ELC 2137 Lab 9: ALU

Tyler Haygood

April 2, 2020

Summary

The Purpose of this lab was to familiarize the class with the basics of Registers, and ALU modules. Using Vivado, and the basys3 board modules were coded for both an ALU, a Register, and a top level file to connect the modules together. The result of the connections was an ALU module that can do a few mathematical operations with the help of registers for data storage.

Expected results tables

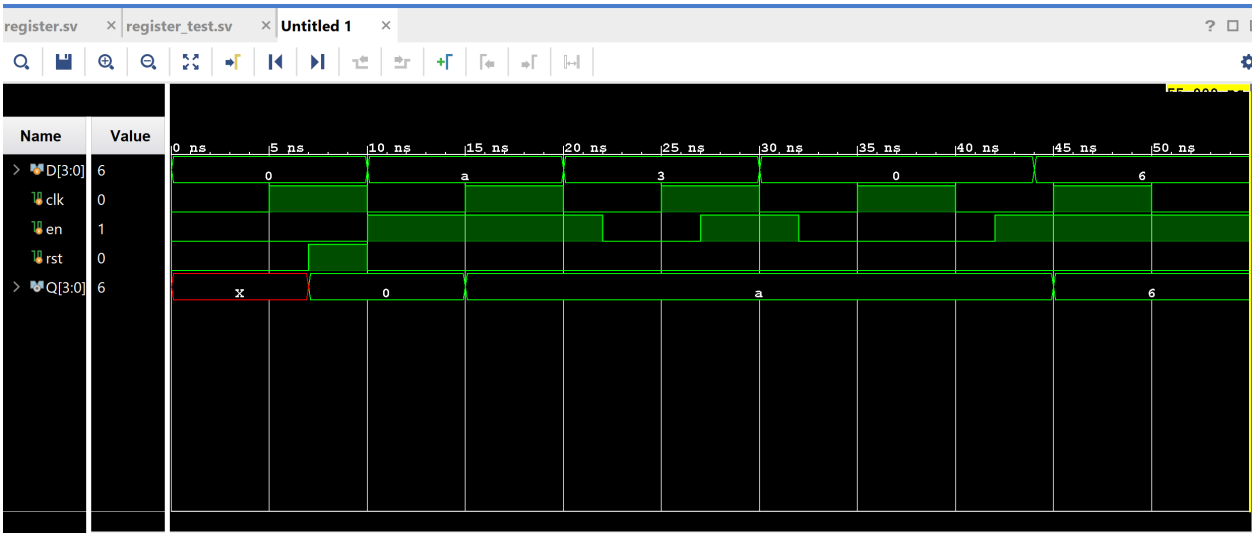
Table 1: *register* expected results table

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→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
in0	5	5	1	1	1
in1	5	5	0	0	0
op	0	1	2	3	4
out	0A	0	0	1	1

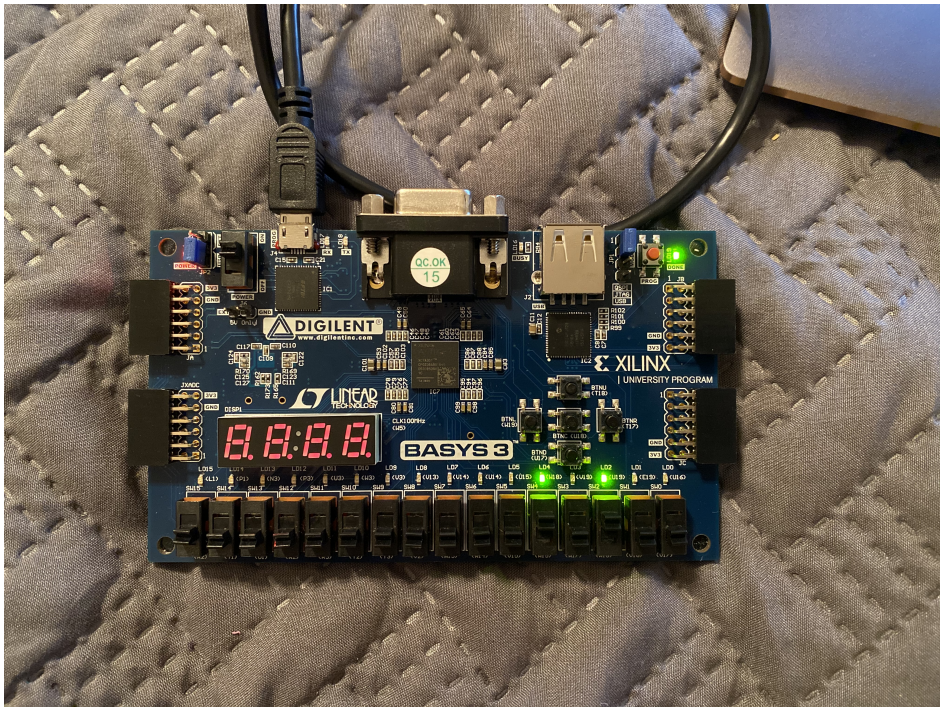
Results

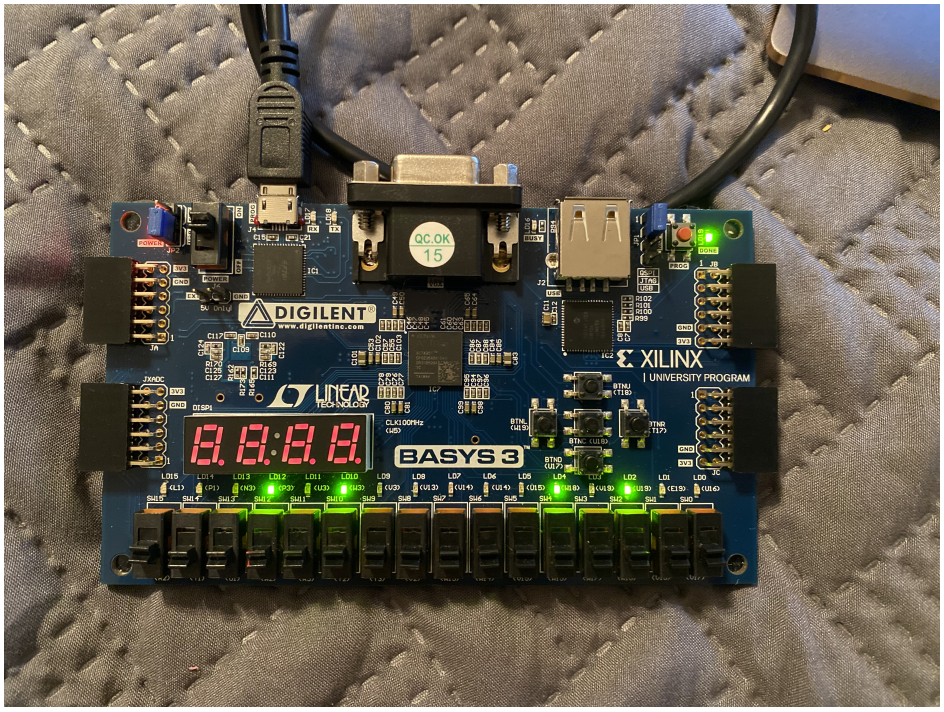


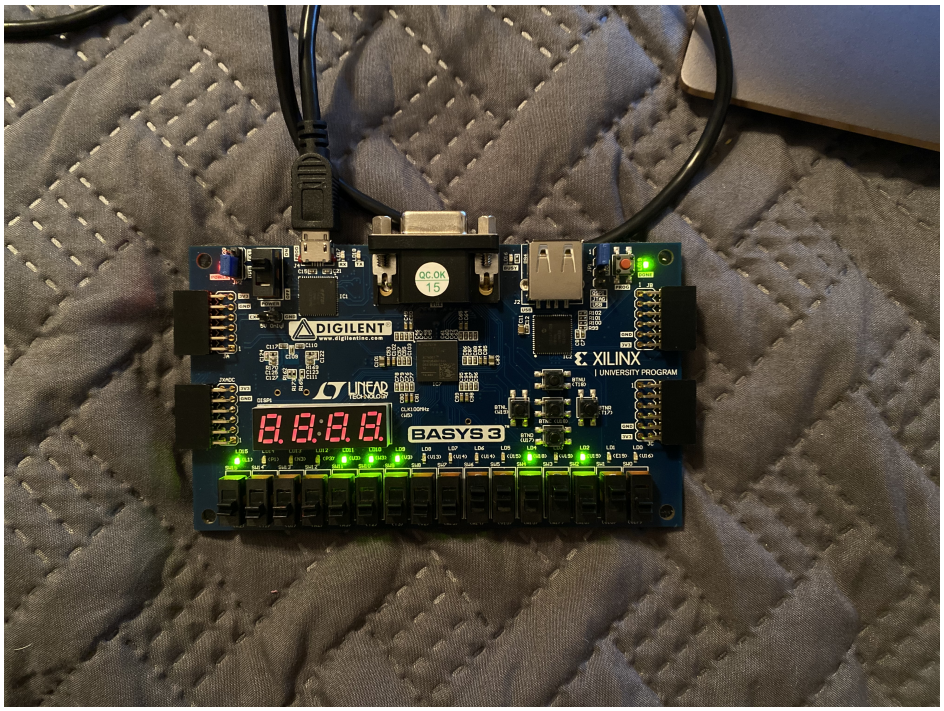
Register Module Simulation

ALU Module Simulation

Board Operation







Board Testing

Code

```
`timescale 1ns / 1ps

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
// 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

```

```
`timescale 1ns / 1ps

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

```

Register, and Register Test-bench

```
`timescale 1ns / 1ps

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
```

```
`timescale 1ns / 1ps

module alu_test();

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

alu #(.N(8)) alu0(.in0(in0), .in1(in1), .op(op), .out(out));

initial begin

    op = 4'b0000; in0 = 8'b00000101; in1 = 8'b00000101; #10;
    op = 4'b0001; in0 = 8'b00000101; in1 = 8'b00000101; #10;
    op = 4'b0010; in0 = 8'b00000001; in1 = 8'b00000000; #10;
    op = 4'b0011; in0 = 8'b00000001; in1 = 8'b00000000; #10;
    op = 4'b0100; in0 = 8'b00000001; in1 = 8'b00000000; #10;

$finish;
end

endmodule
```

ALU, ALU Test-bench

```
`timescale 1ns / 1ps

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

wire [7:0] r1Out, alu1Out, r2Out;
wire [3:0] AluOp;
wire [7:0] Din0;

assign Din0 = sw[7:0];
assign AluOp = sw[11:8];

register #(.N(8)) r1 (.D(Din0) , .clk(clk) , .en(btnD) , .rst(btnC) , .Q(r1Out));

alu #(.N(8)) alu1 (.in0(Din0), .in1(r1Out), .op(AluOp), .out(alu1Out));

register #(.N(8)) r2 (.D(alu1Out) , .clk(clk) , .en(btnU) , .rst(btnC) , .Q(r2Out));

assign led [7:0] = r1Out;

assign led [15:8] = r2Out;

endmodule

```

Top Module