Week 10

* 1. Write a SV program for Control units

a)

module Main\_control(input logic [6:0] opcode,

output logic regWrite, output logic ALUSrc,

output logic memRead,

output logic memWrite,

output logic branch,

output logic [1:0] ALUOp, output logic mem\_reg

);

always\_comb begin

regWrite = 0;

mem\_reg=0;

ALUSrc = 0;

memRead = 0;

memWrite = 0;

branch = 0;

ALUOp = 2'b00;

case (opcode)

7'b0110011: begin // R-type instructions

regWrite = 1;

ALUOp = 2'b10;

end

7'b0010011: begin // I-type instructions

regWrite = 1;

ALUSrc = 1;

ALUOp = 2'b00;

end

7'b0000011: begin // Load instructions

regWrite = 1;

memRead = 1;

ALUSrc = 1;

mem\_reg=1;

end

7'b0100011: begin // Store instructions

memWrite = 1;

ALUSrc = 1;

end

7'b1100011: begin // Branch instructions

branch = 1;

ALUOp = 2'b01;

end

default: begin

regWrite=0;

ALUSrc=0;

memRead=0;

memWrite=0;

branch=0;

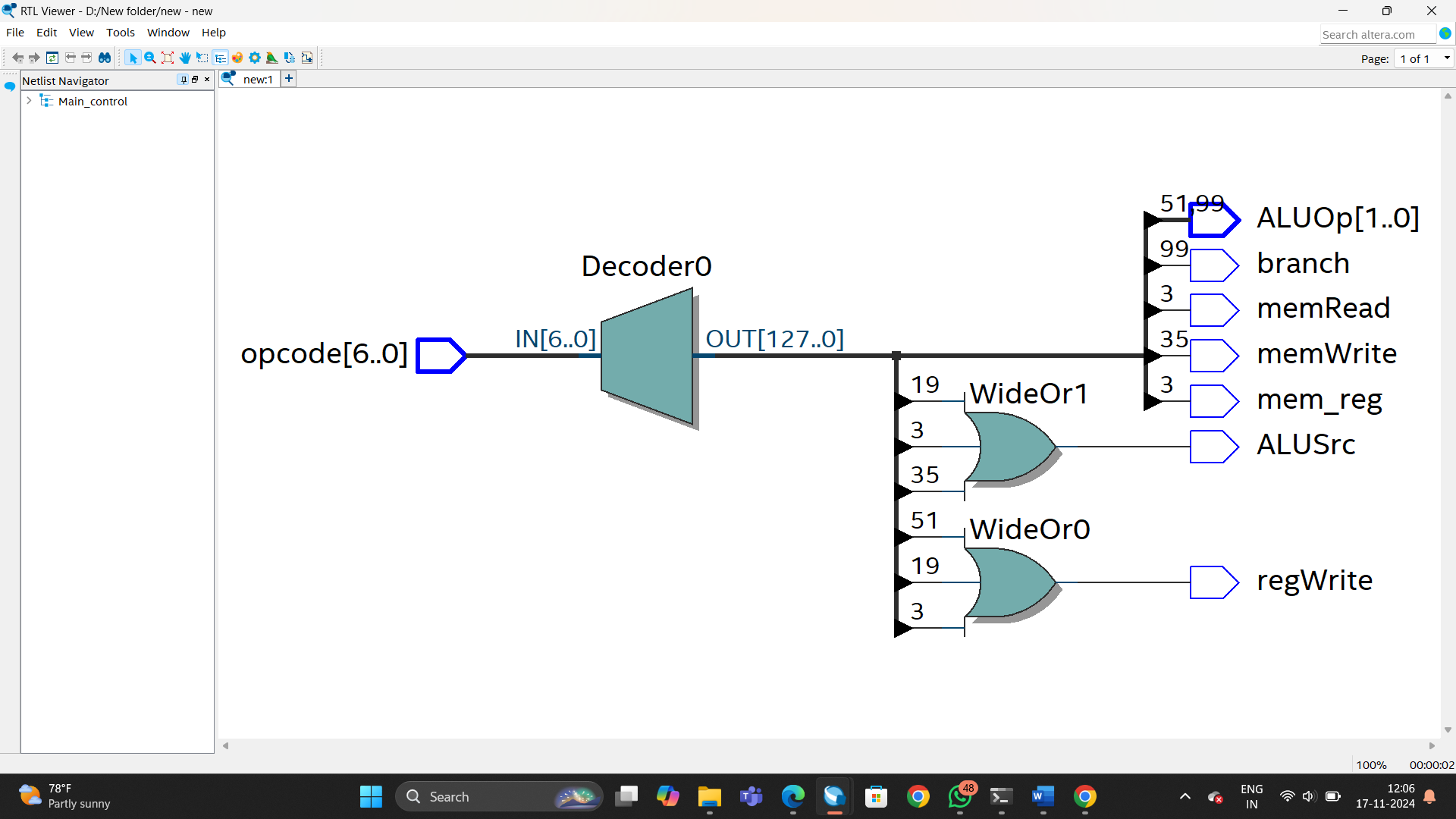
ALUOp=0;

end

endcase

end

endmodule



b)

module Alu\_control( input logic [1:0] ALUOp,

input logic [2:0] funct3,

input logic funct7\_5,

output logic [3:0] ALUControl

);

always\_comb begin

ALUControl = 4'b1111;

if (ALUOp == 2'b00) begin

ALUControl = 4'b0010; // ADD

end

else if (ALUOp == 2'b01) begin

ALUControl = 4'b0110; // SUB

end

else if (ALUOp == 2'b10) begin

case (funct3)

3'b000: ALUControl = funct7\_5 ? 4'b0110 : 4'b0010; // SUB if funct7\_5=1, ADD if funct7\_5=0

3'b111: ALUControl = 4'b0000; // AND

3'b110: ALUControl = 4'b0001; // OR

default: ALUControl = 4'b1111; // Undefined

endcase

end

end

endmodule

