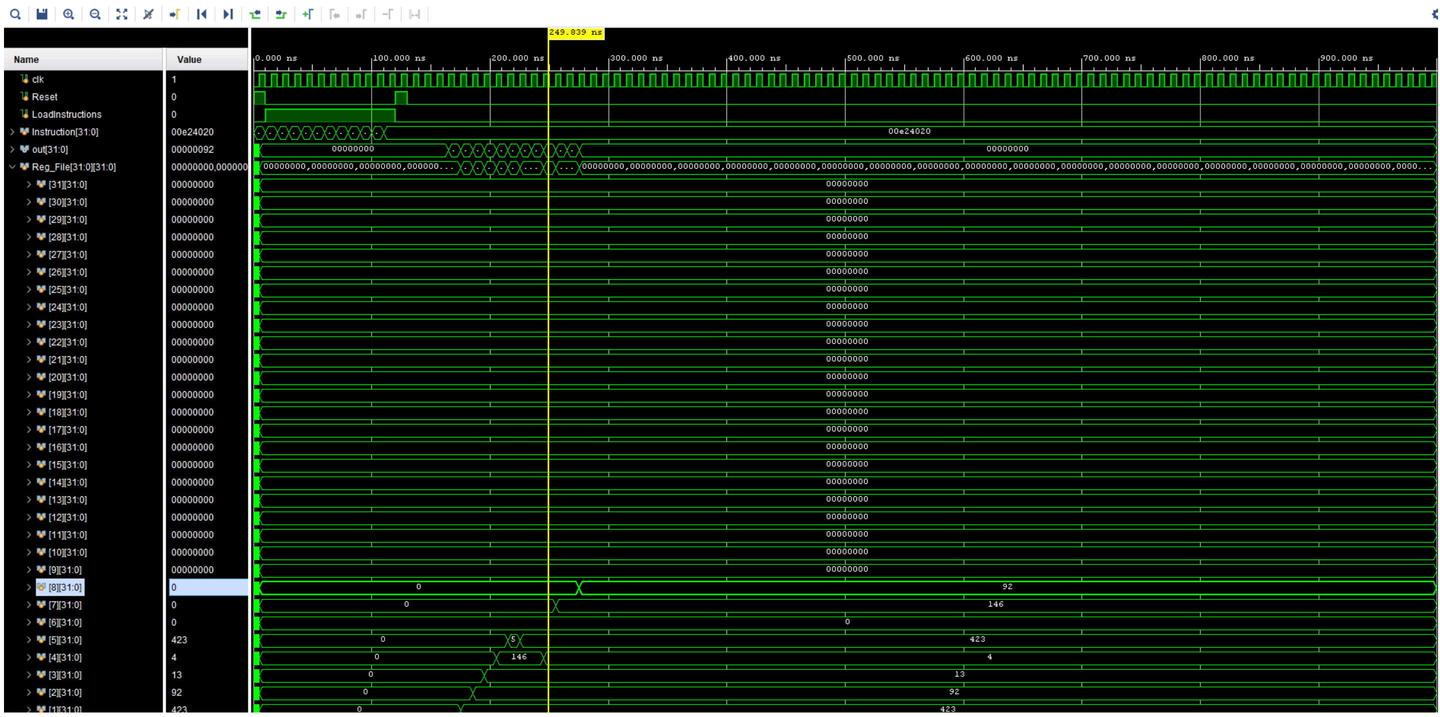
Marissa Ruiz and Maha Noor
EC413 Computer Organization
10 November 2023

Lab 7

1. Prelab: Synthesize the project and generate outputs for the given instruction sequence.



Name	Value
clk	1
Reset	0
LoadInstructions	0
Instruction[31:0]	00e24020
out[31:0]	00000092
Reg_File[31:0][31:0]	00000000,000000
[31][31:0]	00000000
[30][31:0]	00000000
[29][31:0]	00000000
[28][31:0]	00000000
[27][31:0]	00000000
[26][31:0]	00000000
[25][31:0]	00000000
[24][31:0]	00000000
[23][31:0]	00000000
[22][31:0]	00000000
[21][31:0]	00000000
[20][31:0]	00000000
[19][31:0]	00000000
[18][31:0]	00000000
[17][31:0]	00000000
[16][31:0]	00000000
[15][31:0]	00000000
[14][31:0]	00000000
[13][31:0]	00000000
[12][31:0]	00000000
[11][31:0]	00000000
[10][31:0]	00000000
[9][31:0]	00000000
[8][31:0]	0
[7][31:0]	0
[6][31:0]	0
[5][31:0]	423
[4][31:0]	4
[3][31:0]	13
[2][31:0]	92
[1][31:0]	423

The first instruction says to add immediate 0 and 234 to $R1. After 15 clock cycles, we see that $R1 updates accordingly. The same can be said for the next four instructions, which are given in the test bench. At this point, $R1-R$ should all be filled with inputs from the test bench. The next instruction, add $R5, $R1, $R4, also can be seen with a change in $R5 in the waveform, however, the resulting value is incorrect. This is because of a data hazard, the value of $R4 changes after the instruction is decoded, so the actual value of $R4 being added is 0 instead of the new value of $R4. The same happens for the following 5 instructions. Basically, there is a delay in data updates, so the operation yields incorrect results.

2. Add "1 ahead" forwarding

To add the "1 ahead" forwarding, we created a module called ForwardingUnit. This module works by taking in 3 registers, 2 of which are the Rs and Rt coming out of the ID/EX stage. The third register is the Rd EX/MEM stage. The forwarding unit checks to see if Rs or Rt is equal to Rd of EX/MEM and forwards the appropriate register. We wanted to make sure that the updated register is being forwarded.

3. Add “2 ahead” forwarding

2 ahead implements the same logic as 1 ahead except we also add the Rd from the MEM/WB stage. This checks if an updated register value is two stages ahead but still not caught by the actual register file.

4. Add arbitration logic for deciding between 1 and 2 ahead

To decide between 1 and 2 ahead forwarding, we used an if-else statement and checked to see if the register destination for both EXMEM and MEMWB were the same, and if they were, we forwarded the newer version (EXMEM).

5. Add logic for $0 write

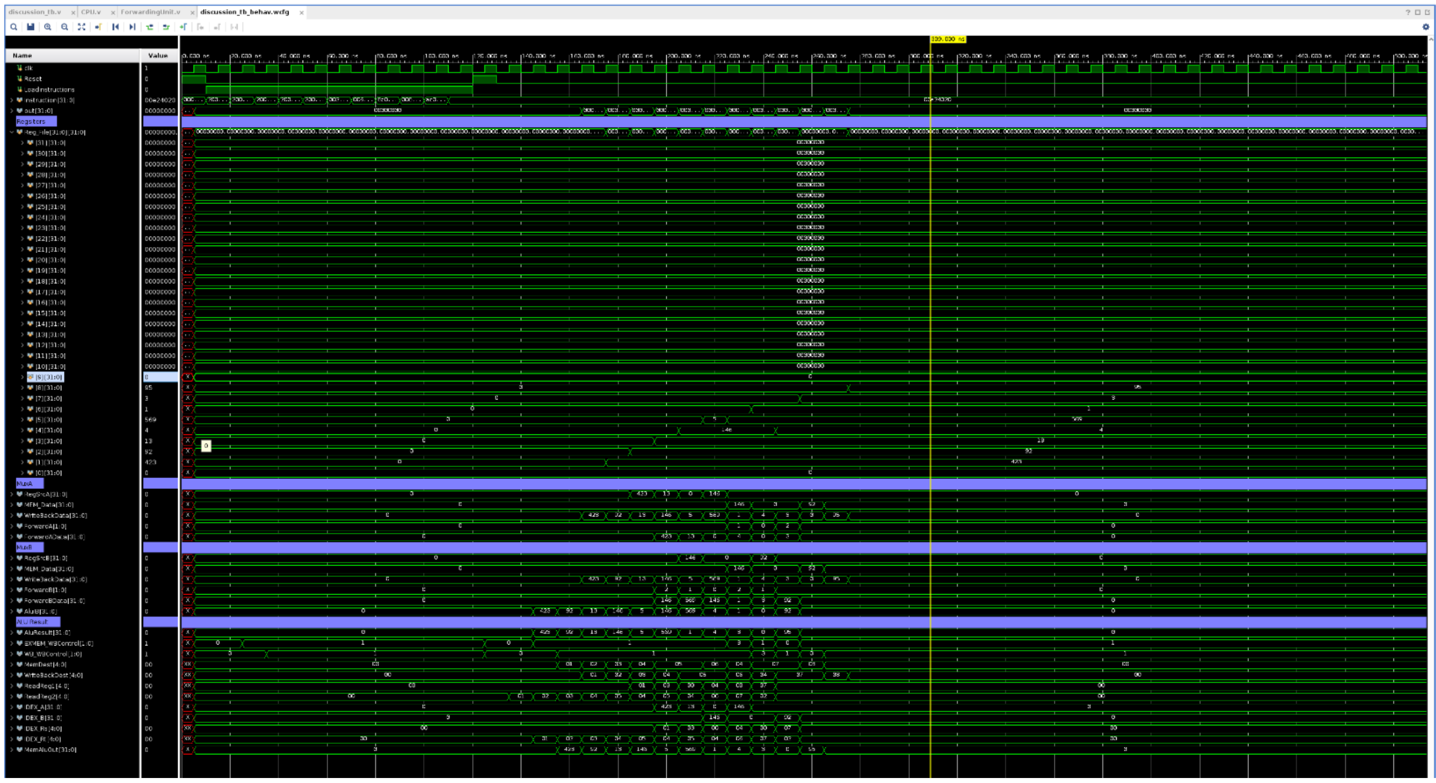
To check for register 0, we added a conditional statement checking to see if any of the Rds from EXMEM and MEMWB are 0. We forwarded it accordingly. The 0 Register always has to equal 0.

6. Add logic for no write

We added extra logic to our conditional statements checking to see if the Rds in EXMEM and MEMWB were actually being written to. Then, we can go ahead and forward that data. If the register is not being written to, we don’t forward that data because it won’t be correct.

7. Check register bypass works

To check if the register bypass works, we ran the simulation checked all the outputs, and made sure that they were correct according to the testbench. While debugging, we were able to find some errors with our logic regarding register bypass and we adjusted accordingly.



Name	Value
clk	1
Reset	0
LoadInstructions	0
instruction[31:0]	00e24020
out[31:0]	00000000
Registers	
Reg_File[31:0][31:0]	00000000,
[31][31:0]	00000000
[30][31:0]	00000000
[29][31:0]	00000000
[28][31:0]	00000000
[27][31:0]	00000000
[26][31:0]	00000000
[25][31:0]	00000000
[24][31:0]	00000000
[23][31:0]	00000000
[22][31:0]	00000000
[21][31:0]	00000000
[20][31:0]	00000000
[19][31:0]	00000000
[18][31:0]	00000000
[17][31:0]	00000000
[16][31:0]	00000000
[15][31:0]	00000000
[14][31:0]	00000000
[13][31:0]	00000000
[12][31:0]	00000000
[11][31:0]	00000000
[10][31:0]	00000000
[9][31:0]	0
[8][31:0]	95
[7][31:0]	3
[6][31:0]	1
[5][31:0]	569
[4][31:0]	4
[3][31:0]	13
[2][31:0]	92
[1][31:0]	423
[0][31:0]	0
MuxA	
RegSrcA[31:0]	0
MEM_Data[31:0]	0
WriteBackData[31:0]	0
ForwardA[1:0]	0
ForwardAData[31:0]	0
MuxB	
RegSrcB[31:0]	0
MEM_Data[31:0]	0
WriteBackData[31:0]	0
ForwardB[1:0]	0
ForwardBData[31:0]	0
AluB[31:0]	0
ALU Result	
AluResult[31:0]	0
EXMEM_WBControl[1:0]	1
WB_WBControl[1:0]	1
MemDest[4:0]	00
WriteBackDest[4:0]	00
ReadReg1[4:0]	00
ReadReg2[4:0]	00
IDEX_A[31:0]	0
IDEX_B[31:0]	0
IDEX_Rs[4:0]	00
IDEX_Rt[4:0]	00
MemAluOut[31:0]	0

Checked with TAs in lab hours.