ELC 2137 Lab 9: ALU

CJ Jones

April 1, 2020

Summary

This lab explores the skill of creating an Arithmetic Logic Unit , or ALU, using 2 numbers, one from a switch and one from using a register to store a number. By the end of this lab, one should be able to explain the difference between combinational and regular sequential logic, and describe the operation of an SR latch, D latch, D flip-flop, and D register. Using Verilog, some skills gained in this lab include: knowing the difference between procedural blocks vs sequential logic, and knowing how to import and modify modules from previous projects. Overall, this lab demonstrated how to utilize software and programmable logic to produce a hardware output.

Results

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

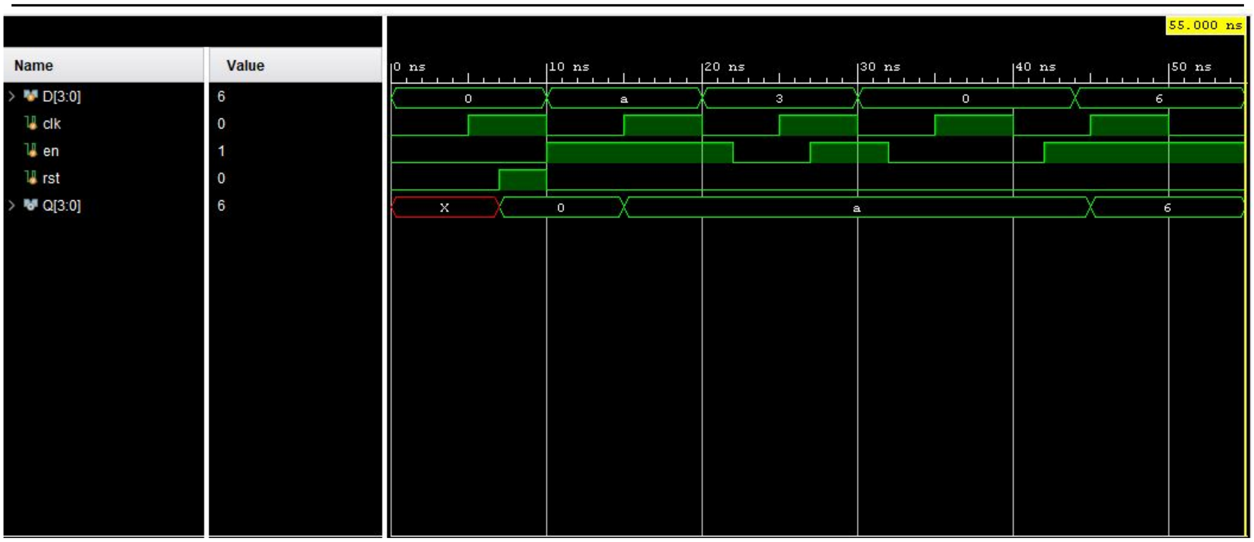

Figure 1: register expected results table and simulation
=

Time (ns):	0-10	10-20	20-30	30-40	40-50	50-60
in0	01	00	01	00	00	01
in1	01	00	00	01	00	01
op	0	1	2	3	4	5
out	02	00	00	01	00	01

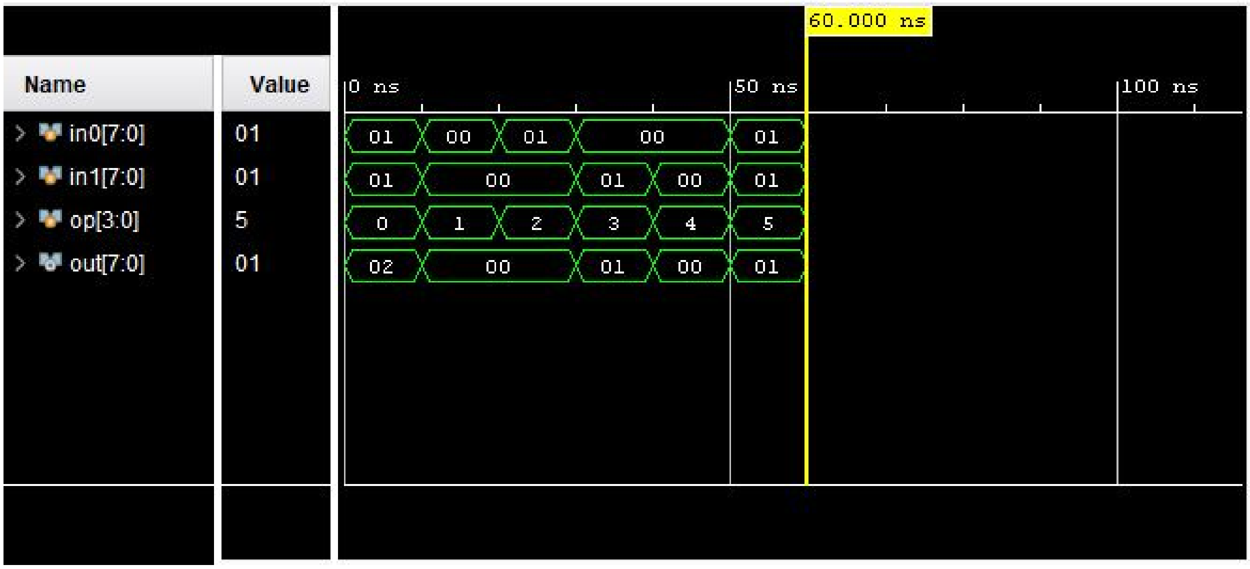

Figure 2: ALU expected results table and simulation

=

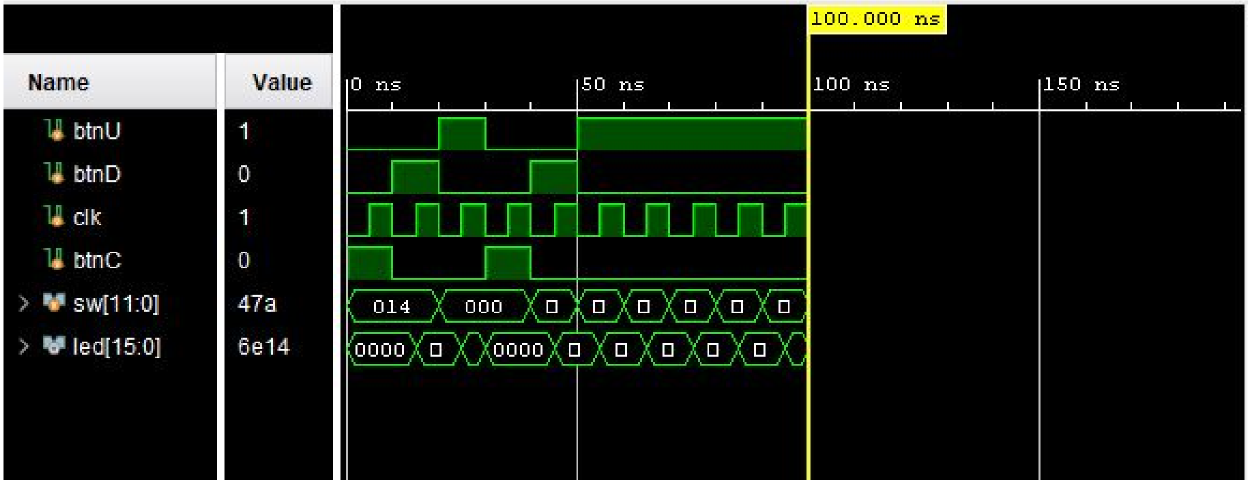

Figure 3: Top Level Simulation

=

Code

Listing 1: Register Source File

```
// Chris Jones , ELC 2137, 2020 -3-24

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
// Notes:
// - Reset is asynchronous , so this
// block needs to execute when rst
// goes high.
// - We want enable to be synchronous
// (i.e. only happens on rising
// edge of clk), so it is left out
// of "sensitivity" list.
endmodule
```

Listing 2: Register Test Bench Code

```
// Chris Jones , ELC 2137, 2020 -3-24
```

```
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 Source File

```
// Chris Jones , ELC 2137, 2020 -3-24

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 4: Alu Test Bench

```
// Chris Jones , ELC 2137, 2020 -3-24

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 = 1; #10;
op = 1; in0 = 0; in1 = 0; #10;
op = 2; in0 = 1; in1 = 0; #10;
op = 3; in0 = 0; in1 = 1; #10;
op = 4; in0 = 0; in1 = 0; #10;
op = 5; in0 = 1; in1 = 1; #10;
$finish;
end
endmodule
```

Listing 5: Top Level Source File

```
//Chris Jones , ELC 2137, 2020 -3-24

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
```