Rv32ic

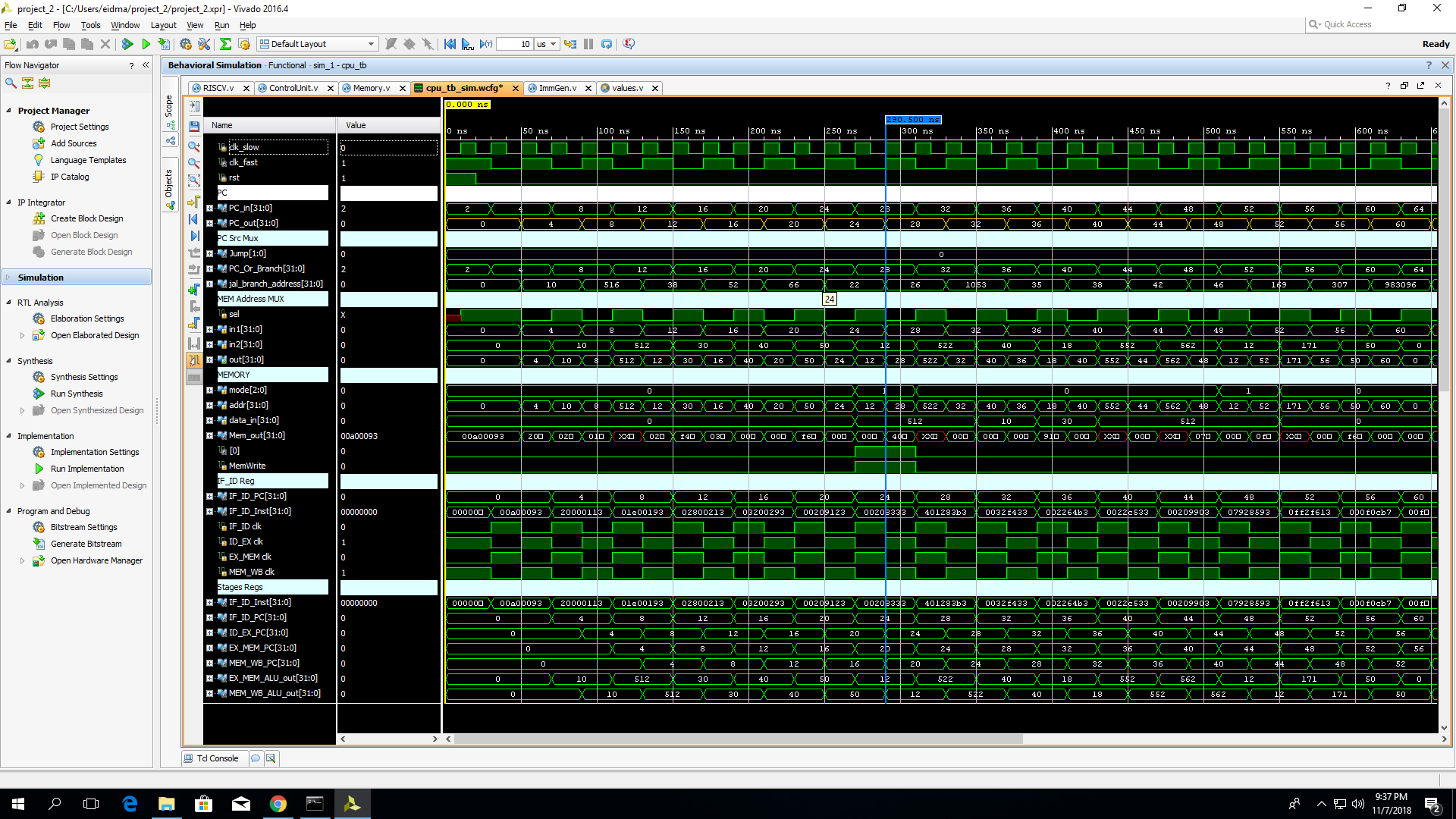
Mazen Amr Eid 900161021, Mohamed Ads 900142564, Khaled Soliman 900162031

RISC-V Integer and Compressed Instruction Implementation

# Testing

To ensure our CPU works fine and executes every instruction correctly, we were testing and debugging on the testcase 2 and some of the instructions of testcase 1 provided for us by the doctor on BlackBoard. This is a sample of our simulation:

## R, I, U, UJ, S, SB Formats Testing:



.text

li x1, 10

li x2, 512

li x3, 30

li x4, 40

li x5, 50

test\_R:

sh x2, 2(x1) #20

add x6, x1, x2

sub x7, x5, x1

and x8, x5, x3

or x9, x4, x2

xor x10, x5, x2 #40

lh x18, 2(x1)

test\_I:

addi x11, x5, 121

andi x12, x5, 0xff

lui x25, 0xf0 # 56

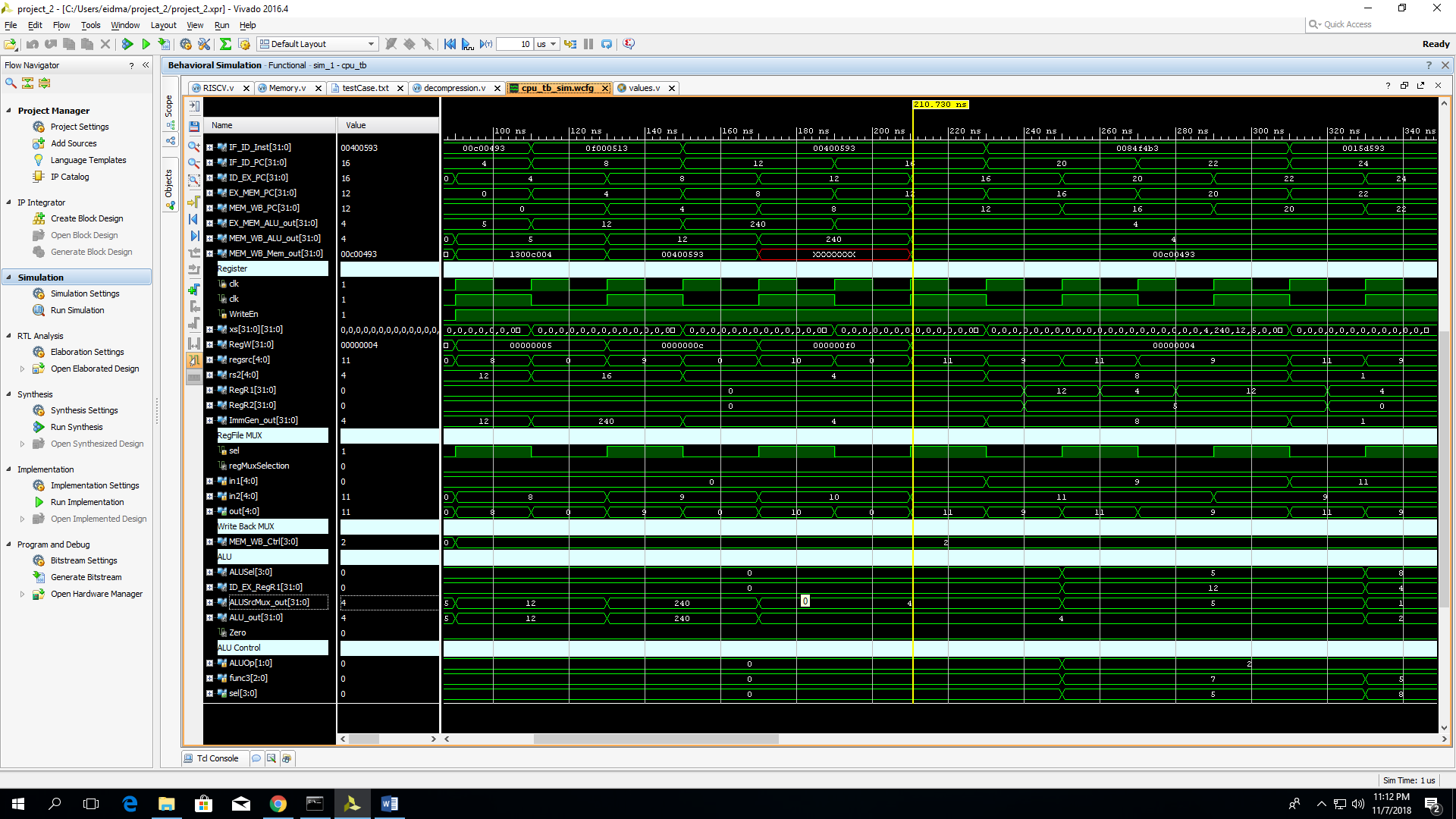
lui x26, 0xff0

ori x13, x5, 0x121

slli x14, x5, 3

srli x15, x5, 3

## b. Compressed Testing



li x8, 5

li x9, 12

li x10, 240

li x11, 4

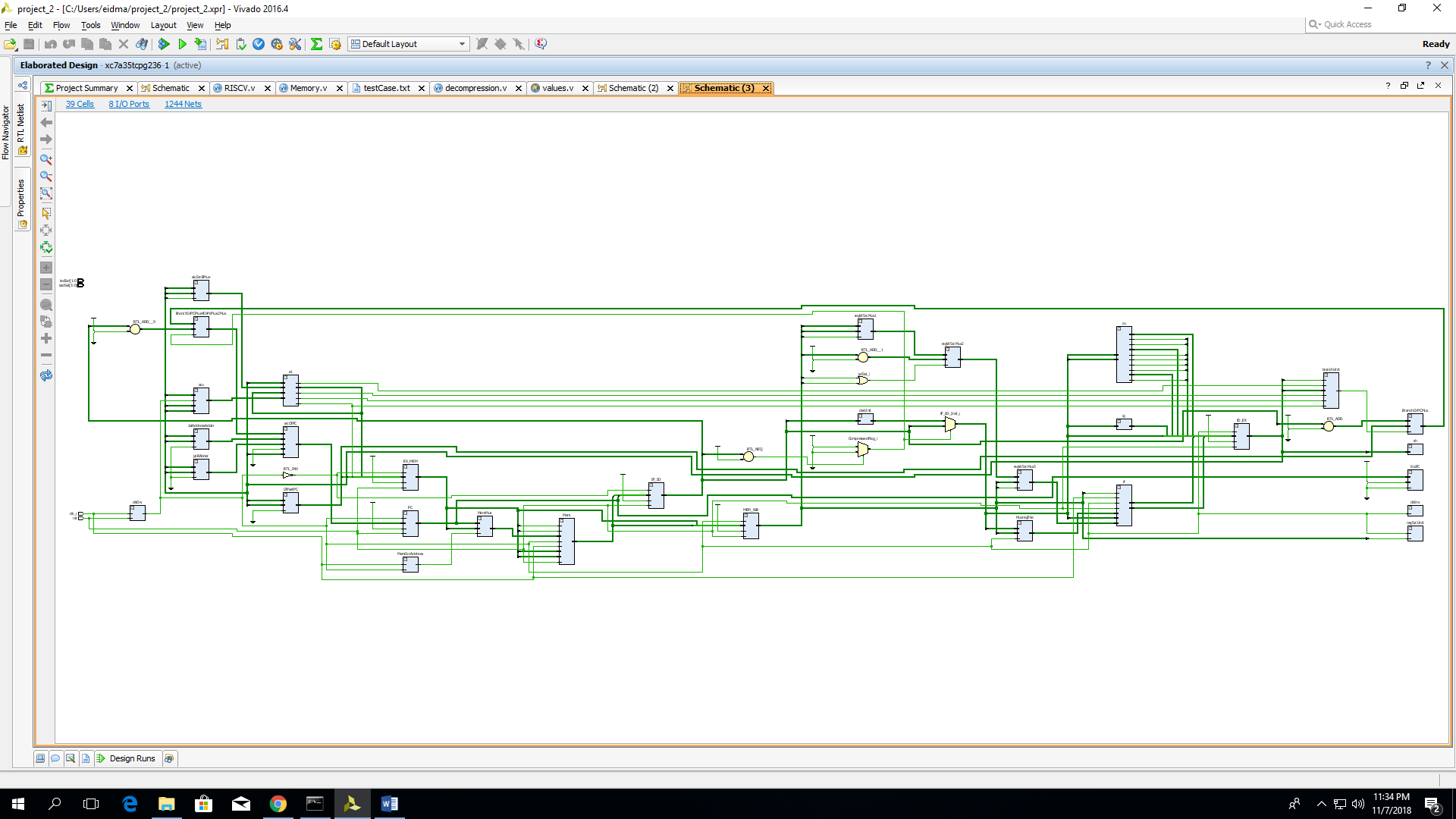
c.and x9, x9, x8

c.srli x11, x11, 1

# Extending Compressed Instructions Design

We read a normal word (the instruction to be fetched) from the memory and in the decode stage, we take this fetched instruction and see if it is a compressed one. In the decode stage, we implemented a unit called “decompression”, this takes the least significant 16 bits of the fetched instruction and decompress it. We check if it is a compressed instruction by ensuring that the least significant 2 bits of the instruction are not 2’b11; if so, then this is a compressed one and sets a compressed flag to 1’b1. Consequentially, if this flag is high, then we choose the decompressed version of the instruction and increment the PC by only 2 (instead of 4). Otherwise, we take the instruction as is and increment the PC by 4. The flow of the instruction continues as normal and gets executed.

# DataPath Diagram:



# Limitations

* We didn’t yet implement or tested the following instructions: auipc, c.addi4spn, c.addi16sp, c.j, c.jr, c.beqz, c.bnez, c.mv.
* We didn’t integrate our forwarding unit yet.
* No enough time to finish the assigned milestones.