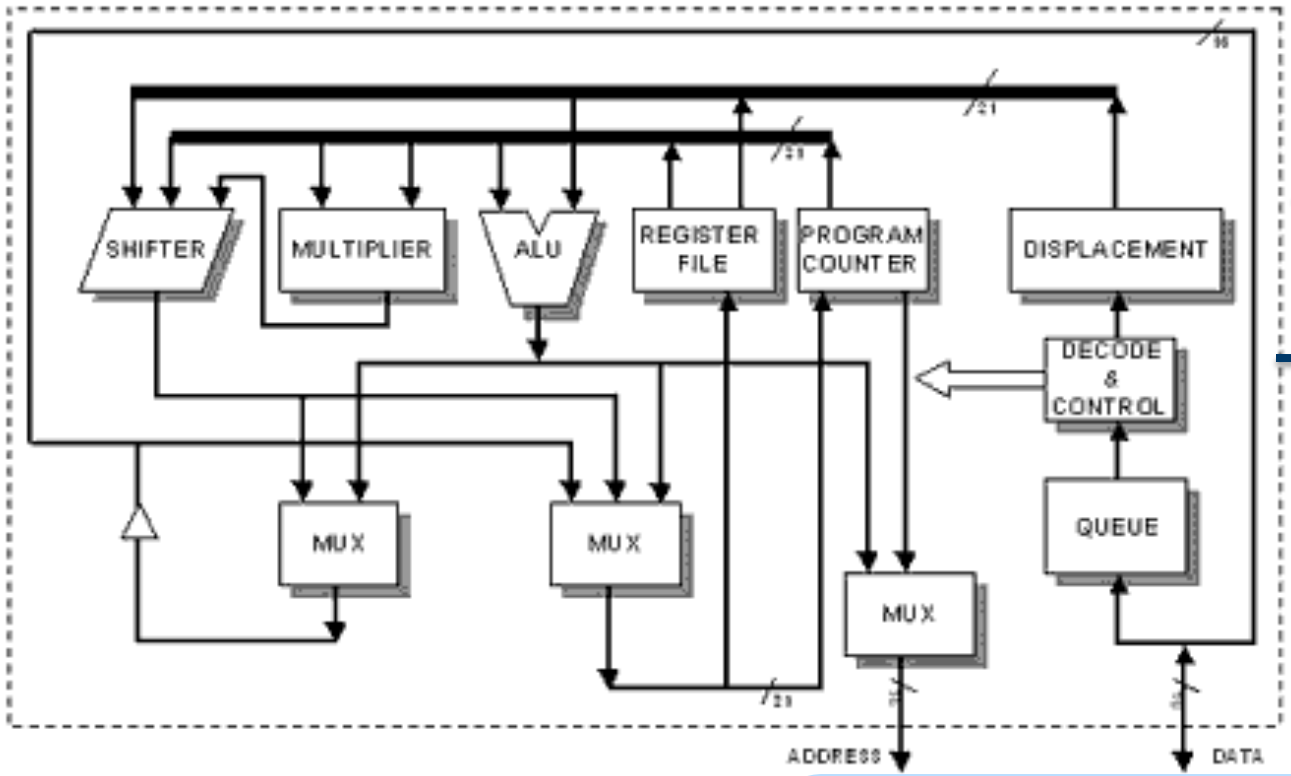
FIG 1.53 Multicycle MIPS microarchitecture. Reprinted from [Patterson04] with permission from Elsevier.

Generic Architecture



Control FSM

Generic Architecture



Control FSM

memory

I/O

·Keyboard
·Mouse
·UART
·Switches
·Etc.

·VGA
·UART
·LCD
·Etc.

The Short-Term Picture

- Start with a review assignment
- Next assignment is a Finite State Machine (FSM) mapped to the terasIC board
  - Thunderbird 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
- Then a VGA assignment
  - Everyone builds a VGA interface
  - VGA version of the Thunderbird…

The Medium-Term Picture

- Lab kits available from DSL service window 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 thinking about your project
  - Project presentations
    - Present your design so far
    - All team members must participate and present

The Long-Term Picture

- Once teams are formed
  - 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
  - December 6th

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 hardware 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 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

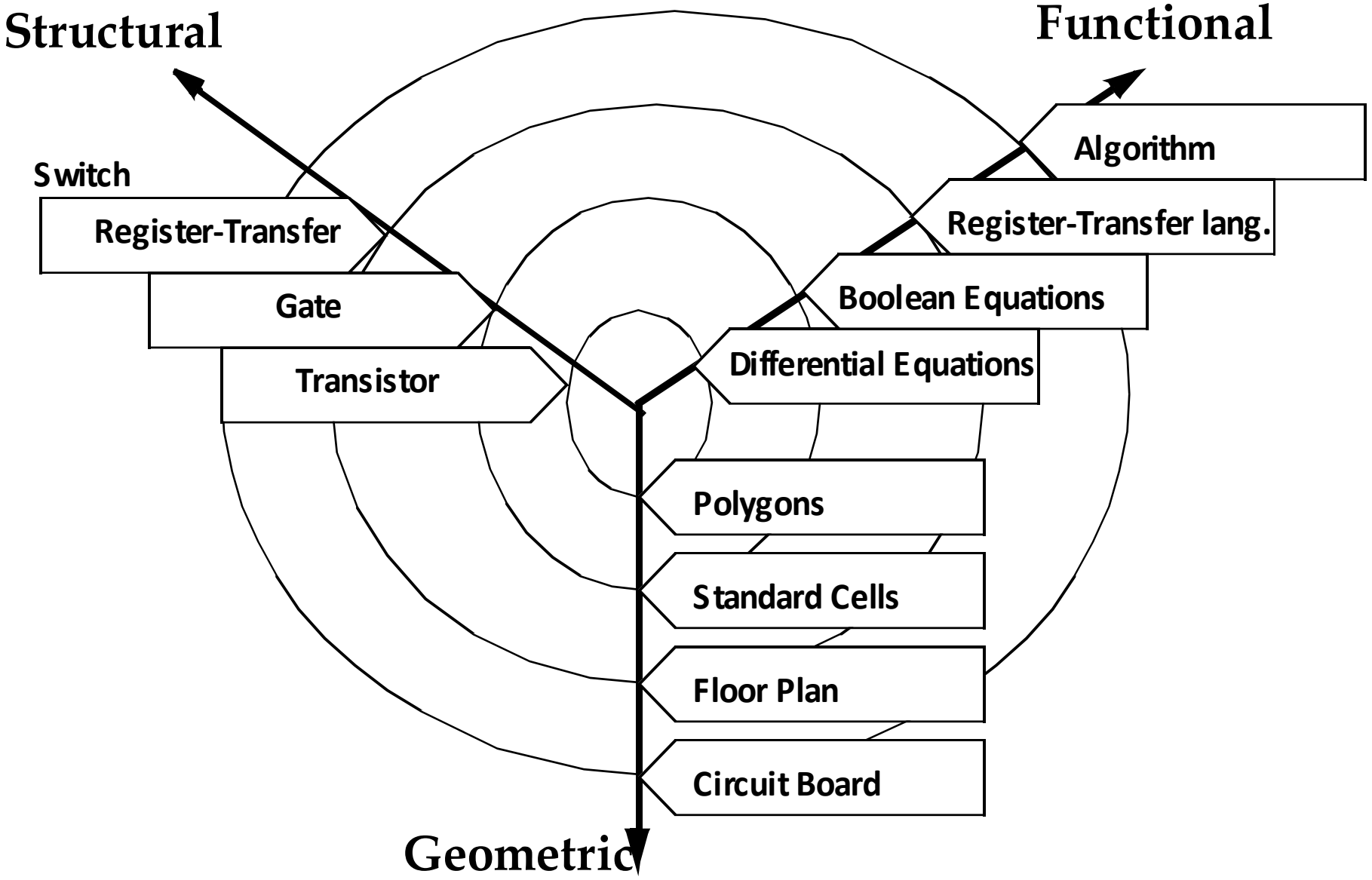
A Couple of Rules

- Don't build complex systems, build compositions of simple ones!
  - Use appropriate abstractions
  - Use hierarchy in your designs
- Don't reinvent the wheel
  - Exploit available resources
  - Find tools that will help you
  - Reuse modules when it makes sense
  - Avoid *not-invented-here* (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 (slope, rise time, fall time)
  - 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 Quartus. How do you know they're the same?
  - Simulation?
  - Correct-by-construction techniques?
  - Formal proofs?
  - Cross your fingers?

CAD Tools

- Mask Level
  - Mentor, Cadence, Spice, Spectre, etc.
- Gate Level
  - Quartus, ISE, Mentor, Cadence, etc.
- RT and up – "High-level" descriptions start to look a lot like software…
  - Verilog, VHDL, System Verilog, SystemC, etc.
  - Quartus, 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
  - Of course, you can make counter-arguments to each of these drawbacks…

Synthesis Tools

- A whole bunch of different CAD tools
  - Quite complex
- Quartus from Altera
  - Targets Altera FPGAs in particular
  - You'll get to know the Altera CAD tools well!

Wrap Up

- 3710 is a project-based course
- Main Goals:
  - Learn about processor design from the ground up by building one
  - Learn about practical details of processor design
    - I/O, memory interfaces, assembly programming, etc.
  - Learn about processor support systems
    - assemblers, linkers, memory loaders, etc.
  - Get experience with a larger design
    - Teamwork
    - Verilog, schematics, CAD tools, hierarchical design, etc.
  - Have Fun!!!