ELC 2137 Lab #9: ALU

Xingpeng Yi

April 1, 2020

Summary

This lab explain the difference between combinational and regular sequential logic, describe the operation of an SR latch, D latch, D flip-flop, and D register, describe the differences in Verilog procedural blocks for combinational versus sequential logic, and import and modify modules from a previous project and use them design a modular system. The lab required students to create an ArithmeticLogic Unit (ALU) capable of a few operations. In order to do mathematical operations with the ALU, we need two numbers. One of these will come from the switches. For the other, we will use a use a register to store a number.

Q&A

None.

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	6	6	6

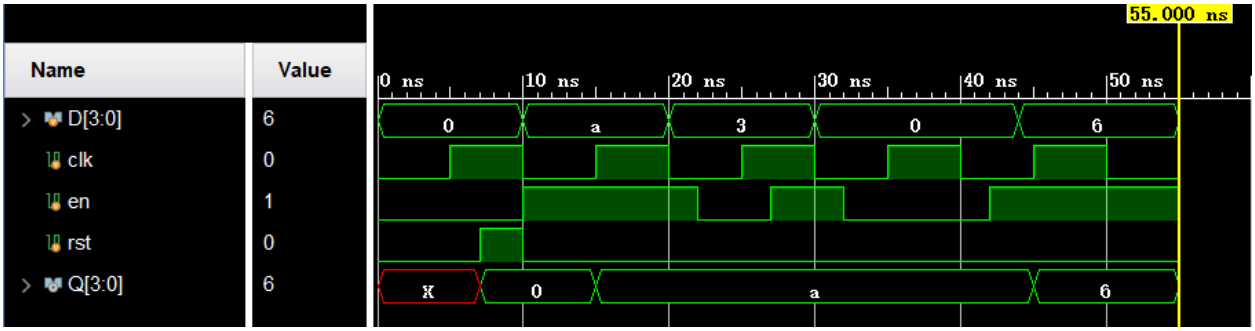

Figure 1: Register module simulation waveform.

Table 2: *alu* expected results table skeleton

Time (ns):	0-10	10-20	20-30	30-40	40-50
in0	00	01	00	00	01
in1	01	00	00	01	01
op	0	1	2	3	4
out	01	01	00	01	00

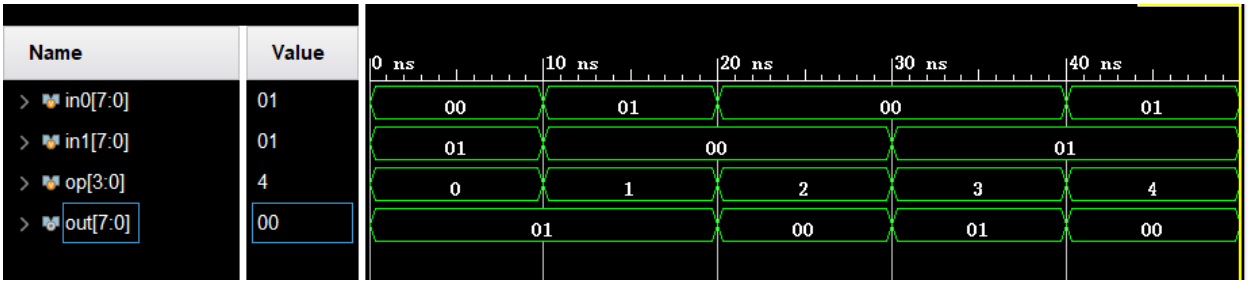

Figure 2: ALU module simulation waveform.

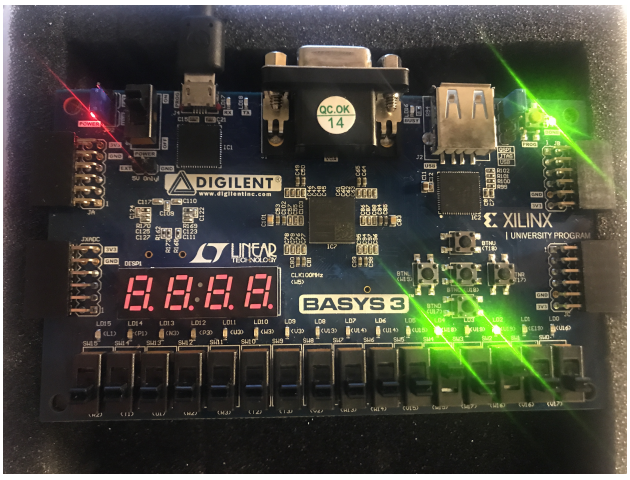

Figure 3: Basys 3 board step1.

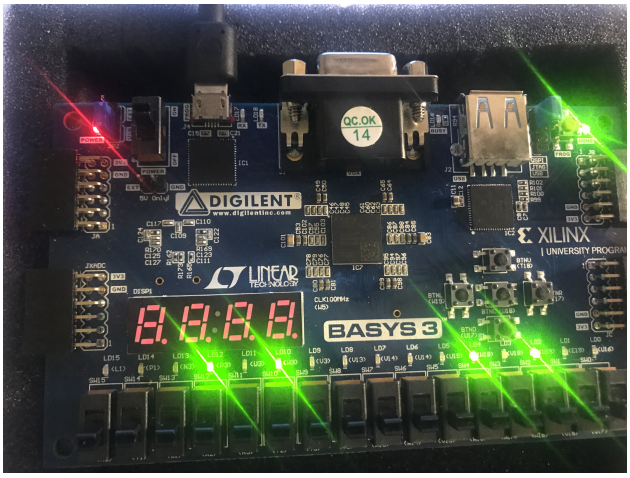

Figure 4: Basys 3 board step2.

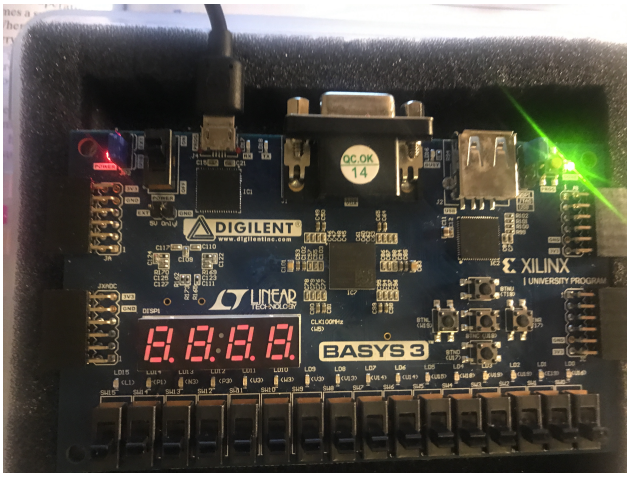

Figure 5: Basys 3 board step3.

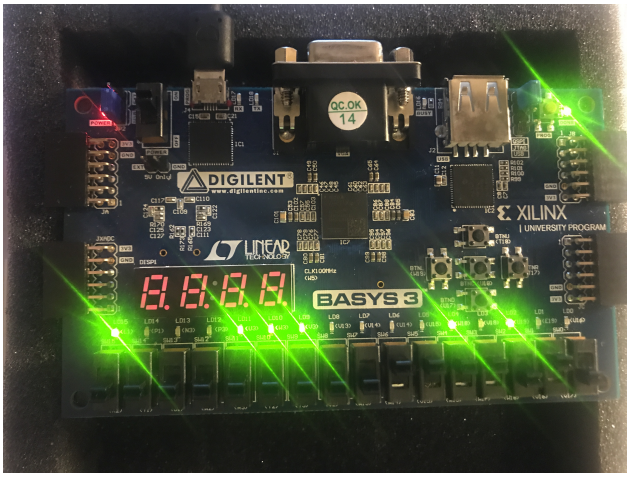

Figure 6: Basys board step4.

Code

Listing 1: register module source file

```
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 <= en ;
        else if ( en ==1)
            Q <= D ;
    end
endmodule
```

Listing 2: ALU module source file

```
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;
            // add the remaining commands
            default : out = in0 ;
        endcase
    end
endmodule
```

Listing 3: Top level source file

```
module top_lab9(
    input [11:0] sw,
```

```
    input clk, btnC, btnD, btnU,
    output reg [15:0] led
    );
    wire [7:0] R1;
    wire [7:0] A1;

    register #(. N (8) ) r1 (. D ( sw[7:0] ) , . clk ( clk ) , . en ( btnD
        ) , . rst ( btnC ) , . Q ( R1 ) ) ;

    alu #(.N(8)) a(.in1(R1), .in0(sw[7:0]), .op(sw[11:8]), .out(A1));

    register #(. N (8) ) r2 (. D ( A1 ) , . clk ( clk ) ,. en (btnU ) , .
       rst ( btnC ) , . Q ( led[15:8] ) ) ;

    assign led[7:0] = R1;

endmodule
```

Listing 4: register module testbench

```
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 5: ALU module testbench

```
module alu_test();
    reg[7:0] in0;
    reg[7:0] in1;
    reg[3:0] op;
    wire[7:0] out;
```

```
    alu #(.N(8)) a1(.in0(in0),.in1(in1),.op(op),.out(out));

    initial begin
        op = 0; in0 = 0; in1 = 1;#10;
        op = 1; in0 = 1; in1 = 0; #10;
        op = 2; in0 = 0; in1 = 0; #10;
        op = 3; in0 = 0; in1 = 1; #10;
        op = 4; in0 = 1; in1 = 1; #10;
        $finish;
        end

endmodule
```