Any questions on lab 2 & lab 3?

- Doing okay with gate-level simulations?
Announcements

• I’m still working to get physical libraries for lab 3 work
  • probably a day or more
  • RTL simulation is working, in the mean time you can get your circuit to work
• RISC-V Specification is taking a little bit more time
  • We are still changing the ISA
• You should get lab 2 done during this week
• Lab 3 is also a lot of work
Lab 2: RISC-V Processor
Lab 3: RISC-V Core
Lab 3: Note on the arbiter
Lab 3: Things you need to do

- Clean up your RISC-V v2 3-stage pipeline with a BTB
- Now change pipeline structure (add stall signals) to deal with the cache memory interface
  - You can’t always put a memory request (when [i|d]memreq_rdy is deasserted)
  - Memory response may take some time (when [i|d]memresp_val is deasserted)
  - Start with the Instruction cache first
- Run RISC-V v2 assembly test programs and benchmarks
  - Globally installed programs
    - ~/cs250/install/{riscv-tests, riscv-bmarks}
  - Local programs in your lab harness
Lab 3: Things you need to do

- Push your design all the way through the flow
- Count instruction mix for all assembly tests and benchmarks
- Measure energy consumption
- Generate analytic energy model
  - A: instruction mix matrix
  - x: energy/instruction
  - b: energy consumption
  - $Ax = b$ holds. Use MATLAB regression to get a best guess to $x$
  - Use your energy model and measure the error
## Area Breakdown

<table>
<thead>
<tr>
<th>Hierarchical cell</th>
<th>Global cell area</th>
<th>Local cell area</th>
<th>Design</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Absolute Total</td>
<td>Combinational</td>
<td>Noncombinational</td>
</tr>
<tr>
<td>riscvProc</td>
<td>7593.8750</td>
<td>5.5296</td>
<td>49.7664</td>
</tr>
<tr>
<td>ctrl</td>
<td>1163.0586</td>
<td>1163.0586</td>
<td>0.0000</td>
</tr>
<tr>
<td>dpAth</td>
<td>517.2266</td>
<td>517.2266</td>
<td>0.0000</td>
</tr>
<tr>
<td>dpAth/alu</td>
<td>19911.9707</td>
<td>19911.9707</td>
<td>19911.9707</td>
</tr>
<tr>
<td>dpAth/clk gate _id_reg_tohist_reg</td>
<td>58.7072</td>
<td>58.7072</td>
<td>58.7072</td>
</tr>
<tr>
<td>dpAth/rfile</td>
<td>43926.4756</td>
<td>43926.4756</td>
<td>43926.4756</td>
</tr>
</tbody>
</table>

### Breakdown of dpAth/alu:
- 19911.9707: Total Area
- 19911.9707: Combinational Area
- 19911.9707: Noncombinational Area
- 19911.9707: Black boxes
- 19911.9707: SNPS CLOCK GATE HGH _riscProcPath Regfile
• If you make your ALU using behavioral Verilog, this is what you get
• It’s always good to start with behavioral Verilog to get things working, but after you get things working, you might optimize your ALU
ALU operations

- Here’s the ALU operations you need to support
  - ADD, SUB, SLT, SLTU, SLTI, SLTIU
  - SL, SR, SRA
  - AND, OR, XOR, NOR
ALU Operations: ADD/SUB

- ADDW xc, xa, xb
  - xc = xa + xb
- SUBW xc, xa, xb
  - xc = xa - xb
- Simple arithmetic for SUBW
  - xc = xa + (~xb + 1)
  - two inputs: xa and ~xb
  - carry input: 1
ALU Operations: SLTU/SLTIU

- Did you notice the operands of SLTU and SLTIU are flipped?
  - SLTU xc, xa, xb: xc = (xa < xb) ? 32’d1 : 32’d0
  - SLTIU xa, xb, imm: xa = (xb < imm) ? 32’d1 : 32’d0
- You can’t use the same adder, since the inputs are in a different place
- The natural alignment is “imm - xb” since we use “xa - xb”.
- We can now know whether imm - xb < 0, or imm - xb >= 0
- imm < xb, imm >= xb. We’re almost there.
- If we can do imm >= xb+1, we know imm > xb
- imm - (xb+1) = imm + (~xb)
- Add with no carry
- Think about the signed case
Suggested Datapath
ALU Operations: SL/SR/SRA

- Use one right signed shifter
- How would you do a shift left with a right shifter?
  - Reverse bits!

- How would you do logical right shift?
  - Control the MSB of your operand
Optimize ALU

- You can make an ALU with one adder, shifter, and a logic unit
- Let’s form groups of three
  - (First letter of your name’s ASCII) % 4