CS250 Section 3

Yunsup Lee

Image Courtesy www.ibm.com
Announcements

- Lab 2: Write and Synthesize a Two-Stage SMIPSv2 Processor is out
  - Lab 2 due on September 24th (Thursday) before class
  - Four late days you can use
- Tutorial 3: Build, Run, and Write SMIPS Programs
  - Talks about using the SMIPS assembler / compiler / linker
- Tutorial 4: Simulating Verilog RTL using Synopsys VCS
  - Uses SMIPSv1 as an example
- Tutorial 5: RTL-to-Gates Synthesis using Synopsys Design Compiler
  - Uses SMIPSv1 as an example
- Next week’s (September 18th) office hour is 1pm - 2pm
Infrastructure

- Tool Servers
  - ilinux[1-3].eecs
  - cory353-[(2*n)l].eecs (Notice that ‘l’ is the lower case of ‘L’)
  - NX server installed on all servers except ilinux[2,3].eecs
- License Servers
  - Two license servers (Take a look at ~cs250/tools/cs250.bashrc)
- SVN Server
  - https://isvn.eecs.berkeley.edu/cs250/<login>
  - Notice that you need to commit the directory that current-* points
Infrastructure

- Please take care of your tracks...
  - Logout your NX session after done
  - Make sure that you don’t have any tools running
- I will email you twice a week if
  - You have a NX session running
  - You have a tool running
- It’s going to be an automated script
  - So if you are actually running something just ignore
Lab 1 Feedback

- Clock constraints
  - change “build/dc-syn/constraints.tcl”
  - change “src/testharness.v” always #<clock cycle/2> clk = ~clk;
- Timing constraints met at synthesis but not after place and route
  - Adding more buffers (e.g., clock buffers)
  - Model is more accurate with extracted parasitics for wires
  - Iterate between synthesis and place and route, topographical mode
- Min & Max power are the same
- Circuit too small
Lab 1 Feedback

- Post Synthesis Area
  - dc-syn/current-dc/reports/*.area.rpt
- Post Synthesis Power
  - dc-syn/current-dc/reports/*.power.rpt
- Post Place+Route Area
  - icc-par/current-icc/reports/chip_finish_icc.sum
  - UTILIZATION RATIOS / Core Area
- Total Cell Count / DFFX* Cell Count
  - icc-par/current-icc/reports/chip_finish_icc.sum
  - MASTER INSTANTIATION INFORMATION
Lab 1 Feedback

- Average Power (min, max)
  - pt-pwr/current-pt/reports/vcdplus.power.avg.{min,max}.report
- Peak Power (min, max)
  - pt-pwr/current-pt/reports/vcdplus.power.time.{min,max}.report
Questions

- Any questions about lab 1?
Question 1

- Tell me about the toolflow.
  - Big picture
  - What’s the order you should go through the build directory?
Question 2

- What’s the difference between the following?
  - vcs-sim-behav
  - vcs-sim-rtl
  - vcs-sim-gl-syn
  - vcs-sim-gl-par
- Why do we do all these simulations?
Question 3

- What’s the difference between the following?
  - icc-par/current-iccdp
  - icc-par/current-icc
Question 4

- What else does PrimeTime analyze?
Question 5

- I want to see the layout of standard cells. What should I do?
Lab 2: SMIPSv2 Processor
Lab 2: Things you need to do

- Write Verilog
  - smipsProc.v, smipsProcCtrl.v smipsProcDpath.v
  - add more modules: for example, register file and ALU
- Run SMIPSv2 test assembly programs and benchmarks
- Globally installed stuff
  - ~/cs250/install/{smips-tests, smips-bmarks}
- Local stuff
  - ~/cs250/lab2/v-smipsv2-2stage/{smips-tests, smips-bmarks}
- Complete the multiply benchmark
- Take a look at vcs-sim-rtl/Makefile
Lab 2: Things you need to do

* Stuff that matters in your build directory
  * vcs-sim-rtl, dc-syn, vcs-sim-gl-syn
* Write two assembly tests and two benchmarks
  * We will run your programs on other people’s pipeline
* Optimize your design
  * We will synthesize your processor with 2ns time constraint
* Measure your IPC for benchmarks
  * Bonus question - Build your own branch predictor
* We will post results on the course homepage
Lab 2: dc-syn is tricky
Lab 2: dc-syn is tricky

- Core = Processor + Memory
- Two makefiles exist
  - Makefile.combinational
    - Synthesizes smipsProc
    - Use this makefile for vcs-sim-gl-syn
  - Makefile.dummy
    - Synthesizes smipsCore_synth
    - Use this makefile for critical path analysis
    - Use this makefile for area analysis
Lab 2: dc-syn is tricky
Lab 2: dc-syn is tricky

Functionally Incorrect!
Lab 2: Adding more modules

- build/vcs-sim-rtl/Makefile
  - vclibsrcs
  - vsrcs
- build/dc-syn/Makefile
  - vclibsrcs
  - vsrcs
- not build/vcs-sim-gl-syn/Makefile
  - why?
Lab 2: Adding more test programs

- build/vcs-sim-rtl/Makefile
  - global_asm_tests
  - global_bmarks
  - local_asm_tests
  - local_bmarks
- build/vcs-sim-gl-syn/Makefile
  - global_asm_tests
  - global_bmarks
  - local_asm_tests
  - local_bmarks
Lab 2: SMIPS Compiler

- After you compile, first run it on the functional simulator
  - to make sure it is running as intended
  - to make sure it doesn’t have any instructions that aren’t SMIPSv2
- The Compiler might generate some instructions that aren’t SMIPSv2
  - Keep your memory accesses aligned
  - Make sure you don’t use arithmetic that aren’t defined
    - e.g., multiply, divide
Lab 2: IPC for benchmarks

- build/vcs-sim-rtl
  - make run-bmarks-perf
- Bonus question - Build your branch predictor
  - why would this make your IPC better?
Lab 2: Misc

- LAB2_ROOT=`pwd`
  - Notice you should use ` instead of ’
  - A Latex weirdness - does anyone know how to fix it?