Task 1:
In order to identify where the data hazards were happening, I analyzed the given instruction sequence on the lab PDF and checked to see if there were any registers that were being written to and subsequently used as sources in the following instructions. This way I would be able to verify my waveform after implementing the hazard detection for load-use, 1-ahead, and 2-ahead logic.

Task 2:
For the load-use hazard detection, I created conditional logic to first check if a register is being read in the EX stage, and additionally check if either ID/EX.RegisterRt/Rd is equal to IF/ID.RegisterRs, or if ID/EX.RegisterRt/Rd is equal to IF/ID.RegisterRs.

We want to check against ID/EX.RegisterRt if it is an I-Type instruction, since this is the instruction that would be written to, but we want to check against ID/EX.RegisterRd if it is an R-Type instruction. To handle this, I use the output wire “RegDestMuxOut”, which comes from the target mux that chooses the appropriate register Rt or Rd depending on instruction type.

If the above conditional passes, I set IFIDWrite and PCWrite to 0, and HazardMux to 1 in order to set control lines to 0 and initiate a stall.

Task 3:
For 1 and 2-ahead hazard detection, I followed the conditional logic for data forwarding on the lecture slides, but compared registers from one stage prior to the ones listed on the slide, since the hazard detection unit is comparing registers in the ID stage instead of the EX stage like the forwarding unit.

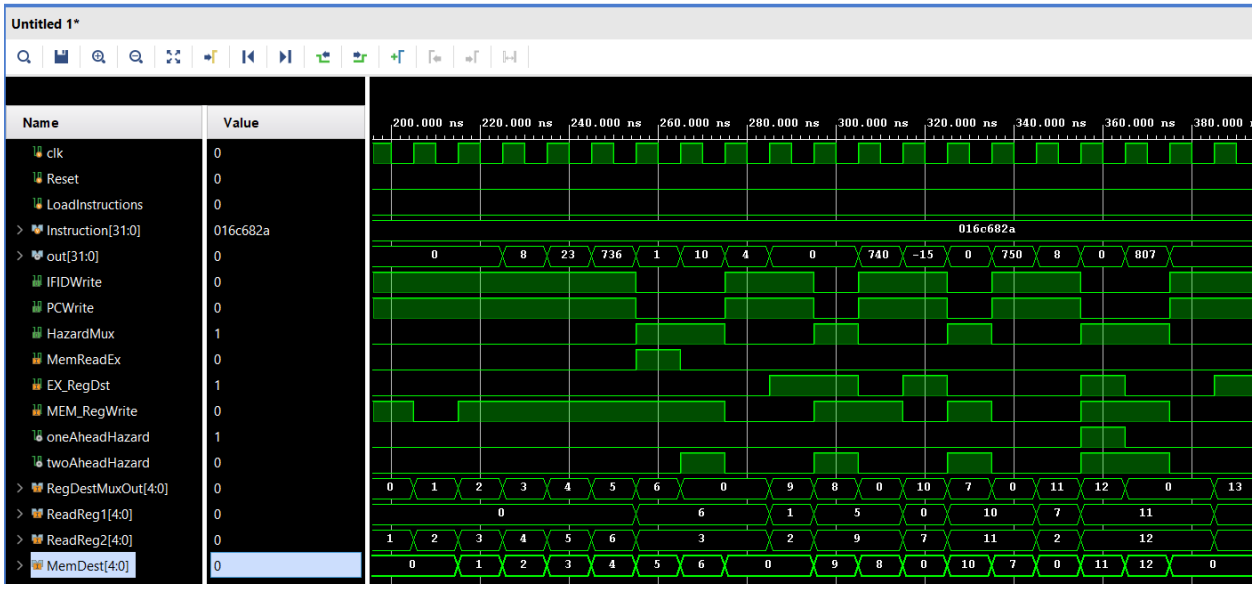
I chose to create two one-bit wires that act as flags for 1 and 2-ahead hazards being detected. The 1-ahead flag is tripped when a register in the EX stage is being written to, and either ID/EX.RegisterRt/Rd is equal to IF/ID.RegisterRs or IF/ID.RegisterRt. See below:

```
assign oneAheadHazard = (EX_RegDst && ((RegDestMuxOut == ReadReg1) ||
(RegDestMuxOut == ReadReg2))) ? 1 : 0;
```

The 2-ahead flag is tripped when a register in the MEM stage is being written to, and ID/EX.RegisterRt/Rd is not equal to IF/ID.RegisterRt/Rs, but EX/MEM.RegisterRd is equal to IF/ID.RegisterRt/Rs. See below:

```
assign twoAheadHazard = (MEM_RegWrite && ((RegDestMuxOut != ReadReg1
&& MemDest == ReadReg1) || (RegDestMuxOut != ReadReg2 && MemDest ==
ReadReg2))) ? 1 : 0;
```

Waveform:


Untitled 1*
Name
Value
clk
Reset
LoadInstructions
Instruction[31:0]
out[31:0]
IFIDWrite
PCWrite
HazardMux
MemReadEx
EX_RegDst
MEM_RegWrite
oneAheadHazard
twoAheadHazard
RegDestMuxOut[4:0]
ReadReg1[4:0]
ReadReg2[4:0]
MemDest[4:0]
0
0
0
016c682a
0
0
0
1
0
1
0
1
0
0
0
200.000 ns
220.000 ns
240.000 ns
260.000 ns
280.000 ns
300.000 ns
320.000 ns
340.000 ns
360.000 ns
016c682a
0 8 23 736 1 10 4 0 740 -15 0 750 8 0 807
0 1 2 3 4 5 6 0 9 8 0 10 7 0 11 12 0 13
0 6 1 5 0 10 7 11
1 2 3 4 5 6 3 2 9 7 11 2 12
0 1 2 3 4 5 6 0 9 8 0 10 7 0 11 12 0