EECS150 Lab Lecture 2
Design Verification

Vincent Lee
Electrical Engineering and Computer Sciences
University of California, Berkeley

Lab Logistics

- T minus three weeks-ish until the project starts
  - We're getting closer to Dan Garcia's hand of God
- For this lab, you will not be synthesizing your design = not using FPGA
- Grades for Lab 0 have been uploaded to bspace so check to make sure you have a score
- Labs are due by the last lab section of the week after they are assigned

Agenda

- Testing Methodologies
- What we forgot to cover last week – Xilinx Report
- Tools and Design Flow: Where are we?
- Software Simulation and Navigating Modelsim
- This Week's Lab: ALU and verification
- Parsing Register Transfer Level (RTL) Descriptions
- Tips, Tricks, Miscellaneous Things to Know

The Xilinx Report Generator

- We tried to go over this with everyone after check off
- Xilinx will generate a report file every time you try to run synthesis to bit stream generation
- Can open the report by running "`make report`" command from same path as Makefile
  - Contains resource utilization, timing information, and more
  - Also displays list of errors and/or warnings

Xilinx Report

Xilinx - ISE Reports (P.15xf) - [Design Summary]

File Edit View Window Help

- Design Overview
  - Summary
  - IOB Properties
  - Module Level Utilization
  - Timing Constraints
  - Pinout Report
  - Clock Report
- Errors and Warnings
  - Parser Messages
  - Synthesis Messages
  - Translation Messages
  - Map Messages
  - Place and Route Messages
  - Timing Messages
  - Bitgen Messages
  - All Implementation Messages
- Detailed Reports
  - Synthesis Report
  - Translation Report
  - Map Report
  - Place and Route Report
  - Post-PAR Static Timing Report
  - Power Report
  - Bitgen Report
- Secondary Reports

Design Properties
- Enable Message Filtering

Optional Design Summary Contents
- Show Clock Report
- Show Failing Constraints
- Show Warnings
- Show Errors

FPGA_TOP_ML505 Project Status (09/06/2012 - 16:00:51)			
Configuration File:	FPGA_TOP_ML505.xreport	Parser Errors:	
Module Name:	FPGA_TOP_ML505	Implementation State:	Programming File Generated
Target Device:	5vlx110tff1136-1	• Errors:	No Errors
Product Version:	ISE 14.1	• Warnings:	21 Warnings (21 new)
Design Goal:	data unavailable	• Routing Results:	All Signals Completely Routed
Design Strategy:	data unavailable	• Timing Constraints:	All Constraints Met
Environment:	System Settings	• Final Timing Score:	0

Device Utilization Summary				
Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	85	69,120	1%	
Number used as Flip Flops	82			
Number used as Latches	3			
Number of Slice LUTs	171	69,120	1%	
Number used as logic	166	69,120	1%	
Number using O6 output only	164			
Number using O5 output only	2			
Number used as Memory	2	17,920	1%	
Number used as Shift Register	2			
Number using O6 output only	2			
Number used as exclusive route-thru	3			
Number of route-thrus	3			
Number using O5 and O6	3			
Number of occupied Slices	60	17,280	1%	
Number of LUT Flip Flop pairs used	184			

Design Summary

Errors, Warnings, Infos…

Program		All Implementation Messages - Errors, Warnings, and Infos	New
xst	⚠	Xst:737 - Found 3-bit latch for signal <ns>. Latches may be generated from incomplete case or if statements. We do not recommend the use of latches in FPGA/CPLD designs, as they may lead to timing problems.	New
xst	⚠	Xst:646 - Signal <Half<0>> is assigned but never used. This unconnected signal will be trimmed during the optimization process.	New
xst	⚠	Xst:647 - Input <GPIO_DIP> is never used. This port will be preserved and left unconnected if it belongs to a top-level block or it belongs to a sub-block and the hierarchy of this sub-block is preserved.	New
xst	⚠	Xst:647 - Input <GPIO_COMPSW<4:1>> is never used. This port will be preserved and left unconnected if it belongs to a top-level block or it belongs to a sub-block and the hierarchy of this sub-block is preserved.	New
xst	⚠	Xst:1780 - Signal <DebugState> is never used or assigned. This unconnected signal will be trimmed during the optimization process.	New
map	⚠	PhysDesignRules:372 - Gated clock. Clock net lab2Lock/ns_or0000 is sourced by a combinatorial pin. This is not good design practice. Use the CE pin to control the loading of data into the flip-flop.	New
par	⚠	Par:288 - The signal GPIO_COMPSW<4>_IBUF has no load. PAR will not attempt to route this signal.	New
par	⚠	Par:288 - The signal GPIO_DIP<0>_IBUF has no load. PAR will not attempt to route this signal.	New
par	⚠	Par:288 - The signal GPIO_DIP<1>_IBUF has no load. PAR will not attempt to route this signal.	New
par	⚠	Par:288 - The signal GPIO_DIP<2>_IBUF has no load. PAR will not attempt to route this signal.	New
par	⚠	Par:288 - The signal GPIO_DIP<3>_IBUF has no load. PAR will not attempt to route this signal.	New
par	⚠	Par:288 - The signal GPIO_DIP<4>_IBUF has no load. PAR will not attempt to route this signal.	New
par	⚠	Par:288 - The signal GPIO_DIP<5>_IBUF has no load. PAR will not attempt to route this signal.	New
par	⚠	Par:288 - The signal GPIO_DIP<6>_IBUF has no load. PAR will not attempt to route this signal.	New
par	⚠	Par:288 - The signal GPIO_DIP<7>_IBUF has no load. PAR will not attempt to route this signal.	New
par	⚠	Par:288 - The signal GPIO_COMPSW<1>_IBUF has no load. PAR will not attempt to route this signal.	New
par	⚠	Par:288 - The signal GPIO_COMPSW<2>_IBUF has no load. PAR will not attempt to route this signal.	New
par	⚠	Par:288 - The signal GPIO_COMPSW<3>_IBUF has no load. PAR will not attempt to route this signal.	New
par	⚠	ParHelpers:361 - There are 12 loadless signals in this design. This design will cause Bitgen to issue DRC warnings.	New
par	⚠	Par:283 - There are 12 loadless signals in this design. This design will cause Bitgen to issue DRC warnings.	New
bitgen	⚠	PhysDesignRules:372 - Gated clock. Clock net lab2Lock/ns_or0000 is sourced by a combinatorial pin. This is not good design practice. Use the CE pin to control the loading of data into the flip-flop.	New
xst	ℹ	Xst:2371 - HDL ADVISOR - Logic functions respectively driving the data and gate enable inputs of this latch share common terms. This situation will potentially lead to setup/hold violations and, as a result, to simulation problems. This situation may come from an incomplete case statement (all selector values are not covered). You should carefully review if it was in your intentions to describe such a latch.	New
xst	ℹ	Xst:1767 - HDL ADVISOR - Resource sharing has identified that some arithmetic operations in this design can share the same physical resources for reduced device utilization. For improved clock frequency you may try to disable resource sharing.	New
xst	ℹ	Xst:2169 - HDL ADVISOR - Some clock signals were not automatically buffered by XST with BUFG/BUFR resources. Please use the buffer_type constraint in order to insert these buffers to the clock signals to help prevent skew problems.	New
map	ℹ	LIT:243 - Logical network GPIO_DIP<7>_IBUF has no load.	New
map	ℹ	LIT:395 - The above info message is repeated 11 more times for the following (max. 5 shown): GPIO_DIP<6>_IBUF, GPIO_DIP<5>_IBUF, GPIO_DIP<4>_IBUF, GPIO_DIP<3>_IBUF, GPIO_DIP<2>_IBUF To see the details of these info messages, please use the -detail switch.	New
map	ℹ	MapLib:562 - No environment variables are currently set.	New

A Note on Errors and Warning

- Errors are bad… Fix them
- Warnings
  - Some generated by Xilinx tools – nothing you can do about it
  - Can be generated by your HDL code – may or may not need to be fixed
- Good practice to always check for warnings coming from you code
  - Don’t always have to fix but…
  - May give hints as to why your synthesized design isn’t working
  - There’s usually a reason why the tools are complaining about your code

Testing Methodologies

- Exhaustive testing
  - Throw every possible input at the circuit (for CL anyways)
  - Ideal and covers everything including edge cases (100% coverage)
  - Scales very poorly
- Random testing
  - Uniform sampling of input space
  - Edge cases randomly sampled (limited coverage)
  - Scales better than exhaustive testing
  - May combine with some target vectors

Testing Methodologies

- Button-up approach
  - Unit test most primitive modules first
  - Work your way up by running integration tests when modules interact with each other
  - For complex systems, very effective verification method
- Top-down approach
  - Start at top level module and work your way down through submodules
  - Stub modules that aren't implemented
  - Good if problems tend to occur towards top

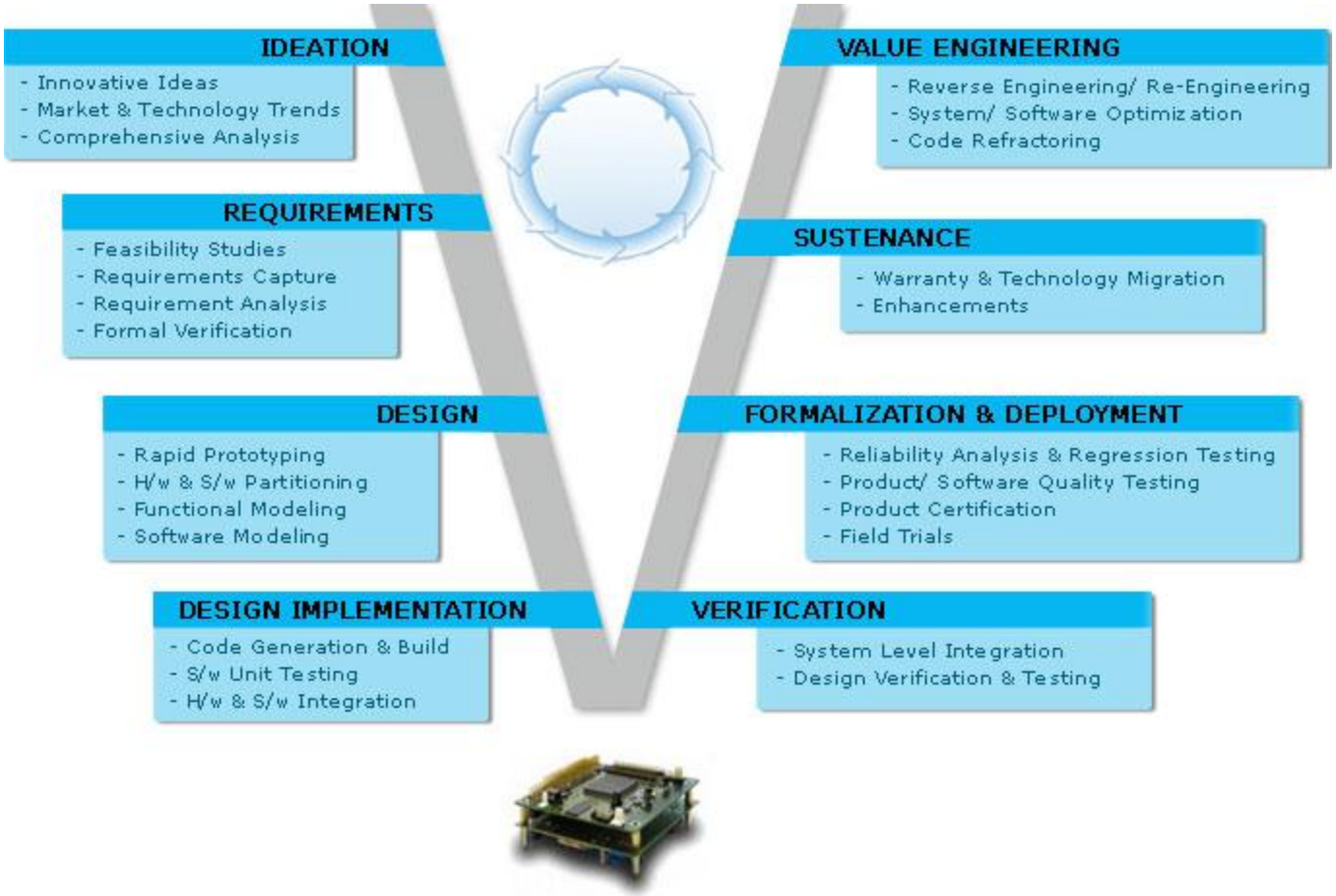
For this lab and class…

- Recommend button-up approach
- Project will have many small modules that compose a very large complex system
- Single bug will cripple your project
- Random testing for ALU and system with many inputs
- Supplement hardcoded test vectors to cover edge cases and frequently used cases

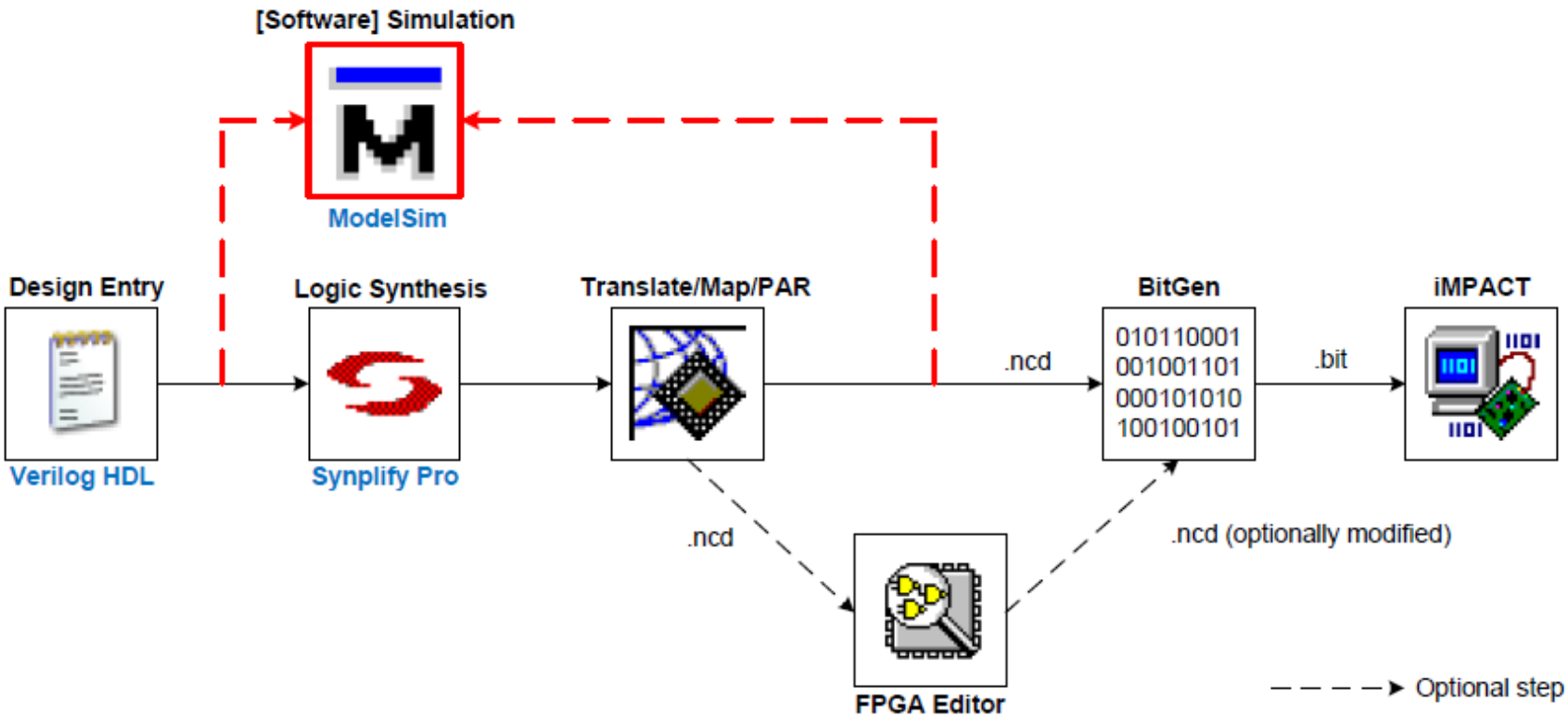
For this lab and class…

- We will only provide basic test cases
- Your job to fill in the gaps – the real world is 50%+ testing and 50%- everything else
- In general, groups who write more/most test cases finish the project faster
- Don't count on us giving you a solution if you fall behind
  - You need to understand your code to do the next checkpoint
- We cannot write testbenches for modules that you designed

Digital Design Development Cycle



Where We Are in the Development Flow…



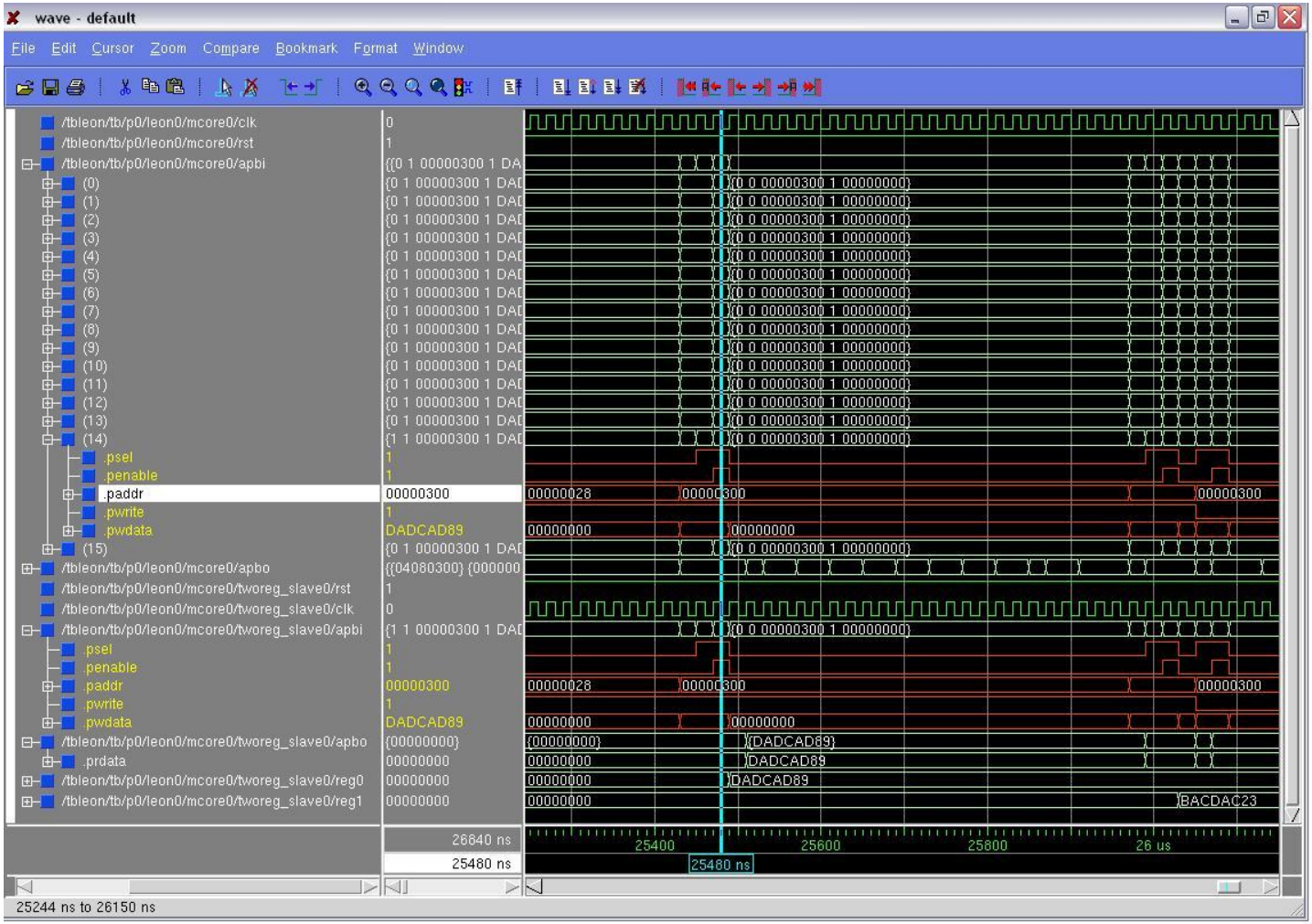
Software Simulation

- Tool that simulates behavior of HDL code before synthesis
- Gives design feedback as waveforms or test messages
- Requires explicit simulation test cases which usually aren't synthesized
- Sometimes may differ from actual synthesis since tools infer logic upon synthesis
- Hopefully irons out most/all bugs before synthesis

Motivation Behind Software Simulation

- Synthesis takes FOREVER… want to be able to do verification quickly in software
- Enhances visibility by allowing access to every signal
- Testing is important
  - A broken product isn't worth anything (also not worth credit)
  - Entire teams in industry devoted to verification and DFT
- Systems are complicated
  - Easy to make mistake in 1000s to millions of lines of code
  - Modules may work separately but not together
- NRE costs are expensive – one bug can cost the entire design

Modelsim Simulation Tool

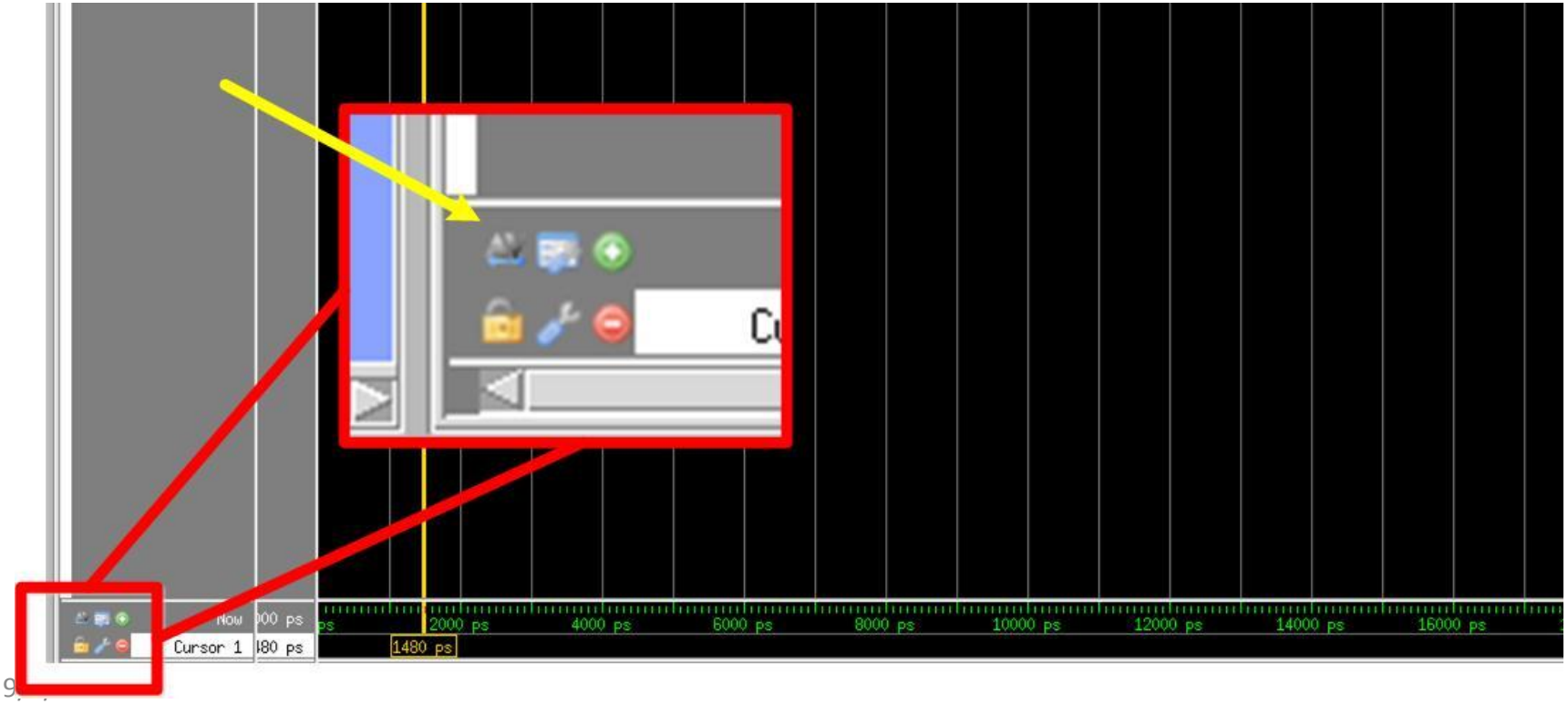


Navigating the Tools

- Modelsim can be invoked by the `vsim` command
- Waveform colors
  - Green waveforms usually good (all values defined)
  - Red waveforms usually bad (undefined values)
  - Blue waveforms usually bad (undriven values)
- Radix (hex, decimal, etc.)
  - Right click signal -> Radix -> Selected Radix

Navigating the Tools

- Use “+” or “-” to zoom in or out
- Toggling leafs on and off - removes excess file path description



This Week's Lab

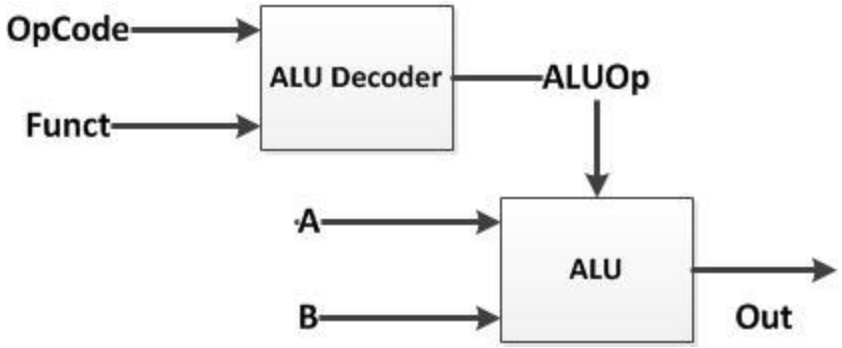
- Objectives:
  - Become familiar with module test harnesses and testbenches
  - Learn how to use Modelsim to simulate in software and debug your modules before synthesis
- Pre-lab requirements:
  - Write verilog for ALU before lab
- Check off requirements:
  - Functional ALU and testbenches
  - Answers to checkoff questions
  - Due Thursday 9/20 @ 8PM

This Week's Lab

- You will be provided the skeletal files for:
  - ALU testbench (ALUTestbench.v)
  - ALU test vector harness (ALUTestVectorTestbench.v)
  - ALU Decoder (ALUdec.v)
  - ALU file (ALU.v)
- You will NOT be synthesizing the design (no makefile on top level)
- Instead you will be verifying your design by using Modelsim (makefile in /sim directory)
- READ THE DOCUMENT AND RTL SPECIFICATION

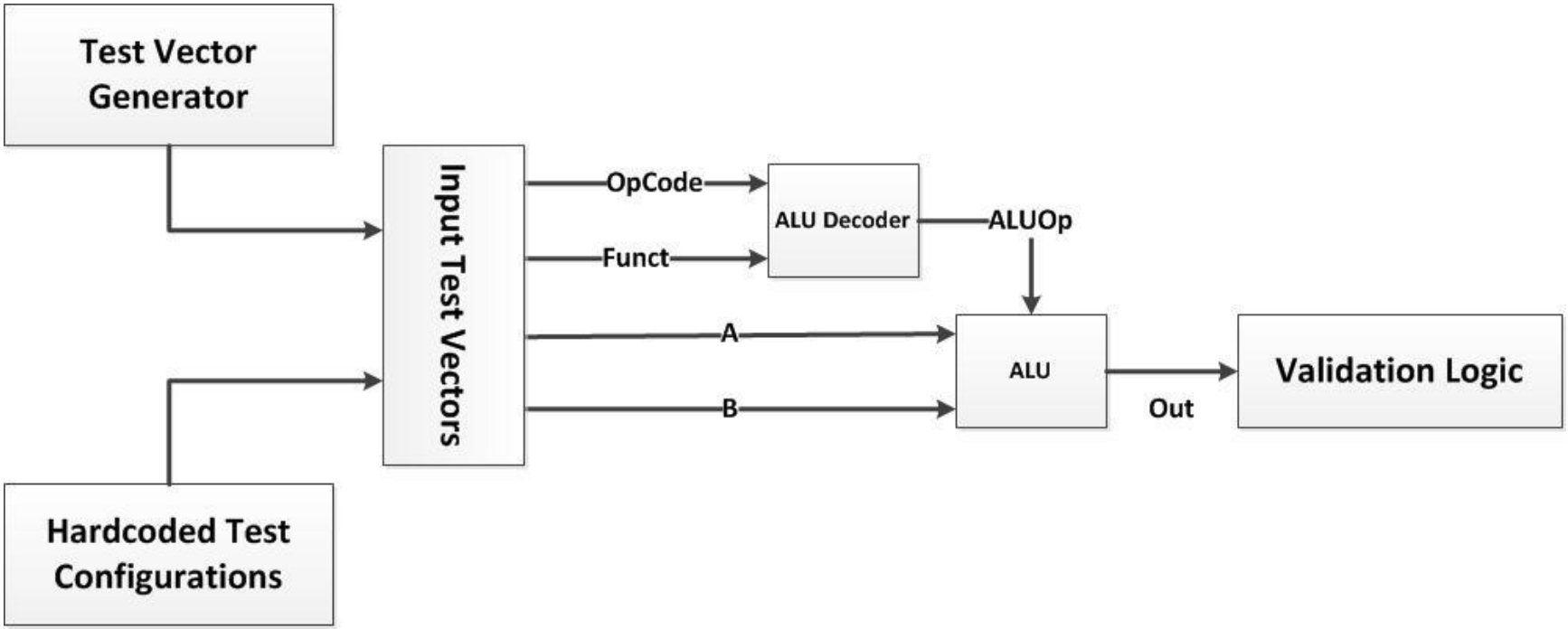
ALU Decoder and ALU

- Controller for the ALU
  - Takes instruction Opcode and Func
  - Deduces appropriate ALUOp
- ALU
  - Takes ALUOp and operands
  - Performs operation specified on operands



ALU Decoder and ALU Simulation Test Harness

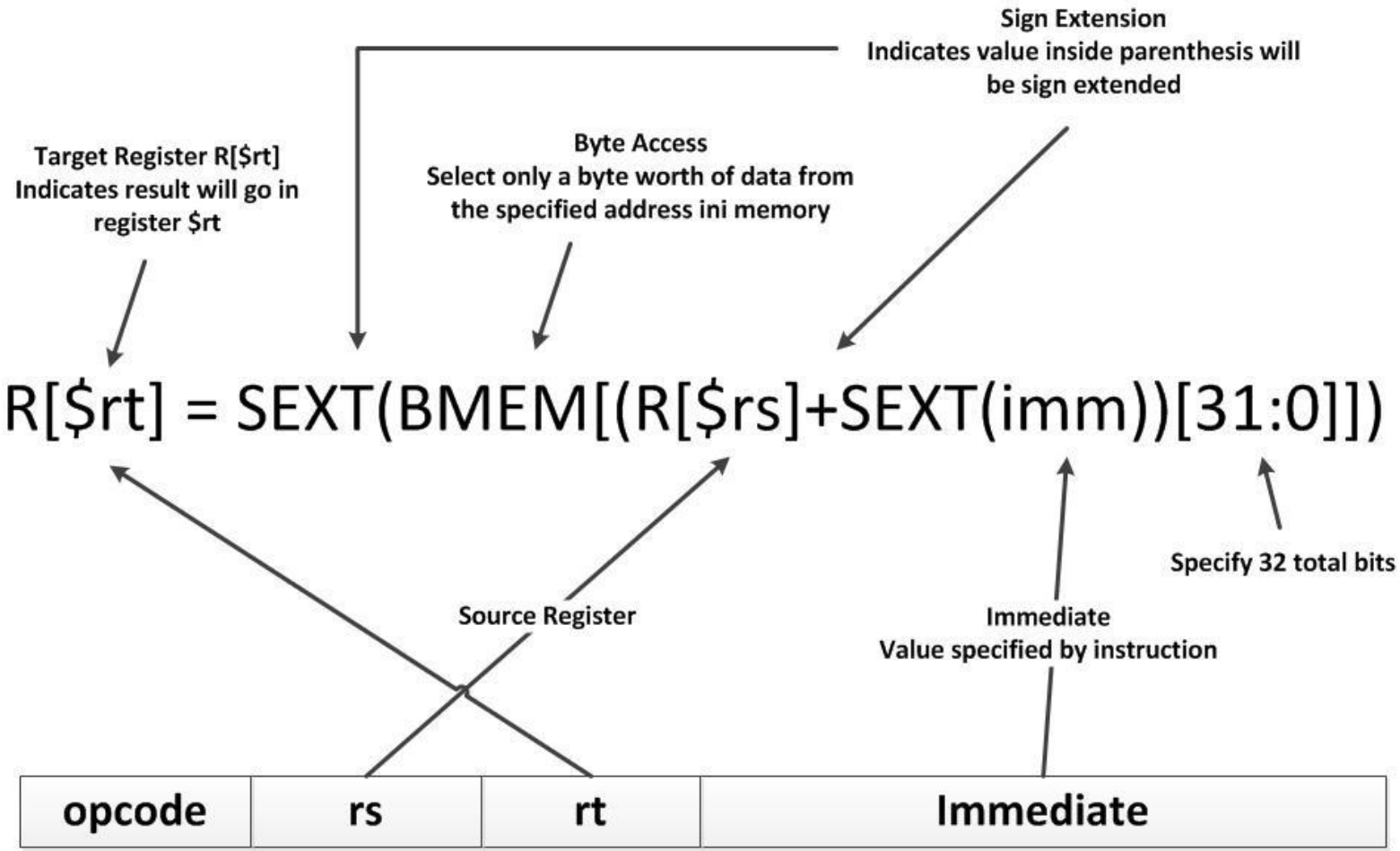
- ALU Decoder and ALU are Devices Under Test (DUTs)
- Equivalent block diagram for simulation:



Register Transfer Level (RTL) Descriptions

- SEXT(*) = sign extend value of *
- ZEXT(*) = zero extend value of *
- PC = memory address of instruction
- >>> = shift right arithmetic
- (A ? B : C) = if A, then evaluate to B, else evaluate to C
- BMEM(*) = byte memory at *
- HMEM(*) = half memory at *
- WMEM(*) = word memory at *

RTL Example: Load Byte (I-Type)



RTL Example: Branch on Less Than or Equal (J-Type)

Program Counter Assignment
Target register is instruction program counter

Conditional Branch
If R[$rs] <= 0, evaluates to SEXT(imm) << 2
Else, evaluates to 0

Two bit shift for byte alignment

$$PC = PC + 4 + (R[\$rs] <= 0 \ ? \ SEXT(imm) << 2 : 0)$$

Account for Next Instruction

Sign Extension

opcode	src	00000	Immediate

Lab Tips and Tricks

- **Read the lab document and understand the specification before you start**
- Watch for signed and unsigned operations – they are different
  - Unsigned comparisons
  - Unsigned loads
- Watch source field ordering – order is important
  - Watch order for SLLV, SRLV, and SRAV
    - R[$rt] >> R[$rs] vs. R[$rs] << R[$rt]
    - This will make your ALU fail the testbench
- Shift right logical (SRL) and shift right arithmetic (SRA) are different
- Watch operation binding strength

Running Simulation

- To run simulation, `cd` to `/sim` directory
- Run `make` in directory
- Simulation output will fill the screen for each testbench you are running
- Simulator looks in `/sim/tests` for `.do` files – each file specifies one test case simulation
- Simulation generates `/results` directory with transcript and waveform file (.wlf)
- Waveform can be viewed with Modelsim or calling ./viewwave

Macro Files

- Macro files help define variables so that you don't have to hardcode:
  - Indicated by .vh file
  - Use macros in verilog by using tick mark `
- Two macro files included in /src directory
  - ALUop.vh – contains ALU operation code macros
  - Opcode.vh – contains MIPS ISA opcode macros

Lab Assignment

- Prelab assignment:
  - Write code for ALU and ALU decoder
- Lab assignment:
  - Complete testbenches and add appropriate test cases
  - Write hardcoded tests and manually add testcases to file
    - Yes we want you to type in the 1s and 0s to the testvector file
  - Simulate and verify that ALU and decoder works correctly
- Checkoff
  - Demonstrate functional ALU and ALU Decoder

Next Week's Lab

- Next week's lab lecture: Overview of Chipscope
- Next week's lab: List Processor and Chipscope
  - Datapath and Controller Design
  - What to do when simulation works but synthesis fails… oh dear…

Questions, comments, or concerns?