Hazard Unit

Lab 7
4 Apr 2017

Objective

- In this lab you will design a “hazard unit” to solve data hazards by:-
  - Forwarding
  - stalling

Step1: The logic of forwarding/stalling

1. Write down a pseudocode to describe the logic of forwarding.
    a. This will help you to understand the idea behind forwarding/stalling before you write your verilog code.
    b. You can use a draft paper/.docx; It won't be submitted.
    c. If you have tried, and still don't understand how it works, please ask the TA.
    d. You can refer to the slides if you need a refreshment.
    e. Note: It should not take longer than 10 mins in case you've studied the lecture.
2. Write down a pseudocode to describe the logic of Stalling.
    a. Follow the same steps of part 1.

Step 2: The Ports

1. Specify the ports of your design.
   a. Please stick with the following names, so you can easily use the provided *hazard_tb.v*
   b. If you need a clarification of why you should use these ports, please ask the TA.

```
module hazard(
    input [4:0] rsD, rtD, rsE, rtE,
    input  [4:0] writeregE,  writeregM, writeregW,
    input  regwriteE, regwriteM, regwriteW,
    input  memtoregE, memtoregM, branchD,
    output reg [1:0] forwardaE, forwardbE,
    output  stallF, stallD, flushE
);
```

Step 3: The Verilog

- This module is a piece of cake if *Step 1* has done successfully.
- Hints:
  - This is a combinational module. So, Don't think of any *always@(posedge clk)*
  - Use *always@(*)* to model your forwarding/stalling logic. You can also use the continuous assignment statement: *assign LHS(wire/output) = RHS (wire, reg, input)*
  - *forwardaE* will be connected to the selector of the "mux" of the "ALU srcA" in IE-stage. Pay attention to the its value in order to forward the correct data.
  - *forwardbE* is identical to *forwardaE,* except it will be connected to "ALU srcB mux".

Step 4a: the ad-hoc testbench

- You can use *hazard_tb.v* to test your design quickly.
- If you changed the ports' names, you have to modify the instantiation in the TB.
- To check if your design is correctly functioning:
  - Add *forwardaE_dump, forwardbE_dump, stall_dump* signals to the waveform.
  - If *forwardaE_dump = forwardbE_dump = 06(hex),* then your forward logic is **correct.**
  - If *stall_dump = 03F(hex),* then your stall logic is **correct.**

Step 4b: Testing against the “processor”.

- If you have finished your pipeline, you can just add the new “hazard unit” to it.
- You can test the complete processor by loading the “instruction memory” with code that has data dependency. If the code has been executed correctly, then your new added hazard unit is working.