ELC 2137 Lab 9: Lab 9 ALU with Input Register

Makenna Meyers

April 2, 2020

Summary

Results

Table 1: *register* expected results table

Time (ns):	0-5	5-10	10-15	15-20	20-25	25-30	30-35	35-40	40-45	45-50	50-55
D (hex)	0	0	A	A	3	3	0	0	0→6	6	6
clk	0	1	0	1	0	1	0	1	0	1	0
en	0	0	1	1	1→0	0→1	1→0	0	0→1	1	1
rst	0	0→1	0	0	0	0	0	0	0	0	0
Q (hex)	X	X→0	0	A	A	A	A	A	A	6	6

Table 2: *alu* expected results table

Time (ns):	0-10	10-20	20-30	30-40	40-50	50-60
in0	1	3	5	1	1	8
in1	2	1	7	1	1	6
op	0	1	2	3	4	5
out	3	2	5	1	0	8

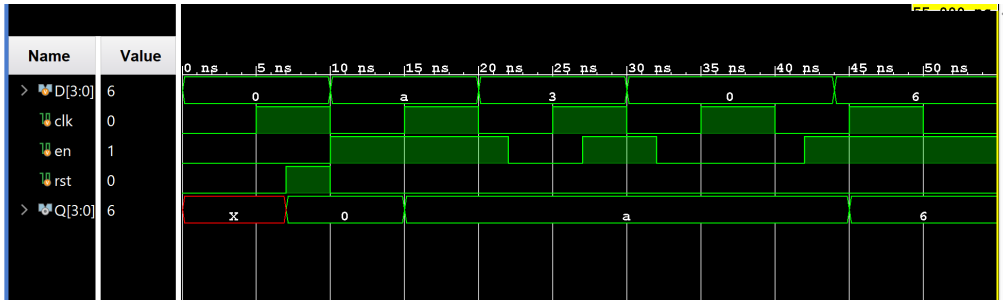

Figure 1: Register Simulation Waveform

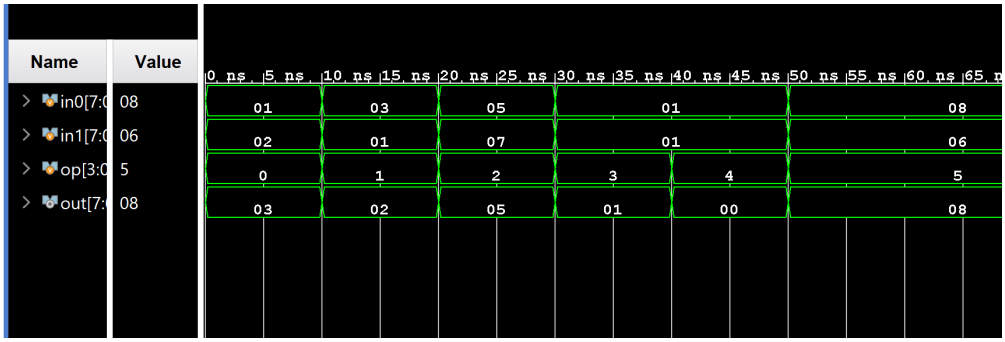

Figure 2: ALU Simulation Waveform

Code

Listing 1: Register Module

```
`timescale 1ns / 1ps
// Makenna Meyers , ELC 2137 , 2020-4-01


module register #(parameter N=1)
    (
    input clk, rst, en,
    input [N-1:0] D,
    output reg [N-1:0] Q
    );

    always @(posedge clk, posedge rst)
    begin
        if (rst==1)
            Q <= 0 ;
        else if (en==1)
            Q <= D ;
    end
endmodule
```

Listing 2: Register Test Bench

```
`timescale 1ns / 1ps
// Makenna Meyers , ELC 2137 , 2020-4-01


module register_test ();

    reg [3:0] D;
    reg clk , en , rst ;
    wire [3:0] Q;

    register #(.N(4)) r(.D(D), .clk(clk),
        .en(en), .rst(rst), .Q(Q) );

    // clock runs continuously
    always begin
        clk = ~clk ; #5;
    end

    // this block only runs once
    initial begin
    clk =0; en =0; rst =0; D=4'h0; #7;
    rst = 1; #3; // reset
    D = 4'hA; en = 1; rst = 0; #10;
    D = 4'h3; #2;
    en = 0; #5;
    en = 1; #3;
    D = 4'h0; #2;
    en = 0; #10;
    en = 1; #2;
```

```
    D = 4'h6; #11;
    $finish ;
    end
endmodule
```

Listing 3: ALU Module

```
`timescale 1ns / 1ps
// Makenna Meyers , ELC 2137 , 2020-4-01


module alu #( parameter N =8)
    (
    output reg [N -1:0] out ,
    input [N -1:0] in0 ,
    input [N -1:0] in1 ,
    input [3:0] op
    );

    // Local parameters
    parameter ADD =0;
    parameter SUB =1;
    parameter AND =2;
    parameter OR =3;
    parameter XOR =4;

    always @*
    begin
        case (op)
            ADD: out = in0 + in1 ;
            SUB: out = in0 - in1 ;
            AND: out = in0 & in1 ;
            OR: out = in0 | in1 ;
            XOR: out = in0 ^ in1 ;
            default : out = in0 ;
        endcase
    end
endmodule
```

Listing 4: ALU Test Bench

```
`timescale 1ns / 1ps
// Makenna Meyers , ELC 2137 , 2020-4-01


module alu_test ();

    reg [7:0]in0;
    reg [7:0]in1;
    reg [3:0]op;
    wire [7:0]out;

    alu #(.N(8)) a(.in0(in0), .in1(in1), .op(op), .out(out));
```

```
    initial begin
        op = 0;
        in0 = 1;
        in1 = 2;
        #10
        op = 1;
        in0 = 3;
        in1 = 1;
        #10
        op = 2;
        in0 = 5;
        in1 = 7;
        #10
        op = 3;
        in0 = 1;
        in1 = 1;
        #10
        op = 4;
        in0 = 1;
        in1 = 1;
    end
endmodule
```