Digital Design and Computer Organization Laboratory

UE23CS251A

3rd Semester, Academic Year 2024-25

TEAM NO.-09

Date:20/11/2024

Name:	SRN:	Section:
M Niranjan	PES2UG23CS308	E
Keerthan P.V	PES2UG23CS272	
Mallikarjun K	PES2UG23CS321	
Mahesh I.A	PES2UG23CS316	

Problem Statement:

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 register of each size of eight bit and save the result in one of the register and perform read operation to know the output.

(a) alu_register_file.v

```
module ALU (
    input [1:0] opcode,
    input [7:0] A,
    input [7:0] B,
    output [7:0] result
);
    assign result = (opcode == 2'b00) ? (A & B) :
                    (opcode == 2'b01) ? (A | B) :
                    (opcode == 2'b10) ? ~(A & B) :
                    (opcode == 2'b11) ? ~(A | B) :
                    8'b00000000;
endmodule

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

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

    assign read_data = registers[read_reg];
endmodule

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

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

    RegisterFile regfile_inst (
        .clk(clk),
        .write_reg(write_reg),
        .write_data(alu_result),
        .write_enable(write_enable),
        .read_reg(read_reg),
        .read_data(read_data)
    );
endmodule
```

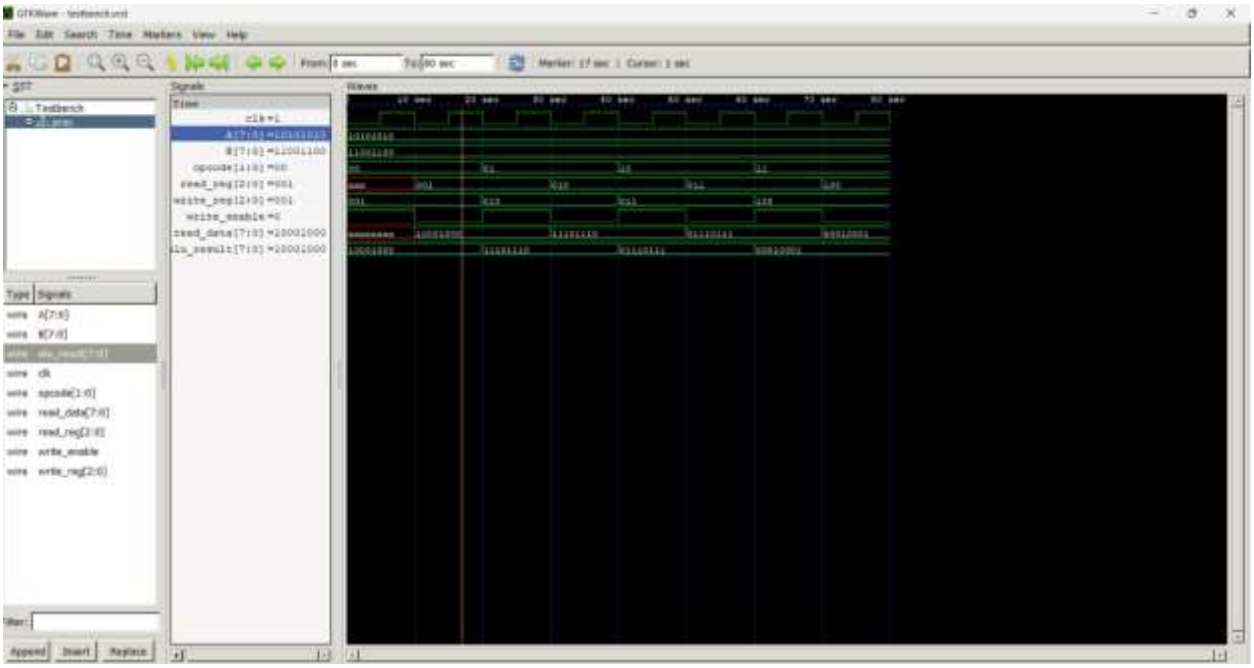
(b) tb_ alu_register_file.v

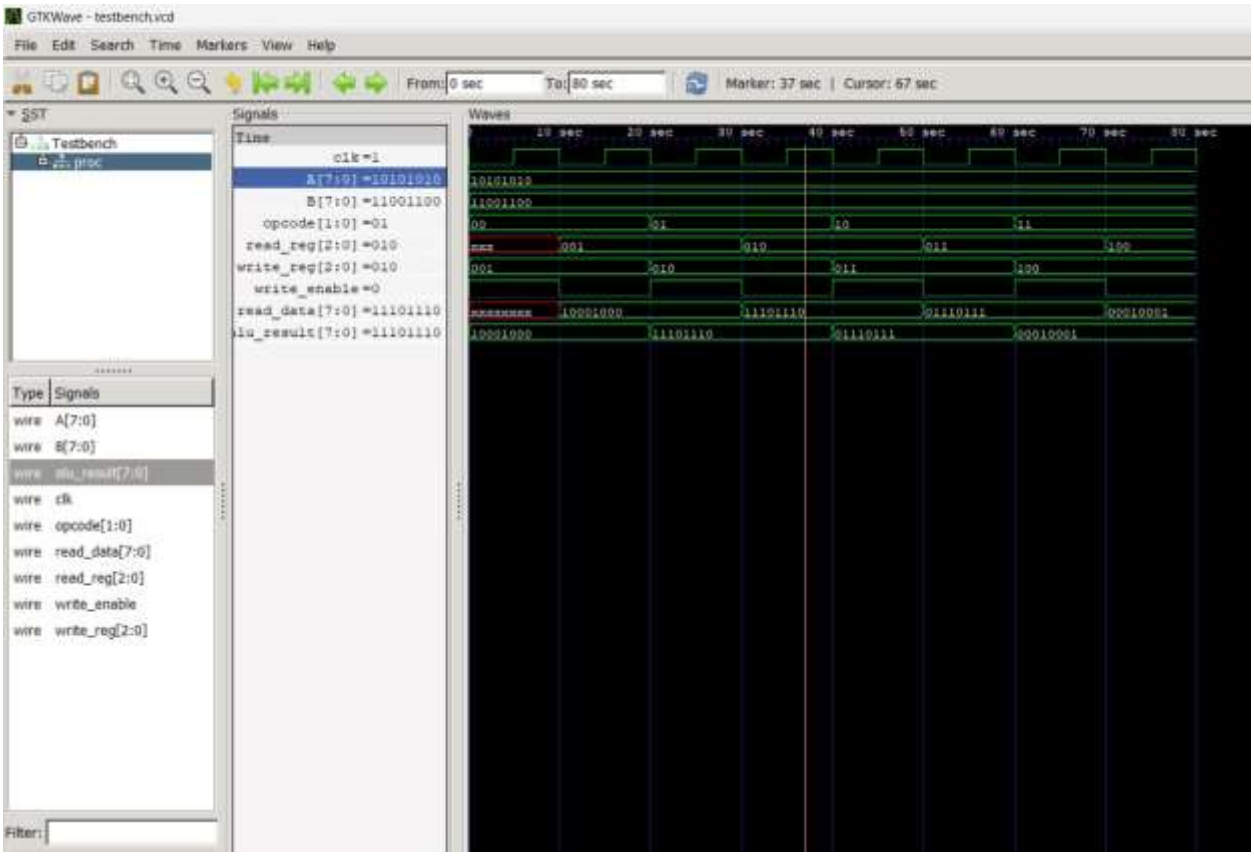
```
module Testbench;
    reg clk;
    reg [1:0] opcode;
    reg [7:0] A, B;
    reg [2:0] write_reg, read_reg;
    reg write_enable;
    wire [7:0] read_data;

    Processor proc (
        .clk(clk),
        .opcode(opcode),
        .A(A),
        .B(B),
        .write_reg(write_reg),
        .write_enable(write_enable),
        .read_reg(read_reg),
        .read_data(read_data)
    );

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

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

        $monitor("At time %t, opcode = %b, A = %b, B = %b, write_reg = %b, read_reg = %b, read_data = %b",
                 $time, opcode, A, B, write_reg, read_reg, read_data);

        opcode = 2'b00;
        A = 8'b10101010;
        B = 8'b11001100;
        write_reg = 8'b001;
        write_enable = 1;
        #10;

        read_reg = 8'b001;
        write_enable = 0;
        #10;

        opcode = 2'b01;
        A = 8'b10101010;
        B = 8'b11001100;
        write_reg = 8'b010;
        write_enable = 1;
        #10;

        read_reg = 8'b010;
        write_enable = 0;
        #10;

        opcode = 2'b10;
        A = 8'b10101010;
        B = 8'b11001100;
        write_reg = 8'b011;
        write_enable = 1;
        #10;

        read_reg = 8'b011;
        write_enable = 0;
        #10;

        opcode = 2'b11;
        A = 8'b10101010;
        B = 8'b11001100;
        write_reg = 8'b100;
        write_enable = 1;
        #10;

        read_reg = 8'b011;
        write_enable = 0;
        #10;

        opcode = 2'b11;
        A = 8'b10101010;
        B = 8'b11001100;
        write_reg = 8'b100;
        write_enable = 1;
        #10;

        read_reg = 8'b100;
        write_enable = 0;
        #10;

        $finish;
    end
endmodule
```

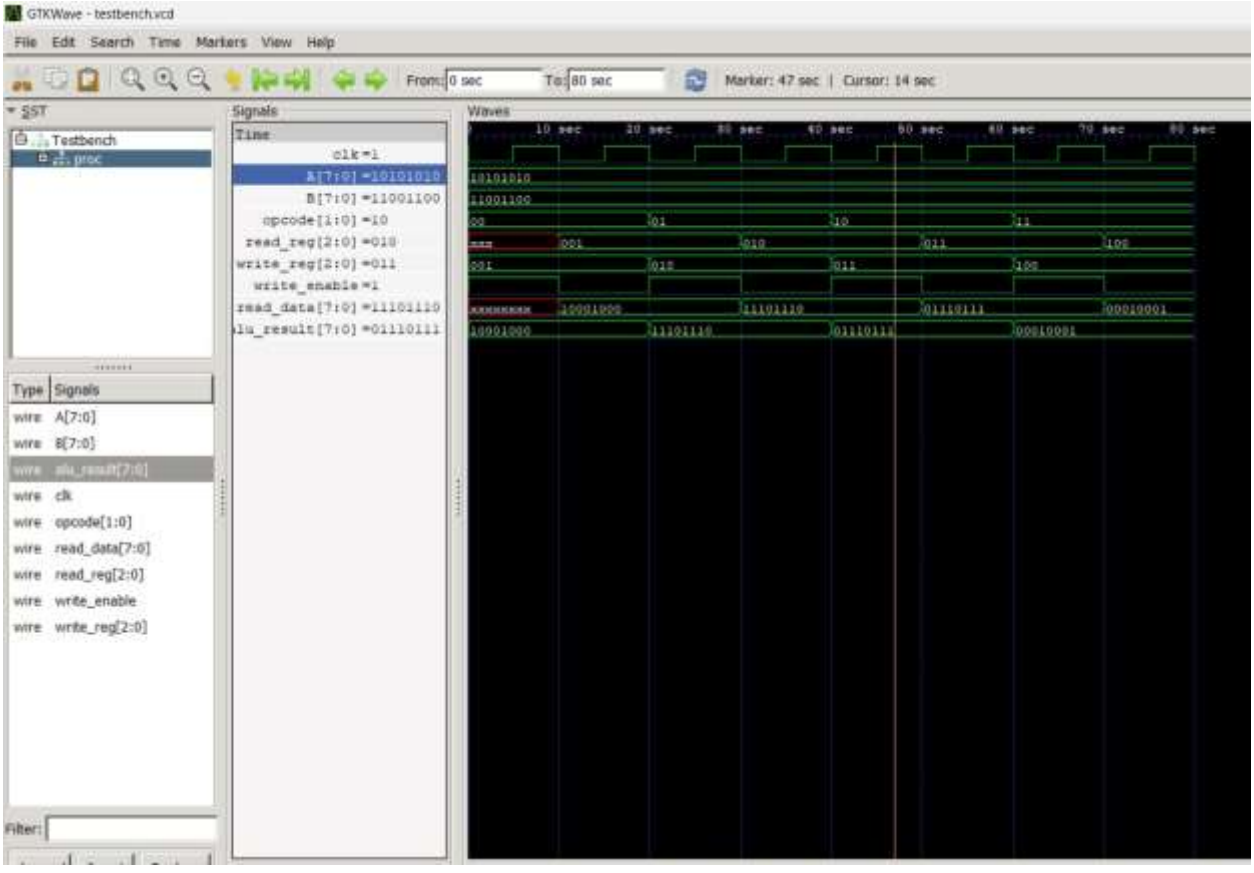
(c).vvp output

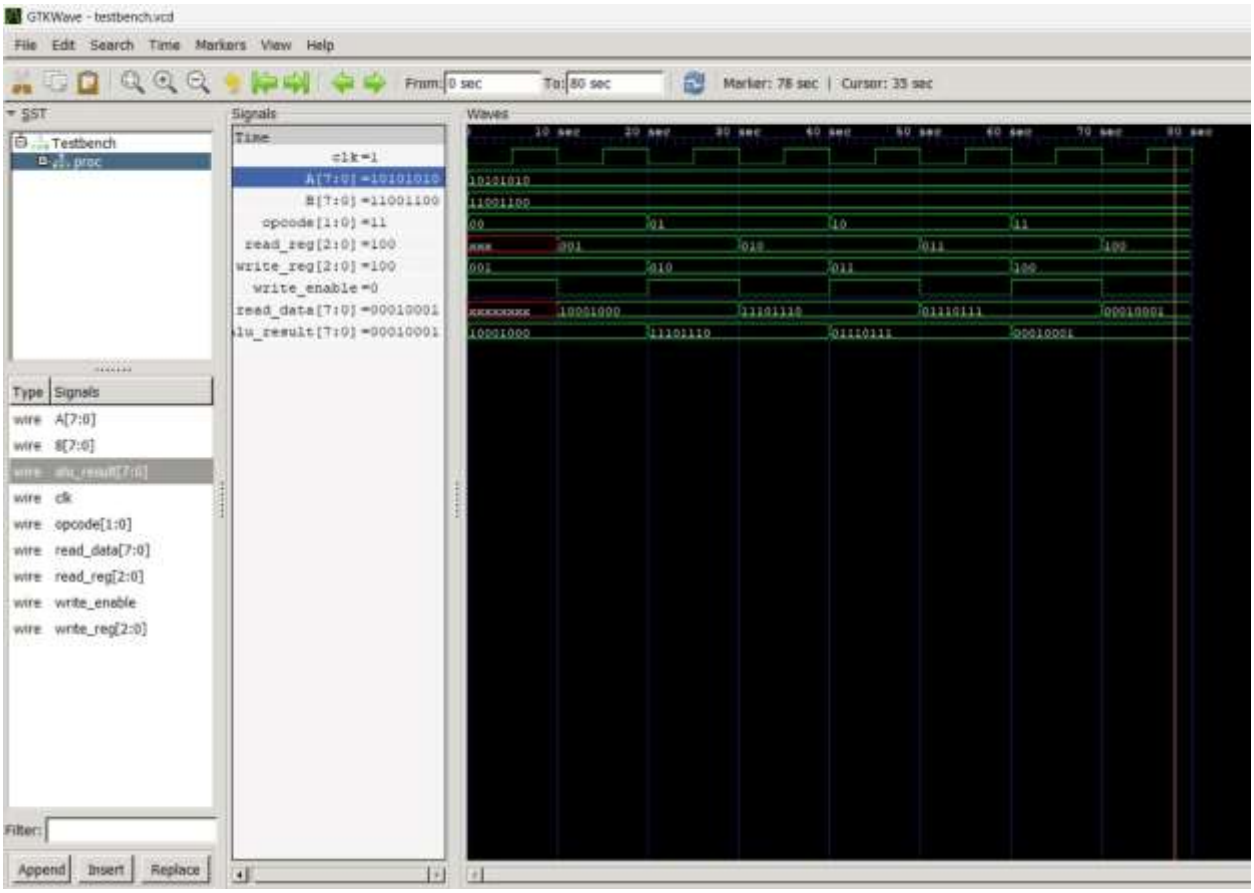
```
PS C:\iverilog\bin> iverilog -o test alu_register_file.v tb_alu_register_file.v
PS C:\iverilog\bin> vvp test
VCD info: dumpfile testbench.vcd opened for output.
At time                    0, opcode = 00, A = 10101010, B = 11001100, write_reg = 001, read_reg = xxx, read_data = xxxxxxxx
At time                   10, opcode = 00, A = 10101010, B = 11001100, write_reg = 001, read_reg = 001, read_data = 10001000
At time                   20, opcode = 01, A = 10101010, B = 11001100, write_reg = 010, read_reg = 001, read_data = 10001000
At time                   30, opcode = 01, A = 10101010, B = 11001100, write_reg = 010, read_reg = 010, read_data = 11101110
At time                   40, opcode = 10, A = 10101010, B = 11001100, write_reg = 011, read_reg = 010, read_data = 11101110
At time                   50, opcode = 10, A = 10101010, B = 11001100, write_reg = 011, read_reg = 011, read_data = 01110111
At time                   60, opcode = 11, A = 10101010, B = 11001100, write_reg = 100, read_reg = 011, read_data = 01110111
At time                   70, opcode = 11, A = 10101010, B = 11001100, write_reg = 100, read_reg = 100, read_data = 00010001
```

(d)gtkwaveform




GTKWave - testbench.vcd
File Edit Search Time Markers View Help
From: 0 sec
To: 80 sec
Marker: 37 sec | Cursor: 67 sec
SST
Testbench
proc
Signals
Time
clk=1
A[7:0]=10101010
B[7:0]=11001100
opcode[1:0]=01
read_reg[2:0]=010
write_reg[2:0]=010
write_enable=0
read_data[7:0]=11101110
alu_result[7:0]=11101110
Waves
10 sec
20 sec
30 sec
40 sec
50 sec
60 sec
70 sec
80 sec
Type Signals
wire A[7:0]
wire B[7:0]
wire alu_result[7:0]
wire clk
wire opcode[1:0]
wire read_data[7:0]
wire read_reg[2:0]
wire write_enable
wire write_reg[2:0]
Filter:


GTKWave - testbench.vcd
File Edit Search Time Markers View Help
From: 0 sec
To: 80 sec
Marker: 47 sec | Cursor: 14 sec
SST
Testbench
proc
Signals
Time
clk=1
A[7:0]=10101010
B[7:0]=11001100
opcode[1:0]=10
read_reg[2:0]=010
write_reg[2:0]=011
write_enable=1
read_data[7:0]=11101110
alu_result[7:0]=01110111
Waves
10 sec
20 sec
30 sec
40 sec
50 sec
60 sec
70 sec
80 sec
Type Signals
wire A[7:0]
wire B[7:0]
wire alu_result[7:0]
wire clk
wire opcode[1:0]
wire read_data[7:0]
wire read_reg[2:0]
wire write_enable
wire write_reg[2:0]
Filter:


GTKWave - testbench.vcd
File Edit Search Time Markers View Help
From: 0 sec
To: 80 sec
Marker: 78 sec | Cursor: 35 sec
SST
Testbench
proc
Type Signals
wire A[7:0]
wire B[7:0]
wire clk
wire opcode[1:0]
wire read_data[7:0]
wire read_reg[2:0]
wire write_enable
wire write_reg[2:0]
Filter:
Append
Insert
Replace
Signals
Time
clk=1
A[7:0]=10101010
B[7:0]=11001100
opcode[1:0]=11
read_reg[2:0]=100
write_reg[2:0]=100
write_enable=0
read_data[7:0]=00010001
alu_result[7:0]=00010001
Waves
10 sec
20 sec
30 sec
40 sec
50 sec
60 sec
70 sec
80 sec