ELC 2137 Lab 9: ALU with input register

Alexander Noll

April 2, 2020

Summary

In the lab, two D register memory units and an Arithmetic Logic Unit (ALU) were created. Using a mixture of sequential and combinational logic in Verilog, the Basys3 board was programed to add, subtract, and compare two numbers. The comparisons were done bitwise using AND, OR, and XOR gates.

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	A	6	6

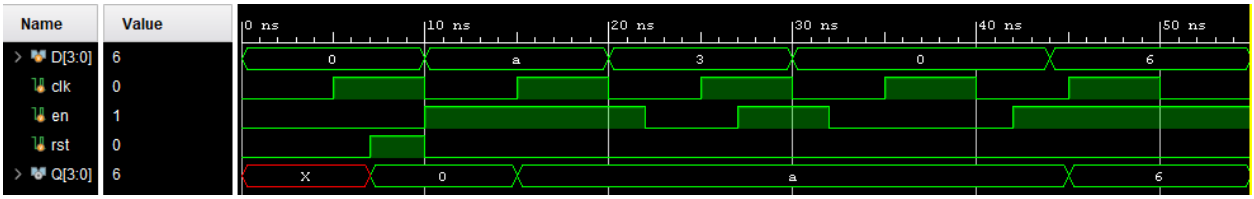

Figure 1: Register Test Waveform

Table 2: *alu* expected results table skeleton

Time (ns):	0-5	5-10	10-15	15-20	20-25
in0	0C	0C	0C	0C	0C
in1	07	07	07	07	07
op	0	1	2	3	4
out	13	05	04	0F	0D

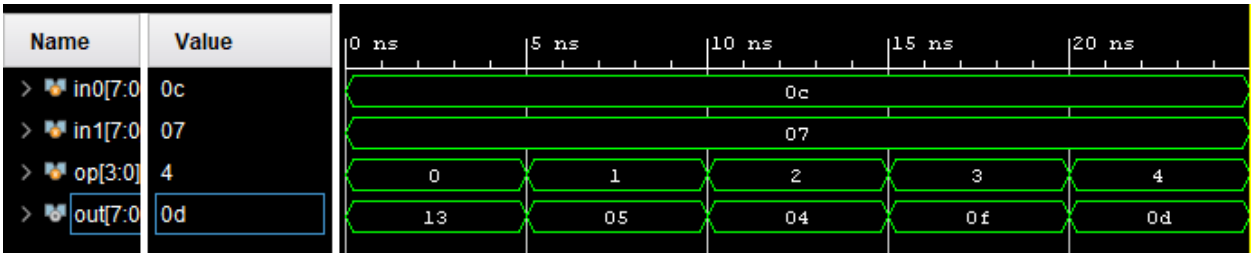

Figure 2: ALU Test Waveform

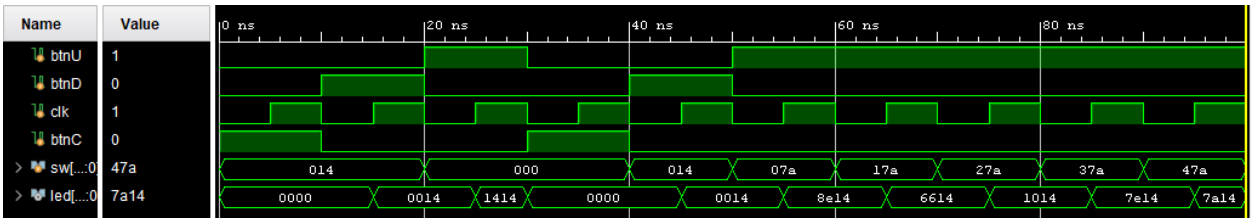

Figure 3: Top Lab 9 Test Waveform

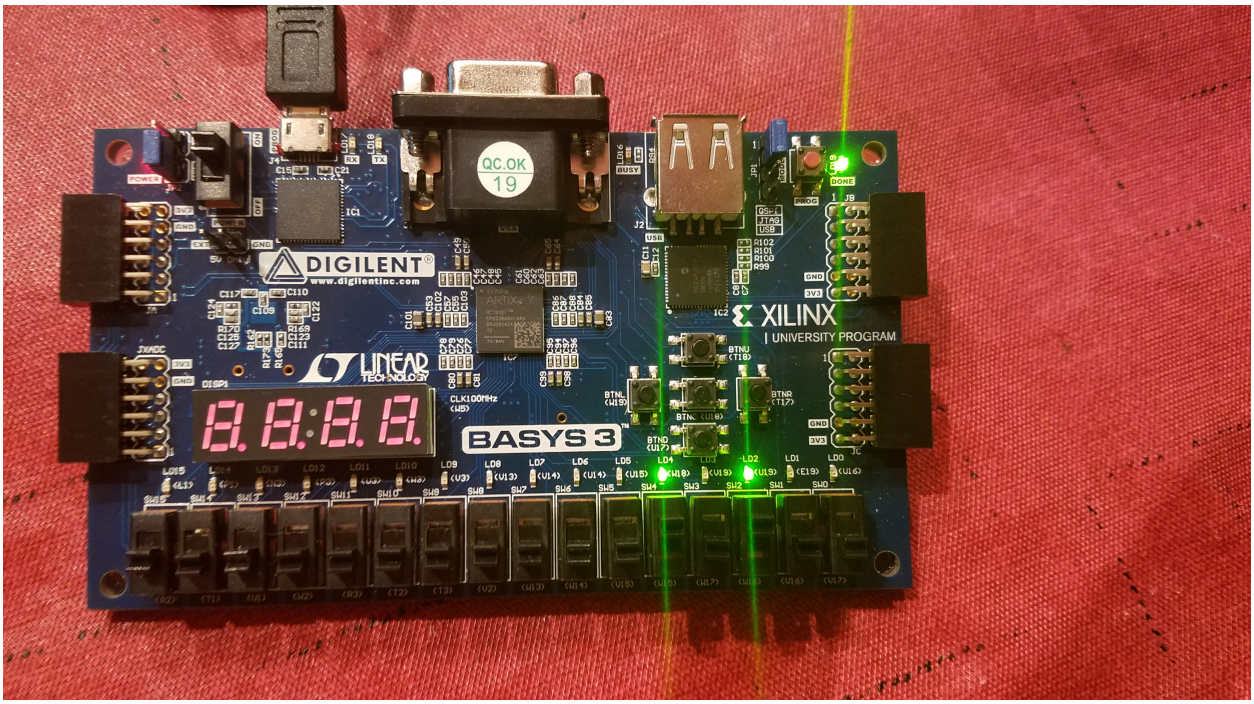

Figure 4: Board Operation Step 1

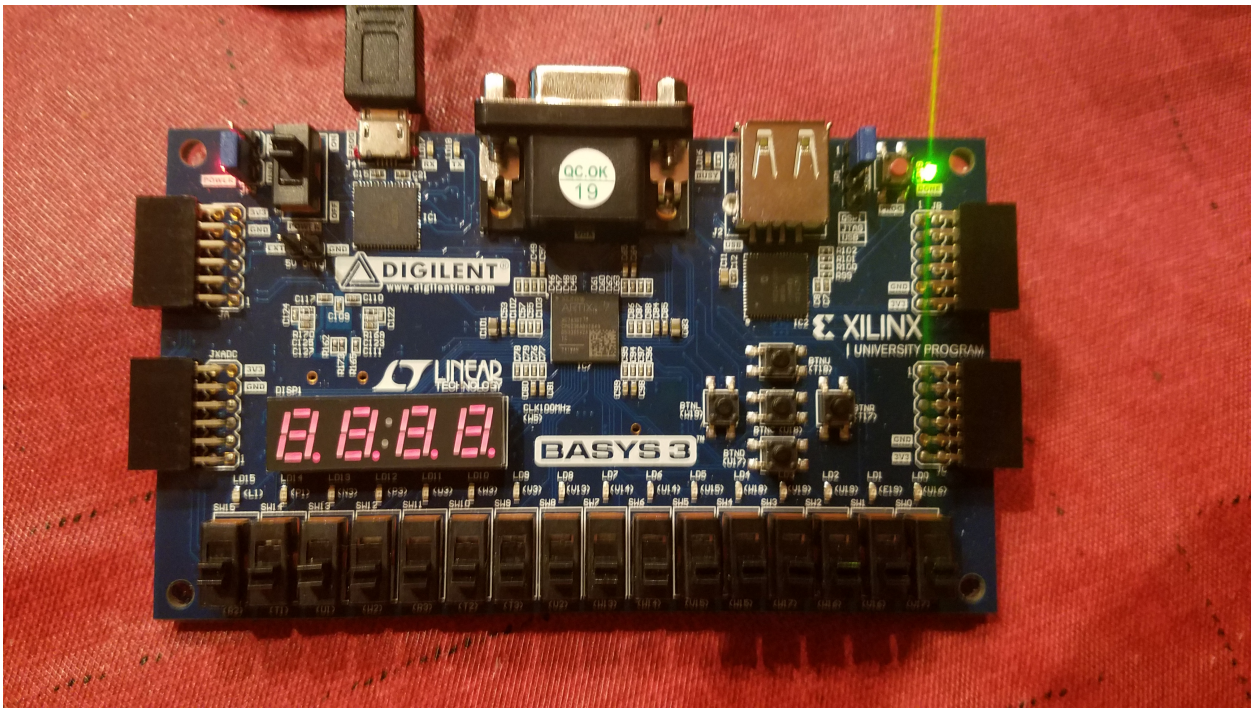

Figure 5: Board Operation Step 2

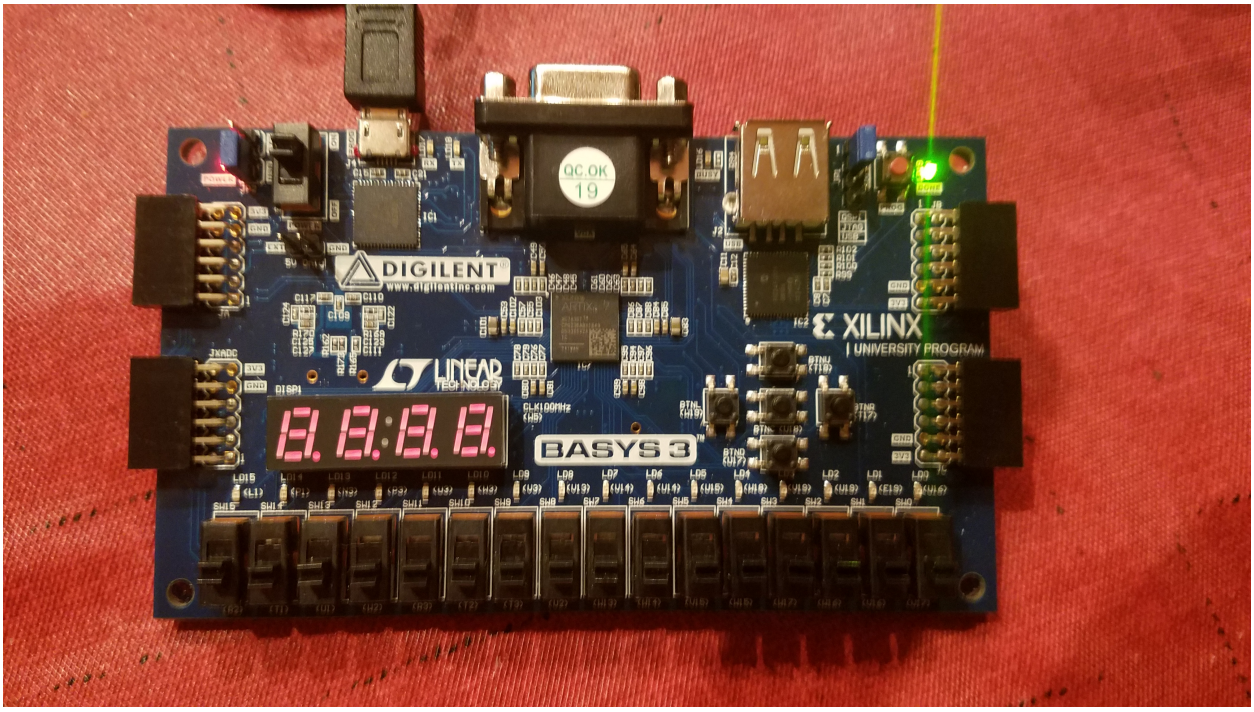

Figure 6: Board Operation Step 3

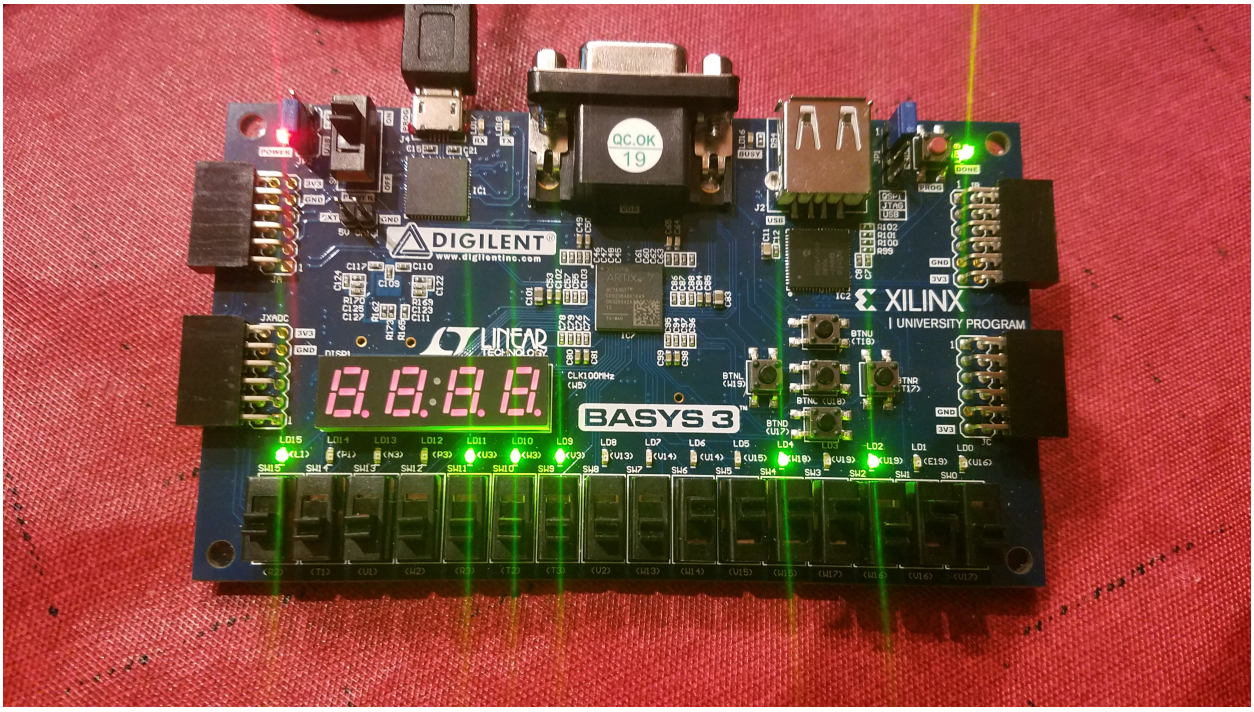

Figure 7: Board Operation Step 4- Operation 0(Add)

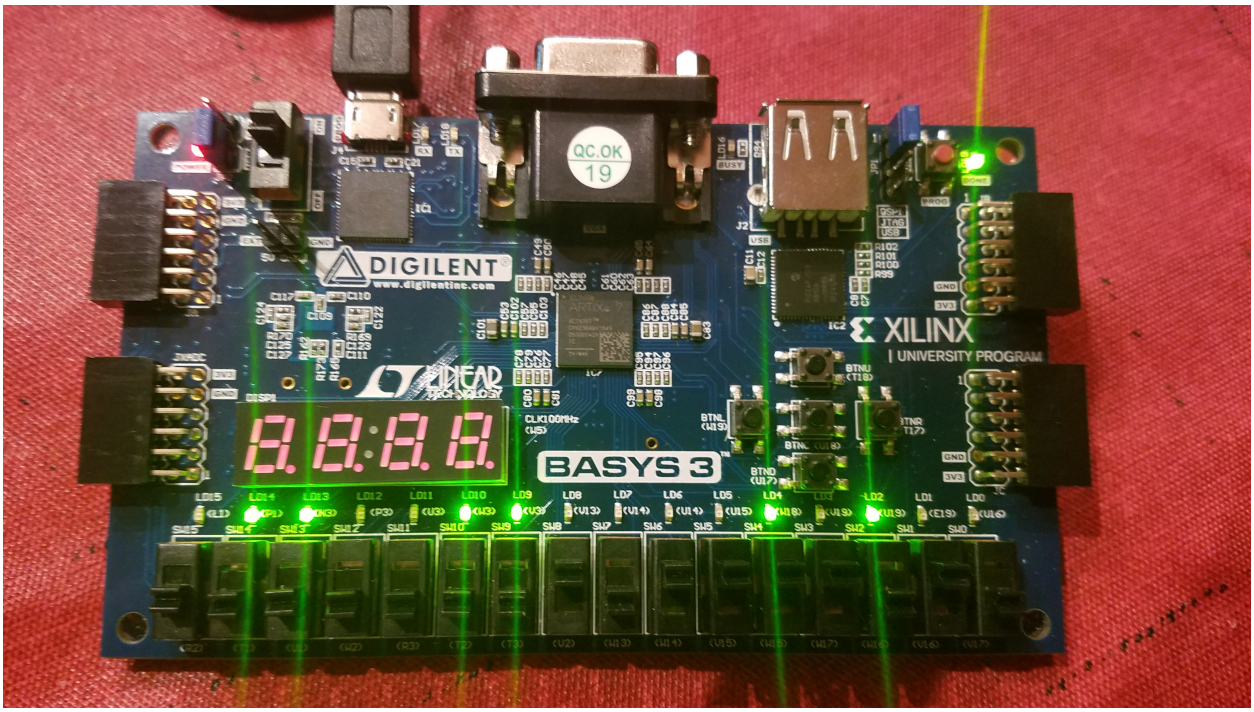

Figure 8: Board Operation Step 5 -Operation 1(Subtract)

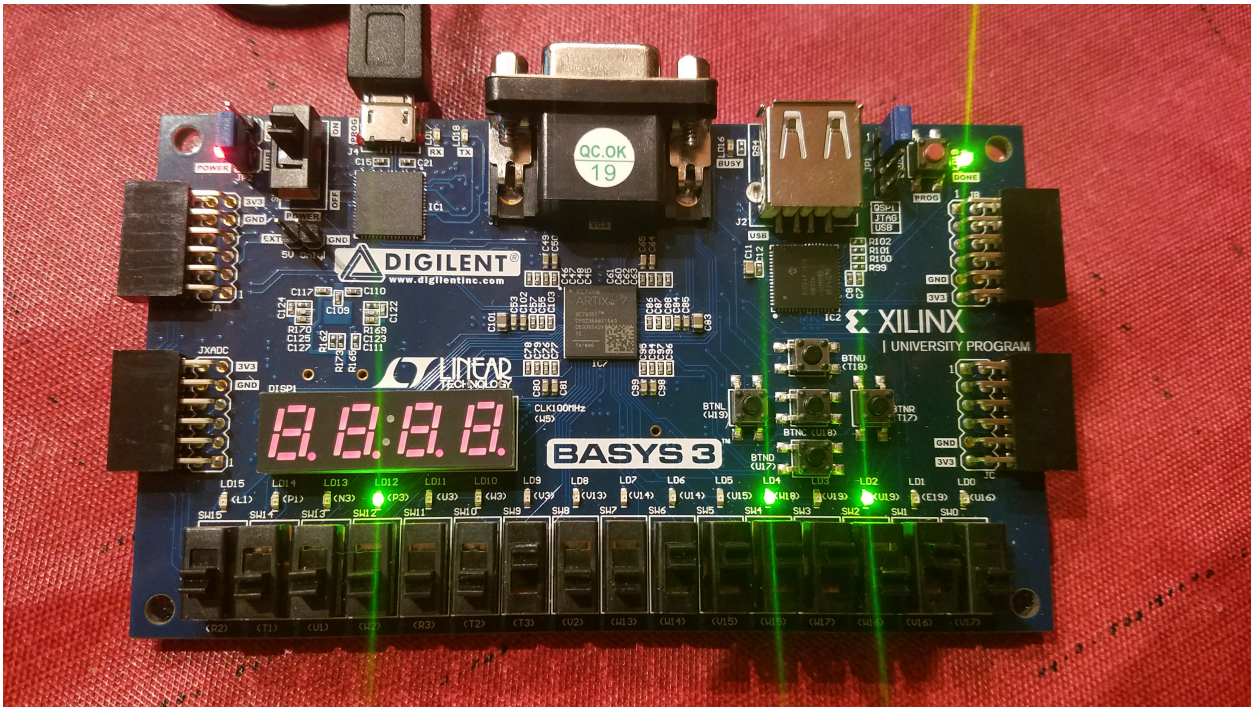

Figure 9: Board Operation Step 6 -Operation 2(AND)

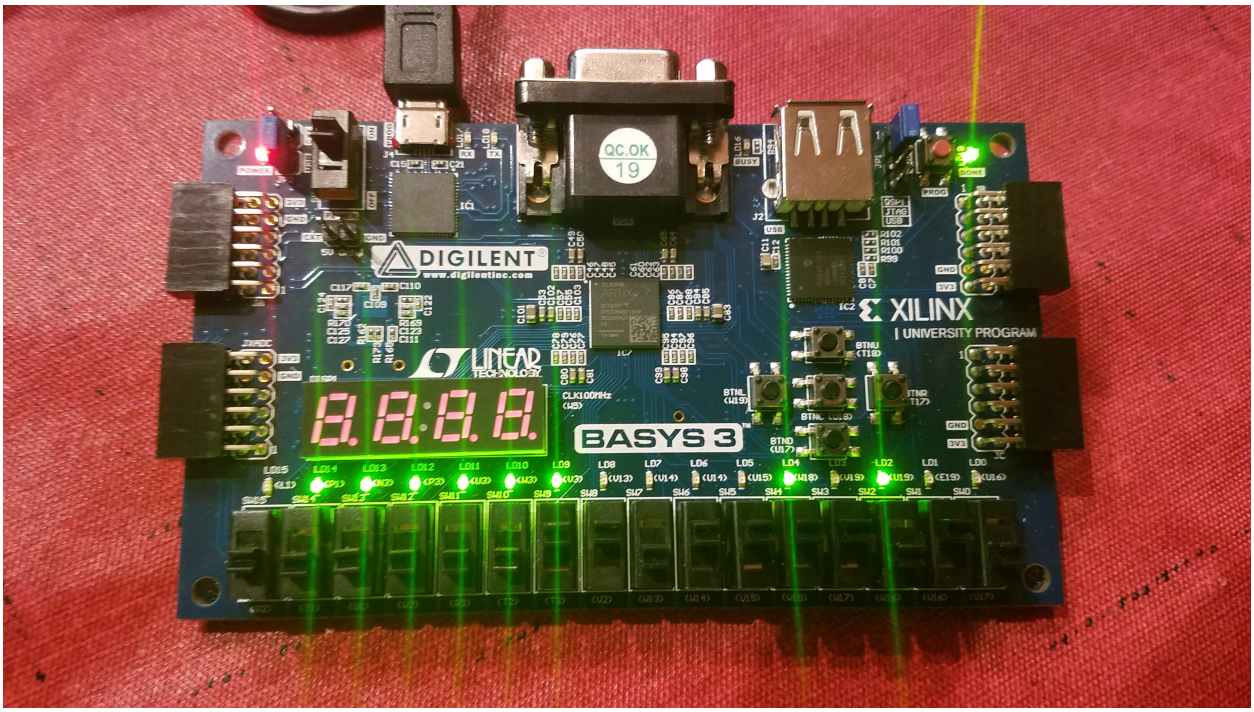

Figure 10: Board Operation Step 7 -Operation 3(OR)

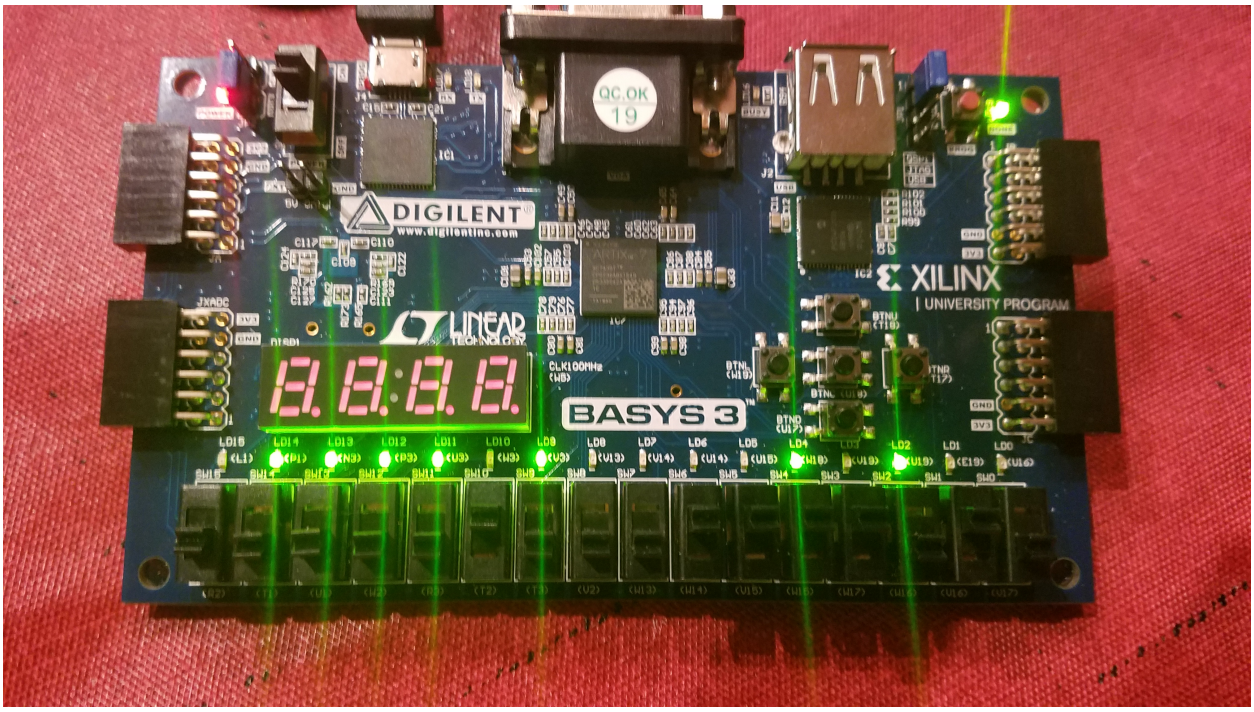

Figure 11: Board Operation Step 8 -Operation 4(XOR)

Code

Listing 1: Verilog Register Source

```
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)
        Q <= 0;
    else
    begin
        if (en)
        Q <= D;
        else
        begin
            Q <= Q;
        end
    end
end
endmodule
```

Listing 2: Verilog ALU Source

```
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
);

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 3: Verilog Top Lab 9 Source

```
module top_lab9(
    input [11:0] sw,
    input btnU, btnD, btnC, reg clk,
    output [15:0] led
    );
    wire [7:0] ALUout;

register #(.N(8)) REG1(.D(sw[7:0]),.en(btnD),.clk(clk),.rst(btnC),.Q(led
   [7:0]));

alu #(.N(8)) ALU(.in0(sw[7:0]),.in1(led[7:0]),.op(sw[11:8]),.out(ALUout));

register #(.N(8)) REG2(.D(ALUout),.en(btnU),.clk(clk),.rst(btnC),.Q(led
   [15:8]));
endmodule
```

Listing 4: Verilog register Test Bench

```
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
```

Listing 5: Verilog ALU Test Bench

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

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

initial
    begin
    in0 = 12;
    in1 = 7;
    op=0;   #5;
    op=1;   #5;
    op=2;   #5;
    op=3;   #5;
    op=4;   #5;

    $finish;
    end
endmodule
```