ES204 Digital Systems
LAB Assignment - 9

Indian Institute of Technology, Gandhinagar
March 20, 2024

Marks : 50

Submission instructions:

Only one student from the team will submit with the word doc name Rollno1_Rollno2.pdf. The PDF will contain the code, testbench and simulation results.

1. **[50 Marks**]

In the last lab you have designed 32-bit Register File containing 8 registers. Extend this to 16 registers. Now, make this register file *dual-port read.* This means that we should be able to give in 2 sets of addresses and read two register contents simultaneously. The data read from the registers should be passed to the ALU which will perform arithmetic and logical operations on these registers' contents. The results (output of the ALU) will be stored back into the Register file.

For instance, we will implement ADD R1, R2, R2 instruction:
This instruction takes in the contents of R1 and R2, adds them and transfers the result (of addition) to R2. There is a separate 1-bit register called Cy which stores the Carry output. The same 1-bit register stores the borrow for SUB instruction.

For this lab:

You are to implement the following instructions:

Arithmetic instruction:
1000: SUB Rx, Ry, Rz

Logical instruction:
1010: XOR Rx, Ry, Rz

(where x, y, and z would be any values between 0..15)

Here, 1000 and 1010 are the operation codes which are used to differentiate between different operations. In this case, they differentiate between SUB and XOR.

Lab 9 ES 204

Birudugadda Srivibhav (22110050)
Reddybathuni Venkat (22110220)

Register file with ALU

1. Code

```
`timescale 1ns / 1ps
module op_codes #(parameter n = 32, k = 16)(
    input clk, reset,en,
    input [3:0] op_code,
    input [3:0] ADDR0, ADDR1, ADDR2
);

reg [n - 1:0] registers [0:k - 1];
reg [n-1:0] X, Y;
reg [n-1:0] D;
reg Bout = 0;
reg [n:0] B;
integer i;
always @(posedge clk) begin
    if (en & reset == 0) begin
        registers[0] <= 32'b000;
        registers[1] <= 32'b001;
        registers[2] <= 32'b010;
        registers[3] <= 32'b011;
        registers[4] <= 32'b100;
        registers[5] <= 32'b101;
        registers[6] <= 32'b110;
        registers[7] <= 32'b111;
        registers[8] <= 32'b1000;
        registers[9] <= 32'b1001;
        registers[10] <= 32'b1010;
        registers[11] <= 32'b1011;
        registers[12] <= 32'b1100;
        registers[13] <= 32'b1101;
        registers[14] <= 32'b1110;
        registers[15] <= 32'b1111;
```

```
   end else if (reset & en & op_code == 4'b1000) begin
        X = registers[ADDR0];
        Y = registers[ADDR1];
        B[0] = 1;
        for (i=0; i<n; i=i+1)
        begin
        D[i] = X[i] ^ ~Y[i] ^ B[i];
        B[i+1] = (X[i] & ~Y[i] ) | (X[i] & B[i] ) | (B[i] & ~Y[i]);
        end
        if(B!= 0)
        begin
        Bout = ~B[n];
        end
        registers[ADDR2] = D;
        end
      else if (reset & en & op_code == 4'b1010) begin
       Bout <= 0;
       registers[ADDR2] <= registers[ADDR0] ^ registers[ADDR1];
       end
end

endmodule
```

2. Test bench

```
`timescale 1ns / 1ps
module op_codes_tb();
reg clk, reset, en;
reg [3:0]op_code;
reg [3:0] ADDR0,ADDR1,ADDR2;

op_codes uut(.clk(clk), .reset(reset), .en(en), .op_code(op_code), .ADDR0(ADDR0),
.ADDR1(ADDR1), .ADDR2(ADDR2));

initial
clk = 1;
always #2 clk = ~clk;

initial
```

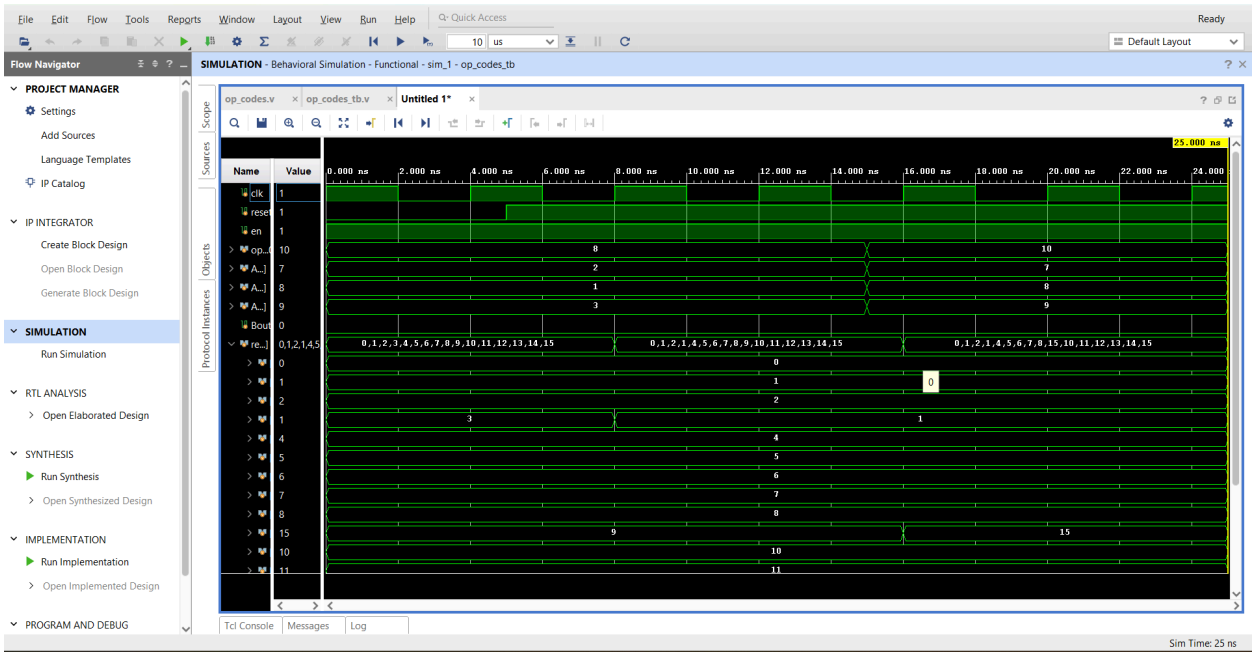
```
begin
en = 1;
op_code = 4'b1000;
ADDR0 = 4'b0010;
ADDR1 = 4'b0001;
ADDR2 = 4'b0011;
reset = 0;
#5;
reset = 1;
#10;
op_code = 4'b1010;
ADDR0 = 4'b0111;
ADDR1 = 4'b1000;
ADDR2 = 4'b1001;
#10;
$finish();
end
endmodule
```

3. Simulation

First we apply sub operation on register 2 and 1 and store the value in register 3;

Later we applied xor on register 7 and 8 and stored in 9.

XOR of 1000 and 0111 is 1111 which is 15 as shown in figure.


File Edit Flow Tools Reports Window Layout View Run Help
Quick Access
Ready
Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
PROGRAM AND DEBUG
SIMULATION - Behavioral Simulation - Functional - sim_1 - op_codes_tb
op_codes.v
op_codes_tb.v
Untitled 1*
Scope
Sources
Objects
Protocol Instances
Name
Value
clk
reset
en
op...
A...]
A...]
A...]
Bout
re...]
25.000 ns
0.000 ns
2.000 ns
4.000 ns
6.000 ns
8.000 ns
10.000 ns
12.000 ns
14.000 ns
16.000 ns
18.000 ns
20.000 ns
22.000 ns
24.000
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
0,1,2,1,4,5,6,7,8,9,10,11,12,13,14,15
0,1,2,1,4,5,6,7,8,15,10,11,12,13,14,15
Tcl Console
Messages
Log
Sim Time: 25 ns