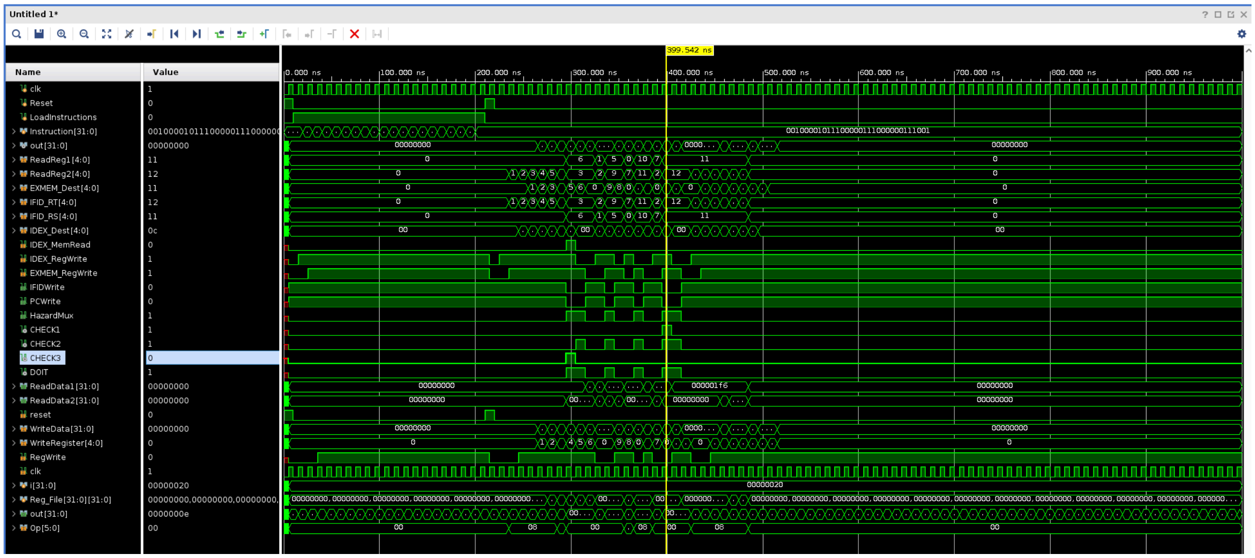
EC 413 LAB 8

Siara Patel, Vanshika Chaddha

Waveform:



Name	Value
clk	1
Reset	0
LoadInstructions	0
Instruction[31:0]	0010000101110000011100000
out[31:0]	00000000
ReadReg1[4:0]	11
ReadReg2[4:0]	12
EXMEM_Dest[4:0]	11
IFID_RT[4:0]	12
IFID_RS[4:0]	11
IDEX_Dest[4:0]	0c
IDEX_MemRead	0
IDEX_RegWrite	1
EXMEM_RegWrite	1
IFIDWrite	0
PCWrite	0
HazardMux	1
CHECK1	1
CHECK2	1
CHECK3	0
DOIT	1
ReadData1[31:0]	00000000
ReadData2[31:0]	00000000
reset	0
WriteData[31:0]	00000000
WriteRegister[4:0]	0
RegWrite	0
clk	1
i[31:0]	00000020
Reg_File[31:0][31:0]	00000000,00000000,00000000,
out[31:0]	0000000e
Op[5:0]	00

Check One: One Ahead

A one ahead hazard (read after write) occurs when the previous instruction is writing to a register that the current instruction is reading from — it essentially needs a value that a prior instruction has not written yet to the register.

The way we implemented our **Check One** is by first seeing if IDEX_RegWrite is equal to 1 (if the previous instruction is writing for a register). Then we have 2 separate cases — one for I type and another for R type.

The first check looks at whether IDEX_Regwrite is high (meaning that the previous instruction is writing to a register) and whether !IDEX_MemRead (which means it's not a LW). Then we go on to whether the OP = 00 (R type) and IDEX_Dest != 0 (because a NOP essentially just sets the register to 0). If it is an R type, we compare the IDEX_Dest value with both IDEX_RS and IDEX_RS. If not, then we only compare IDEX_RT with IDEX_Dest.

CHECK1 = (IDEX_RegWrite && !IDEX_MemRead) && ((Op == 2'b00 && IDEX_Dest != 0 && ((IFID_RS == IDEX_Dest) || (IFID_RT == IDEX_Dest))) || (Op != 2'b00 && IDEX_Dest != 0 && (IFID_RS == IDEX_Dest)));

Check Two: Two Ahead

A two ahead hazard occurs when an incoming instruction depends on a register written to from two previous instructions. This is when an incoming instruction relies on an instruction in the memory stage. To implement this check, we check if IDEX_RegWrite is one and that IDEX_MemRead is low. This ensures that we aren't performing a LW. We compare the incoming registers IFID_RS and IFID_RT to the destination register in the EXMEM stage (two stages ahead). We also check to see if the incoming instruction is R-type or I-type with the same logic implemented for one ahead (check1). The only difference is we use the destination register from the EX/MEM stage.

Check Three: Load Hazard

A load use hazard occurs due to the fact that lw writes into its destination register in the MEM stage. The subsequent instruction may need that register when it enters the EX stage but the register has not updated with the new value. Our Check 3 detects this hazard. It first looks at whether the instruction is a LW (IDEX_MemRead is high). Then it looks at whether IDEX_Dest != 0 as a NOP essentially sets a register to 0. It then compares IFID_RS with IDEX_Dest