Digital Design and Computer Organization Laboratory
Hackathon 2024

3rd Semester, Academic Year 2024

Date: 20/11/2024

Name:	SRN:	Section: H
Rithvik Rajesh Matta	PES2UG23CS485	Team: 5
Rishi A Sheth	PES2UG23CS479	
Rohan	PES2UG23CS489	
Ritesh M	PES2UG23CS483	

1. Design a iverilog code for ALU, register file and concatenate Alu – Reg for

- 8-bit processor which will take 2-bit op code to perform only logical (and, or, Nand ,nor) operation
- Memory should have 8 registers of each size of eight bit and save the result in one of the registers and perform read operation to know the output.

DESIGN FILE CODE:

Hackathon > new > processor.v

```
module ALU (
    input [7:0] A,
    input [7:0] B,
    input [1:0] opcode,
    output reg [7:0] result
);
    always @(*) begin
        case (opcode)
            2'b00: result = A & B;
            2'b01: result = A | B;
            2'b10: result = ~(A & B);
            2'b11: result = ~(A | B);
            default: result = 8'b00000000;
        endcase
    end
endmodule

module RegisterFile (
    input clk,
    input [2:0] write_addr,
    input [7:0] write_data,
    input write_enable,
    input [2:0] read_addr,
    output reg [7:0] read_data
);
    reg [7:0] registers [0:7];

    always @(posedge clk) begin
        if (write_enable) begin
            registers[write_addr] <= write_data;
        end
    end

    always @(*) begin
        read_data = registers[read_addr];
    end
endmodule

module Processor (
    input clk,
    input [7:0] A,
    input [7:0] B,
    input [1:0] opcode,
    input [2:0] write_addr,
    input write_enable,
    input [2:0] read_addr,
    output [7:0] read_data
);
    wire [7:0] alu_result;

    ALU alu (
        .A(A),
        .B(B),
        .opcode(opcode),
        .result(alu_result)
    );

    RegisterFile reg_file (
        .clk(clk),
        .write_addr(write_addr),
        .write_data(alu_result),
        .write_enable(write_enable),
        .read_addr(read_addr),
        .read_data(read_data)
    );
endmodule
```

TESTBENCH FILE CODE:

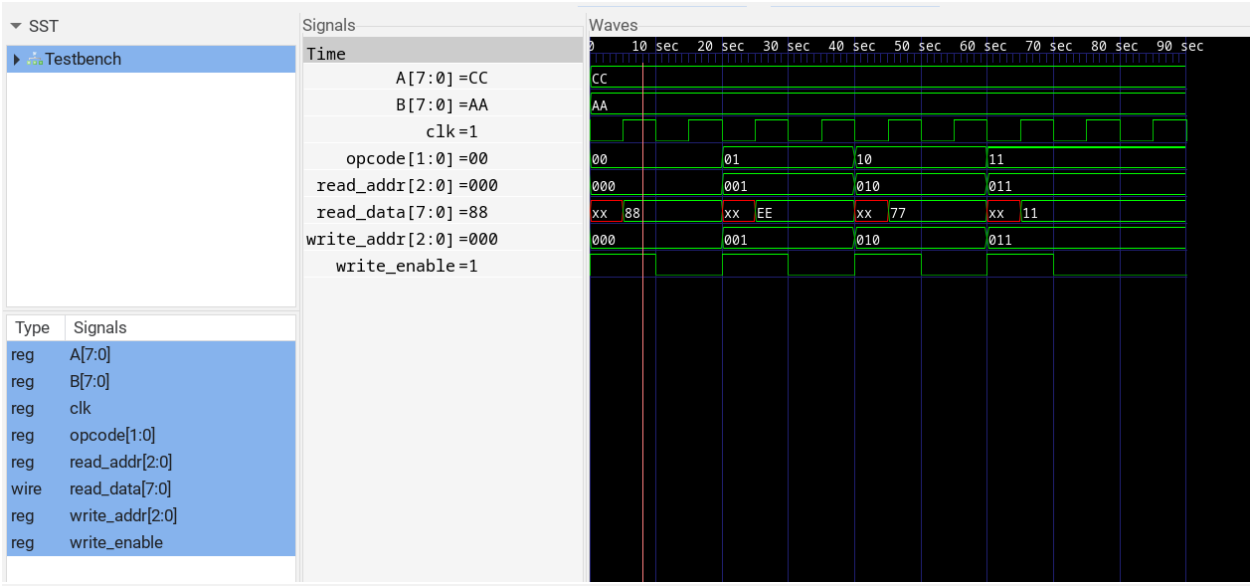
Hackathon > new > processor_tb.v

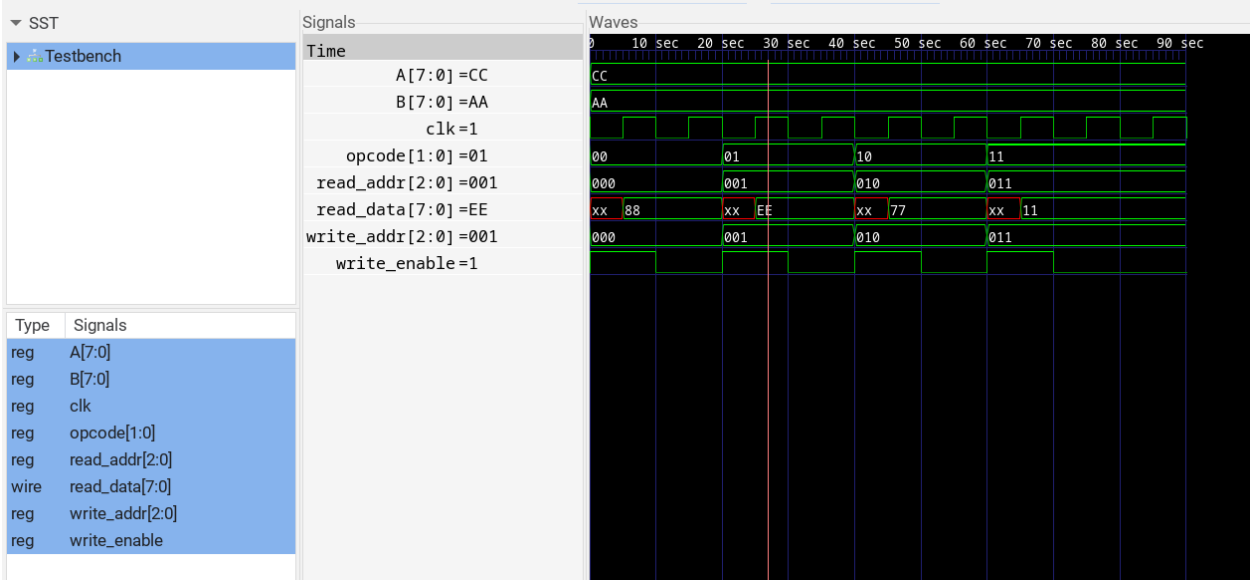
```
module Testbench;reg clk;reg [7:0] A;reg [7:0] B;reg [1:0] opcode;reg [2:0] write_addr;reg write_enable;reg [2:0] read_addr;wire [7:0] read_data;

    Processor uut (.clk(clk),.A(A),.B(B),.opcode(opcode),.write_addr(write_addr),.write_enable(write_enable),.read_addr(read_addr),.read_data(read_data)
    );

    initial begin
        clk = 0;
        forever #5 clk = ~clk;
    end

    initial begin
        $dumpfile("processor_tb.vcd");
        $dumpvars(0, Testbench);
    end

    initial begin
        A = 8'b11001100; B = 8'b10101010; opcode = 2'b00;
        write_addr = 3'b000; write_enable = 1; read_addr = 3'b000;
        #10;
        write_enable = 0;
        #10;
        $display("Inputs: A = %b, B = %b, Opcode = %b", A, B, opcode);
        $display("AND Result: %b", read_data);

        A = 8'b11001100; B = 8'b10101010; opcode = 2'b01;
        write_addr = 3'b001; write_enable = 1; read_addr = 3'b001;
        #10;
        write_enable = 0;
        #10;
        $display("Inputs: A = %b, B = %b, Opcode = %b", A, B, opcode);
        $display("OR Result: %b", read_data);

        A = 8'b11001100; B = 8'b10101010; opcode = 2'b10;
        write_addr = 3'b010; write_enable = 1; read_addr = 3'b010;
        #10;
        write_enable = 0;
        #10;
        $display("Inputs: A = %b, B = %b, Opcode = %b", A, B, opcode);
        $display("NAND Result: %b", read_data);

        A = 8'b11001100; B = 8'b10101010; opcode = 2'b11;
        write_addr = 3'b011; write_enable = 1; read_addr = 3'b011;
        #10;
        write_enable = 0;
        #10;
        $display("Inputs: A = %b, B = %b, Opcode = %b", A, B, opcode);
        $display("NOR Result: %b", read_data);

        #10;
        $finish;
    end
endmodule
```

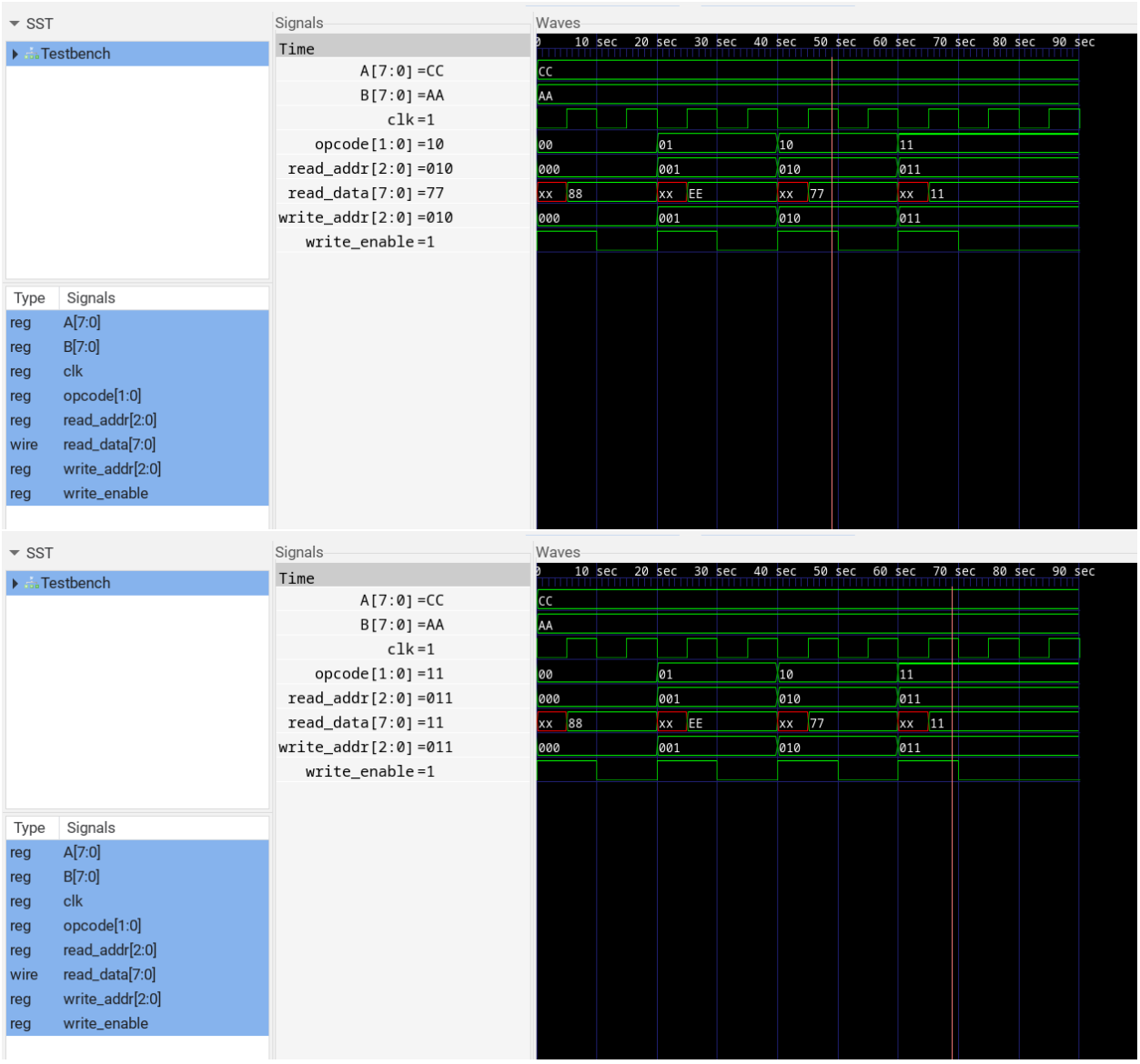
TERMINAL:

```
rtk5@penguin:~/PES/DDCO-sem-3/Hackathon/new$ iverilog -o test processor.v processor_tb.v
rtk5@penguin:~/PES/DDCO-sem-3/Hackathon/new$ vvp test
VCD info: dumpfile processor_tb.vcd opened for output.
Inputs: A = 11001100, B = 10101010, Opcode = 00
AND Result: 10001000
Inputs: A = 11001100, B = 10101010, Opcode = 01
OR Result: 11101110
Inputs: A = 11001100, B = 10101010, Opcode = 10
NAND Result: 01110111
Inputs: A = 11001100, B = 10101010, Opcode = 11
NOR Result: 00010001
rtk5@penguin:~/PES/DDCO-sem-3/Hackathon/new$ gtkwave processor_tb.vcd

GTKWave Analyzer v3.3.118 (w)1999-2023 BSI

[0] start time.
[90] end time.
GTKWAVE | Touch screen detected, enabling gestures.
Exiting.
rtk5@penguin:~/PES/DDCO-sem-3/Hackathon/new$
```

GTKWAVE:


SST
Testbench
Type
Signals
reg A[7:0]
reg B[7:0]
reg clk
reg opcode[1:0]
reg read_addr[2:0]
wire read_data[7:0]
reg write_addr[2:0]
reg write_enable
Signals
Time
A[7:0] =CC
B[7:0] =AA
clk =1
opcode[1:0] =00
read_addr[2:0] =000
read_data[7:0] =88
write_addr[2:0] =000
write_enable =1
Waves
0 10 sec 20 sec 30 sec 40 sec 50 sec 60 sec 70 sec 80 sec 90 sec
CC
AA
00 01 10 11
000 001 010 011
xx 88 xx EE xx 77 xx 11
000 001 010 011


SST
Testbench
Type
Signals
reg A[7:0]
reg B[7:0]
reg clk
reg opcode[1:0]
reg read_addr[2:0]
wire read_data[7:0]
reg write_addr[2:0]
reg write_enable
Signals
Time
A[7:0] =CC
B[7:0] =AA
clk =1
opcode[1:0] =01
read_addr[2:0] =001
read_data[7:0] =EE
write_addr[2:0] =001
write_enable =1
Waves
0 10 sec 20 sec 30 sec 40 sec 50 sec 60 sec 70 sec 80 sec 90 sec
CC
AA
00 01 10 11
000 001 010 011
xx 88 xx EE xx 77 xx 11
000 001 010 011


SST
Testbench
Type
Signals
reg A[7:0]
reg B[7:0]
reg clk
reg opcode[1:0]
reg read_addr[2:0]
wire read_data[7:0]
reg write_addr[2:0]
reg write_enable
Signals
Time
A[7:0] =CC
B[7:0] =AA
clk =1
opcode[1:0] =10
read_addr[2:0] =010
read_data[7:0] =77
write_addr[2:0] =010
write_enable =1
Waves
0 10 sec 20 sec 30 sec 40 sec 50 sec 60 sec 70 sec 80 sec 90 sec
CC
AA
00 01 10 11
000 001 010 011
xx 88 xx EE xx 77 xx 11
000 001 010 011

SST
Testbench
Type
Signals
reg A[7:0]
reg B[7:0]
reg clk
reg opcode[1:0]
reg read_addr[2:0]
wire read_data[7:0]
reg write_addr[2:0]
reg write_enable
Signals
Time
A[7:0] =CC
B[7:0] =AA
clk =1
opcode[1:0] =11
read_addr[2:0] =011
read_data[7:0] =11
write_addr[2:0] =011
write_enable =1
Waves
0 10 sec 20 sec 30 sec 40 sec 50 sec 60 sec 70 sec 80 sec 90 sec
CC
AA
00 01 10 11
000 001 010 011
xx 88 xx EE xx 77 xx 11
000 001 010 011