ELC 2137 Lab #9: ALU with Input Register

Kyle Monk

April 2, 2020

Summary

In this lab, an Arithmetic Logic Unit (ALU) was created that could do some simple calculations like addition and subtraction, as well as combinational logic like AND, OR, and XOR. One of the numbers being operated on was inputted through the switches in a basys3 board, while a register was used to store a second number.

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	6	6

Table 2: *alu* expected results table

Time (ns):	0-10	10-20	20-30	30-40	40-50	50-60
in0	14	14	14	14	14	14
in1	7A	7A	7A	7A	7A	7A
op	0	1	2	3	4	5
out	8E	9A	10	7E	6E	14

Results

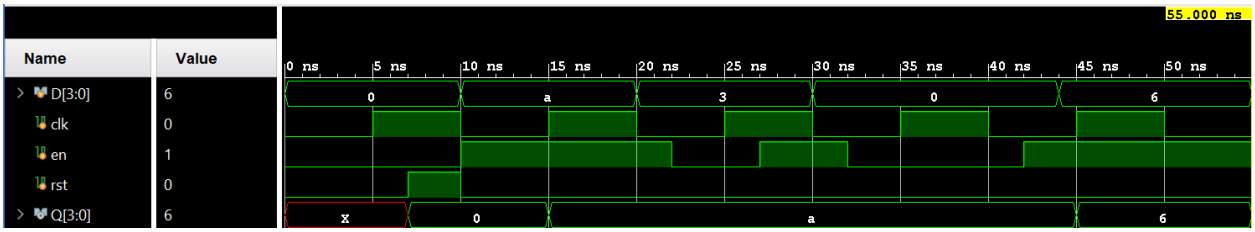

Figure 1: Register Simulation Waveform

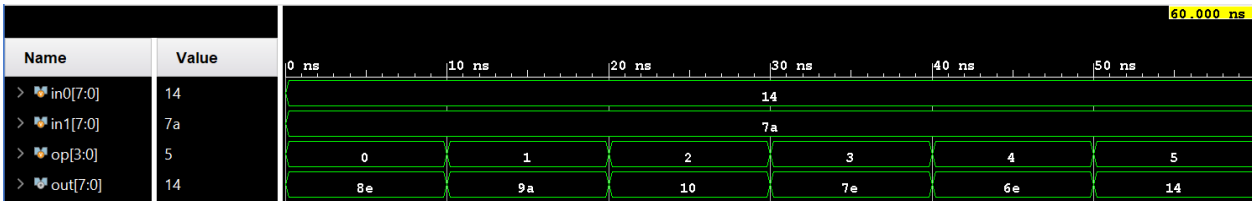

Figure 2: ALU Simulation Waveform

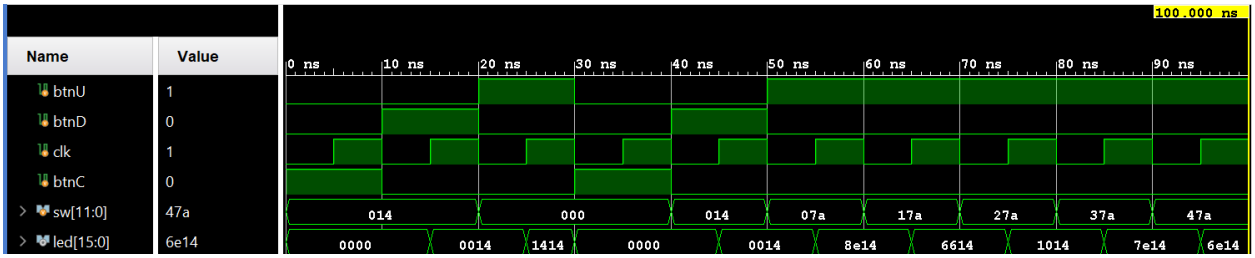

Figure 3: Top Level Simulation Waveform

Code

Listing 1: register Design

```
‘timescale 1ns / 1ps
//Lab 9, Created by Kyle Monk 04-01-20

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 Design

```
‘timescale 1ns / 1ps
//Lab 9, created by Kyle Monk 04-01-20

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 Design

```
`timescale 1ns / 1ps
//Lab 9, created by Kyle Monk 04-01-20

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 Design

```
`timescale 1ns / 1ps
//Lab 9, created by Kyle Monk, 04-01-20

module alu_test();

reg [7:0] in0, 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 = 8'h14; in1 = 8'h7A; #10;
op = 1; in0 = 8'h14; in1 = 8'h7A; #10;
op = 2; in0 = 8'h14; in1 = 8'h7A; #10;
op = 3; in0 = 8'h14; in1 = 8'h7A; #10;
op = 4; in0 = 8'h14; in1 = 8'h7A; #10;
op = 5; in0 = 8'h14; in1 = 8'h7A; #10;
$finish;
end
endmodule
```

Listing 5: top_lab9 Design

```
`timescale 1ns / 1ps
//Lab 9, created by Kyle Monk 04-01-20

module top_lab9(
input btnU, btnD, clk, btnC, input [15:0] sw,
output [15:0] led);

wire [7:0] r1out, aluout, r2out;

register #(.N(8)) r1(.clk(clk), .D(sw[7:0]), .en(btnD), .rst(btnC), .Q(
   r1out));
alu #(.N(8)) a1(.in0(sw[7:0]), .in1(r1out), .op(sw[11:8]), .out(aluout));
register #(.N(8)) r2(.clk(clk), .D(aluout), .en(btnU), .rst(btnC), .Q(
   r2out));
assign led = {r2out, r1out};

endmodule
```