ELC 2137 Lab 09: ALU with Input Register

Sebastian Lopez

April 2, 2020

Summary

In this lab I explain the difference between combinational and regular sequential logic. I also describe the operation of an SR latch, D latch, D flip-flop, and D register, as well as the differences in Verilog procedural blocks for combinational versus sequential logic. Lastly, I imported and modified modules from a previous project, and use them to design a modular system.

Expected results tables

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	A	A→6	6

Table 2: *alu* expected results table skeleton

Time (ns):	0-10	10-20	20-30	30-40	40-50	50-60
in0	1	3	5	7	9	b
in1	2	4	6	8	a	c
op	0	1	2	3	4	5
out	3	ff	4	0f	3	b

Code

Listing 1: Register Code

```
‘timescale 1ns / 1ps
// Sebastian Lopez  ELC 2137, 2020-04-02

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 // register
```

Listing 2: ALU Code

```
`timescale 1ns / 1ps
// Sebastian Lopez  ELC 2137, 2020-04-02

module ALU #( parameter N = 8)(

input [N - 1:0] in0 ,
input [N - 1:0] in1 ,
input [3:0] op,
output reg [N -1:0] out
) ;

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 // ALU
```

Listing 3: Top Lab9 Code

```
`timescale 1ns / 1ps
// Sebastian Lopez  ELC 2137, 2020-04-02

module top_lab9 #( parameter N = 8)(

input btnU, btnD, clk, btnC,
```

```
input [11:0] sw,
output [15:0] led
);

wire [7:0] reg1_out, ALU1_out, reg2_out;

register #(.N(8)) r1(
.D(sw [7:0]), .en(btnD), .clk(clk), .rst(btnC),
.Q(reg1_out));

assign led [7:0] = reg1_out;

ALU #(.N(8))  a1(
.in1(reg1_out), .in0(sw[7:0]), .op(sw[11:8]),
.out(ALU1_out));

register #(.N(8))  r2(
.D(ALU1_out), .en(btnU), .clk(clk), .rst(btnC),
.Q(reg2_out));

assign led [15:8] = reg2_out;


endmodule // Top Lab9
```

Listing 4: Register test Code

```
‘timescale 1ns / 1ps
// Sebastian Lopez  ELC 2137, 2020-04-02

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)) ;

always begin
clk = ~ clk ; #5;
end

initial begin
clk = 0; en = 0; rst = 0; D = 4’h0 ; #7;
rst = 1; #3;

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 // register_test
```

Listing 5: ALU Test Code

```
`timescale 1ns / 1ps
// Sebastian Lopez  ELC 2137, 2020-04-02

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 = 4;
#10
op = 2;
in0 = 5;
in1 = 6;
#10
op = 3;
in0 = 7;
in1 = 8;
#10
op = 4;
in0 = 9;
in1 = 10;
#10
op = 5;
in0 = 11;
in1 = 12;
#10
$finish;

end
endmodule // ALU_test
```

Results

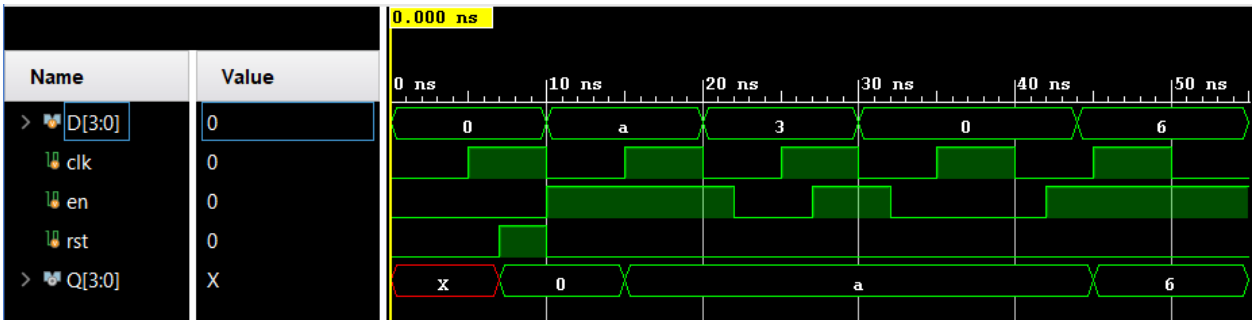

Figure 1: Register Waveform

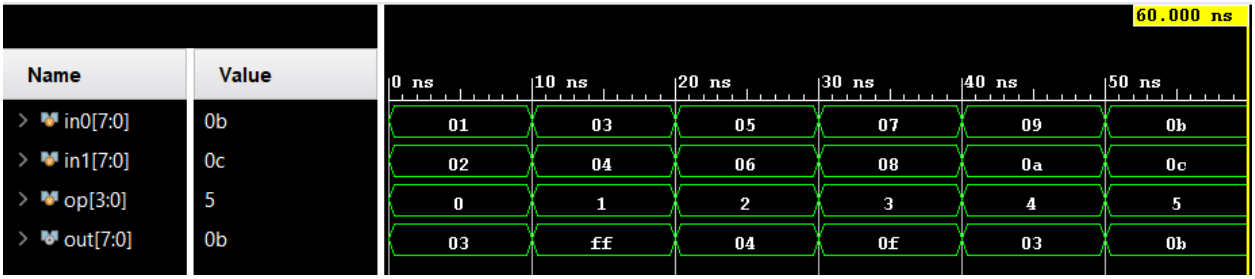

Figure 2: ALU Waveform

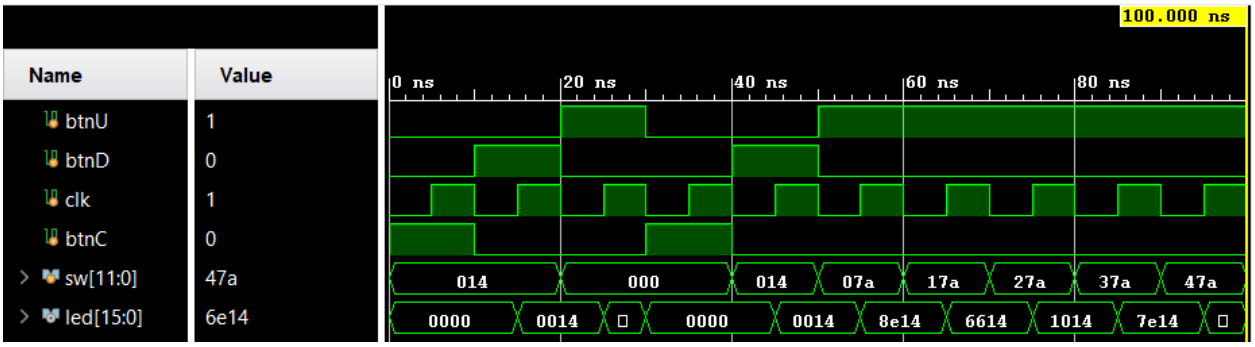

Figure 3: Basys 3 Lab 9 Behavior Waveform