Discussion Session 7

CS/ECE 552

Ramkumar Ravi

12 Mar 2012

Mark your calendar

- IMPORTANT DATES
  - **03/14** – Design Overview due
  - **03/19** – HW4 due; HW5 out
  - **03/23** – Project Demo I (also due in class are workplan and schedule for demo 2. If you have already included this in your project plan, then you need not submit again)
  - **04/02** – SPRING BREAK :)

TODAY

→ ***IMPORTANT: Only King-Lab machines support Wiscalculator***

→ Demo1 tools usage and other requirements

→ HW4 RF bypass discussion

→ Problem 5 discussion

→ other questions/concerns

Project Tools – WISC-SP12 ASSEMBLER

- Information on this tool is available here:
  http://pages.cs.wisc.edu/~david/courses/cs552/s12/includes/assembler.html
- This tool assembles programs to be loaded into memory.
  - Inputs: *.asm file
  - Outputs: an object file and a listing for your reference
- The path /p/course/cs552-david/public/html/S12/handouts/bins must be present in your .bashrc.local or .cshrc.local
- USAGE: **assemble.sh <your_file>.asm**
- The above command will create 6 files, loadfile_all.img, loadfile.lst, loadfile_0,1,2,3.img. The assembler produces a warning in case of any errors in the *.asm file
- For Demo I, you need to use only the loadfile_all.img (place this file in the same location where your memory2c.v file is)

WISC-SP12 SIMULATOR/DEBUGGER

- Information on this tool called **Wiscalculator** is available here:
  http://pages.cs.wisc.edu/~david/courses/cs552/s12/includes/wisc-sim.html
- This tool reads the output of the assembler and executes it
  - Inputs: loadfile_all.img (for DEMO I)
  - Outputs: .trace and .ptrace
- The path /p/course/cs552-david/public/html/S12/handouts/bins must be present in your .bashrc.local or .cshrc.local
- USAGE: **wiscalculator loadfile_all.img**
- The above command will run the program and print a step by step listing of the instructions executed and what registers they write to

EXAMPLE OUTPUTS

```
(24)$ assemble.sh myfile.asm

Created the following files

loadfile_0.img  loadfile_1.img  loadfile_2.img  loadfile_3.img  loadfile_all.img  loadfile.lst



(25)$ wiscalculator loadfile_all.img

WISCalculator v1.0

Author Derek Hower

Type "help" for more information

Loading program...
Executing...
slbi r1, 0
INUM:       0 PC: 0x0000 REG: 1 VALUE: 0x0000
slbi r1, 85
INUM:       1 PC: 0x0002 REG: 1 VALUE: 0x0055
slli r2, r1, 8
INUM:       2 PC: 0x0004 REG: 2 VALUE: 0x5500
bnez r2, 2
INUM:       3 PC: 0x0006
halt
program halted
INUM:       4 PC: 0x000a
Program Finished
```

WISC-SP12 VERILOG COMMAND LINE SIMULATION

- Information on this tool called **wsrun.pl** is available here:
  http://pages.cs.wisc.edu/~david/courses/cs552/s12/handouts/wsrun.html
- This tool does a lot of stuff → Please go through the page to understand
- For Demo 1, lets say you want to run easyTest.asm on your processor. Then you should use

```
wsrun.pl -prog /p/course/cs552-david/public/html/S12/project/tests/public/complex_demo1/easyTest.asm proc_hier_bench *.v
```

- If your design is right and there are no errors, you will see an output that says:
  Verilog Simulation successful
  Final log, saved in summary.log
  CPI numbers and total instruction count will be printed

WISC-SP12 SYNTHESIS SCRIPT

- Information on this tool called **synth.pl** is available here:
  http://pages.cs.wisc.edu/~david/courses/cs552/s12/includes/synthesis.html
- This tool does a lot of stuff → Please go through the page to understand
- Remember that the code that you submit for DEMO I needs to be synthesizable. If the synthesis files are missing or if the cell area of the processor is reported as 0, NO CREDIT WILL BE GIVEN FOR DEMO I
- Also note that TB files are not supposed to be synthesized

GENERAL PROJECT STRUCTURE

- Top Level module: **proc_hier.v** (Your processor with clk-rst module added. Similar to what you did for HW3)
- In **proc_hier.v**, you will instantiate **proc.v** which will contain your processor code
- You will use **proc_hier_bench.v** as the testbench for non-pipelined processor (WARNING: Do not EDIT this file)
- You will use the single cycle memory for Demo I. The source code along with the synthesizable version is available here

  http://pages.cs.wisc.edu/~david/courses/cs552/S12/includes/single-cycle-mem.html

SUBMISSION INSTRCUTIONS

- We will use handin for Demo I submission. All your .v files need to be in one folder called **demo1** which needs to be submitted (illutsrative purposes)
- You will use handin command to submit your folder

  handin cs552-1 demo1 <path_to_your_demo1_folder>
- There will be ONE electronic submission per project group
- Failure to follow naming conventions, top level directory structure could cause automatic grading to FAIL (POINTS WILL BE DEDUCTED)
- **Vcheck** for all *.v files (other than testbenches) is MANDATORY. You would be reusing most of the modules designed in HW 1, 2 and 3.

Running Vcheck on Multiple files

contributed by – Mona Jalal

Copy the following into a file, filename.sh:

```
#! /bin/csh -f

foreach d (*.v)

        java Vcheck $d > $d.vcheck.out

end
```

Close the file and run chmod +x filename.sh.

Then type ./filename.sh.

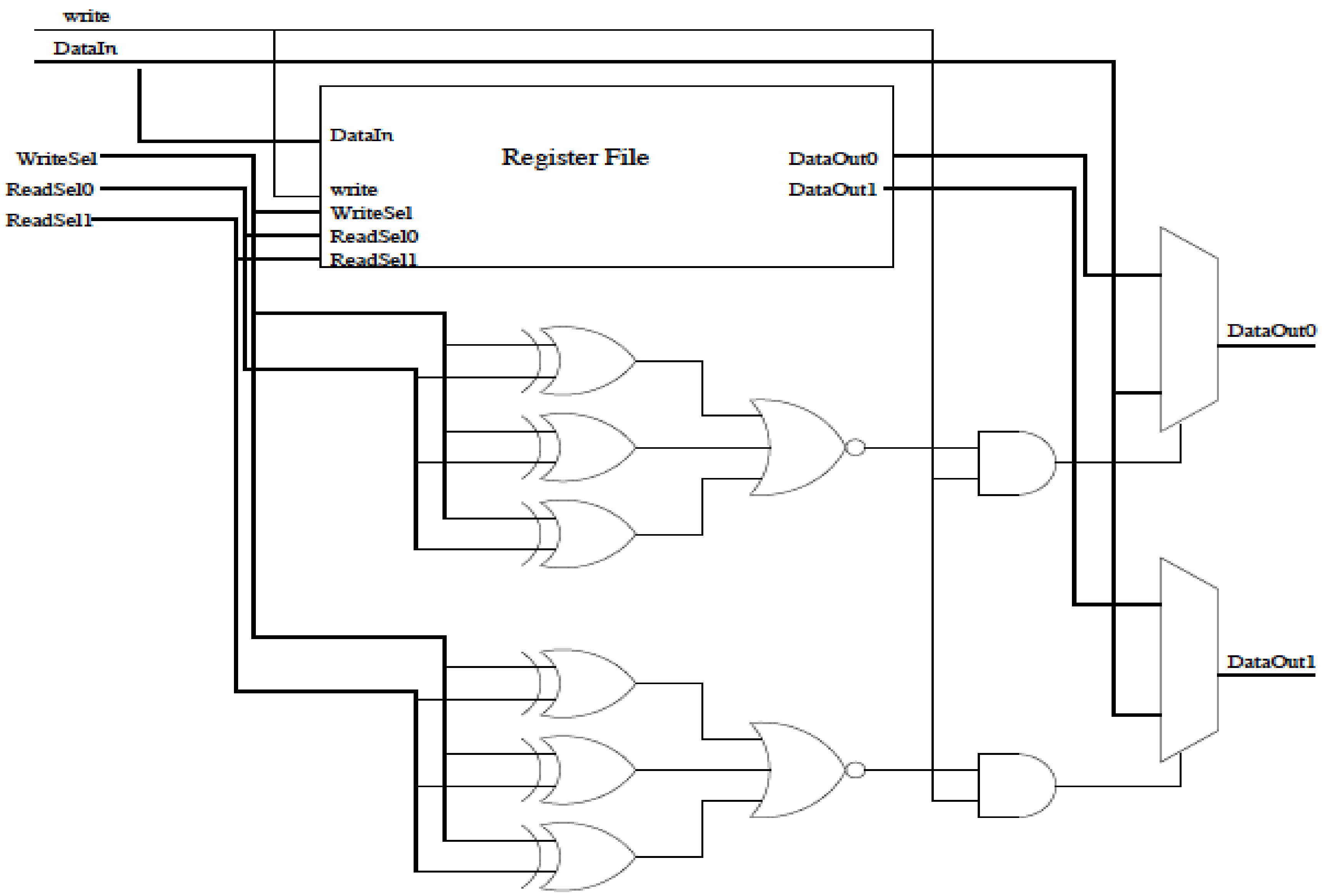
This will Vcheck all the .v files in your current directory, assuming your Vcheck

class files are also located there.

HW 4 problems

(Problem 1 and Problem 5)

RF Bypass – Problem 1 (sample



Problem 5 – Example solution for (a)

→ Dependences

L1:	add $3, $4, $2	R: $4, $2, W: $3	RAW on $3 from L1 to L2, L1 to L3, and L1 to L4
L2:	sub $5, $3, $1	R: $3, $1, W: $5	RAW on $6 from L3 to L4
L3:	lw $6, 200($3)	R: $3, W: $6	
L4:	add $7, $3, $6	R: $3, $6, W: $7	

→ With NO forwarding

L1:	add $3, $4, $2	RAW hazard on $3 from L1 to L2,
	NOP	
	NOP	
L2:	sub $5, $3, $1	
L3:	lw $6, 200($3)	RAW hazard on $6 from L3 to L4
	NOP	
	NOP	
L4:	add $7, $3, $6	