In LAB: implement ripple-carry adder in Verilog and program it on the FPGA board

Step 1. Half Adder (code):

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:       19:24:22 04/24/2018
// Design Name:
// Module Name:       HalfAdder
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module HalfAdder(
      input a,
      input b,
```

```
    output sum,
    output cout
    );

        xor(sum, a, b);
        and(cout, a, b);

endmodule
```

Step 2. Full Adder (code):

Full Adder

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:      19:25:24 04/24/2018
// Design Name:
// Module Name:      FullAdder
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
```

```
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module FullAdder(
    input a,
    input b,
    input cin,
    output sum,
    output cout
    );

        wire c1, c2, s;

        HalfAdder T1(.a(a), .b(b), .sum(s), .cout(c1));
        HalfAdder T2(.a(cin), .b(s), .sum(sum), .cout(c2));
        or(cout, c1, c2);
endmodule
```

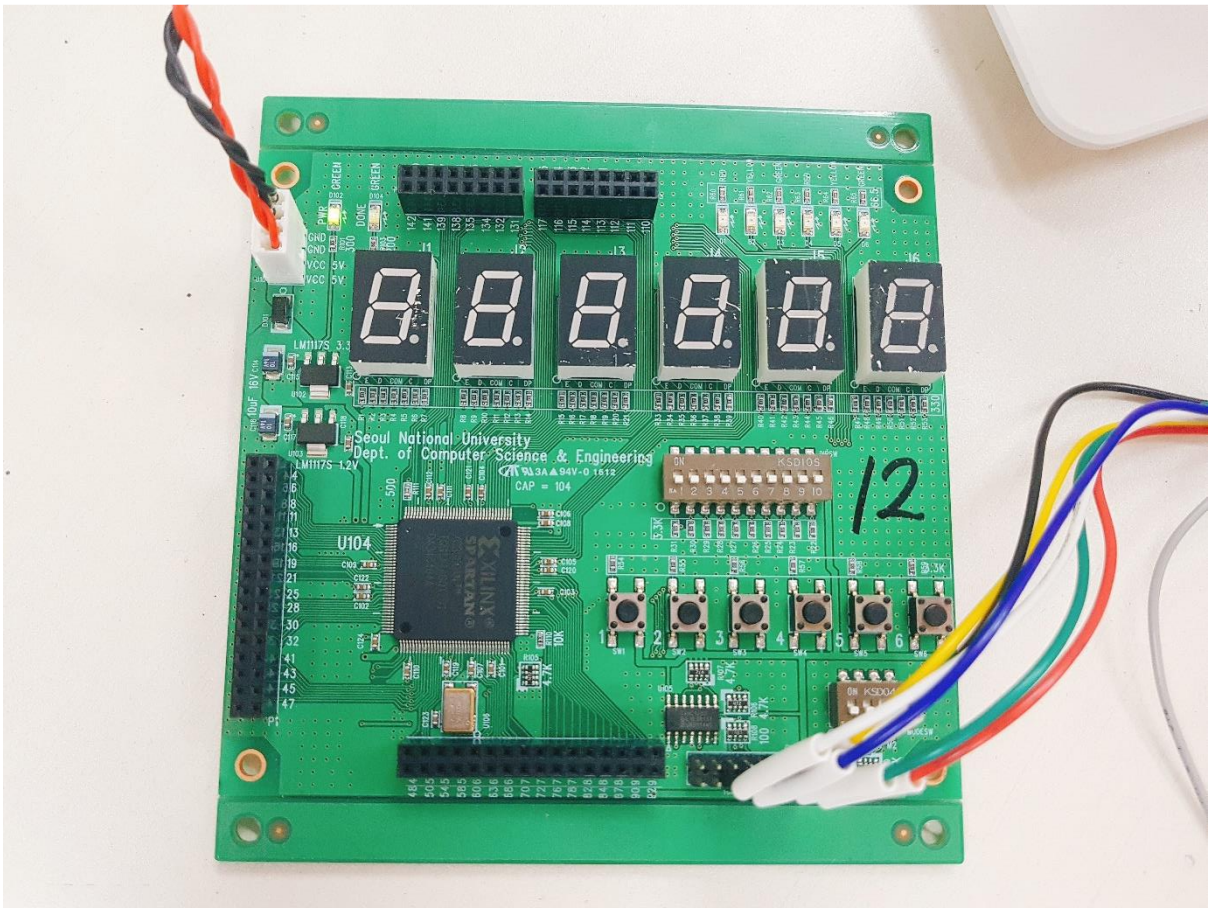
Step 3. Ripple-Carry Adder (code):

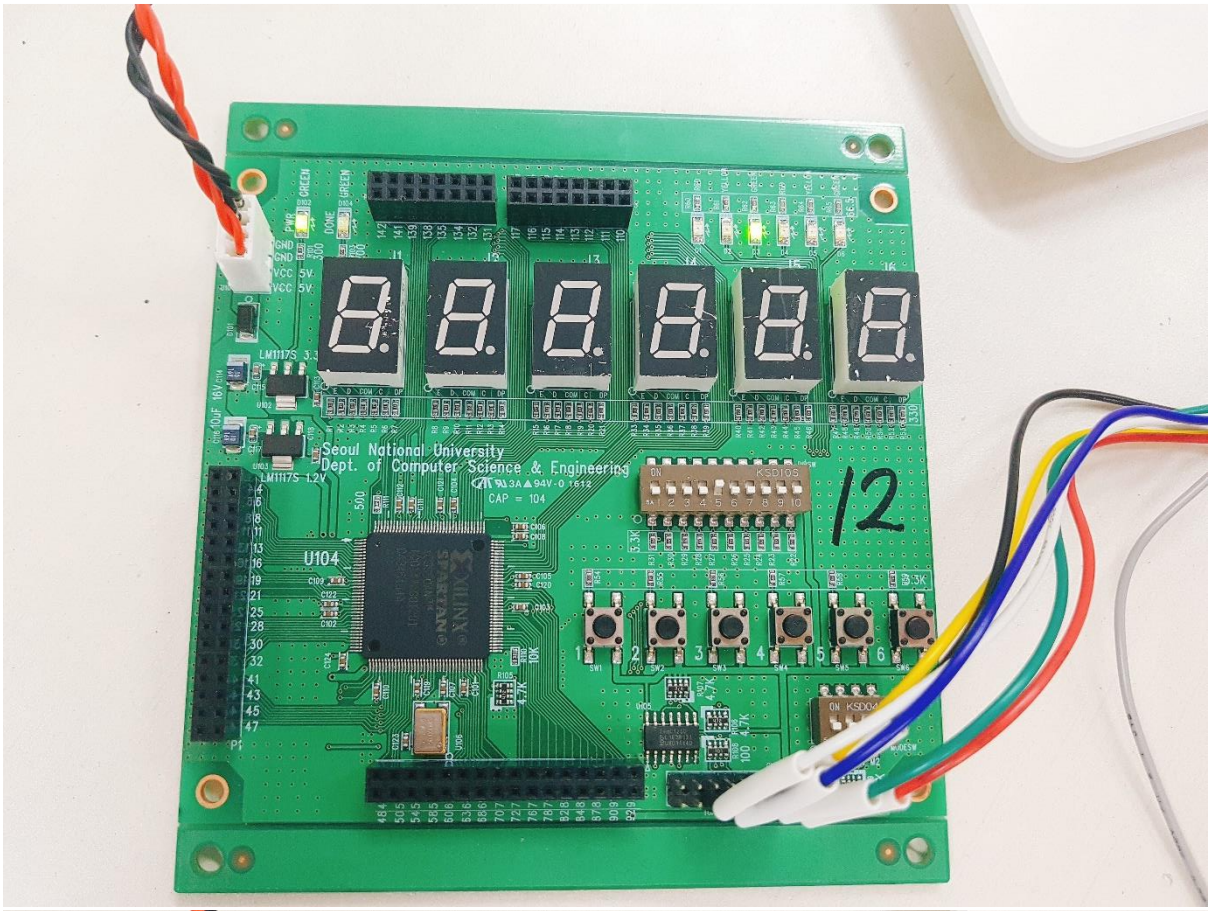
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
```

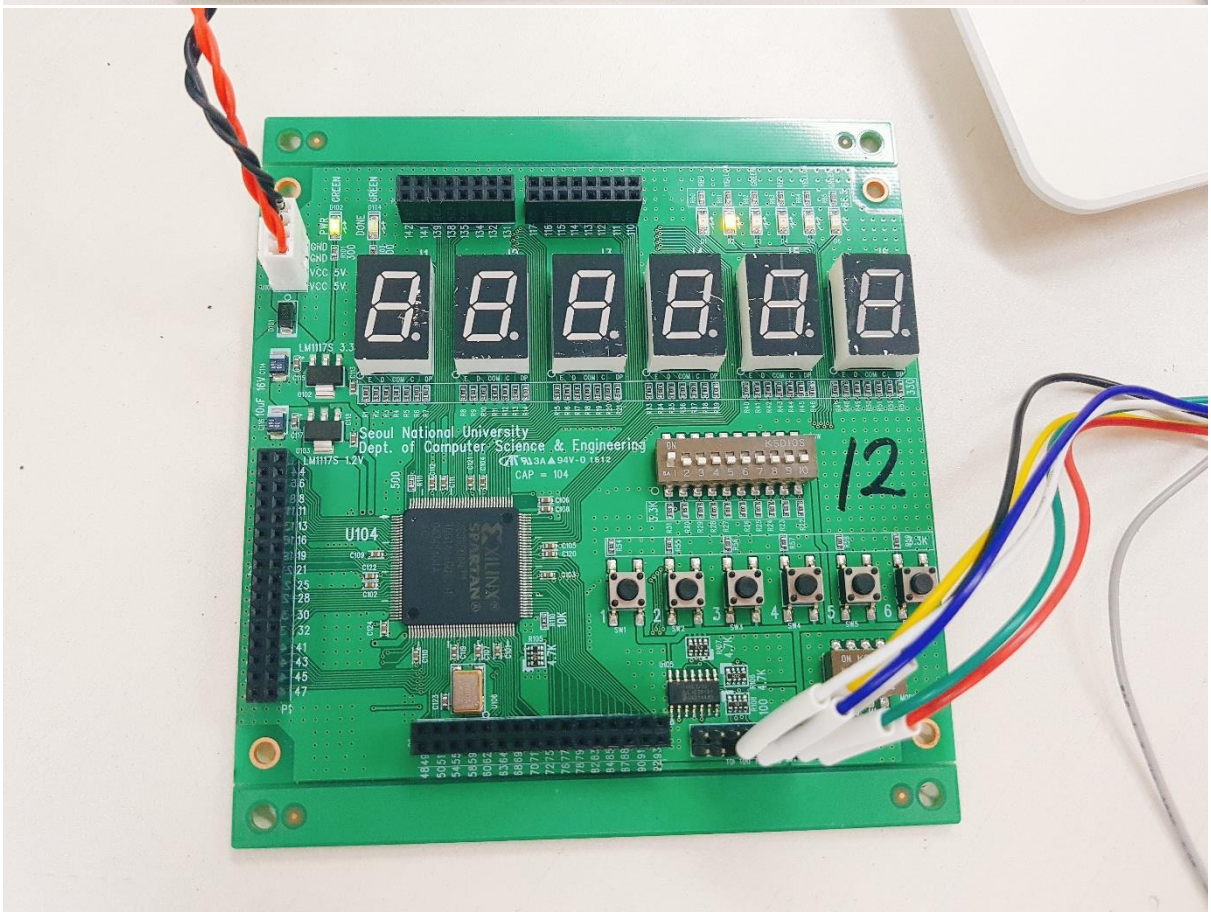
```
// Engineer:
//
// Create Date:       19:30:26 04/24/2018
// Design Name:
// Module Name:       RippleCarryAdder
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module RippleCarryAdder(
      input [1:0] a,
      input [1:0] b,
      input c0,
      output [1:0] s,
      output c2
      );

            wire c1;
```

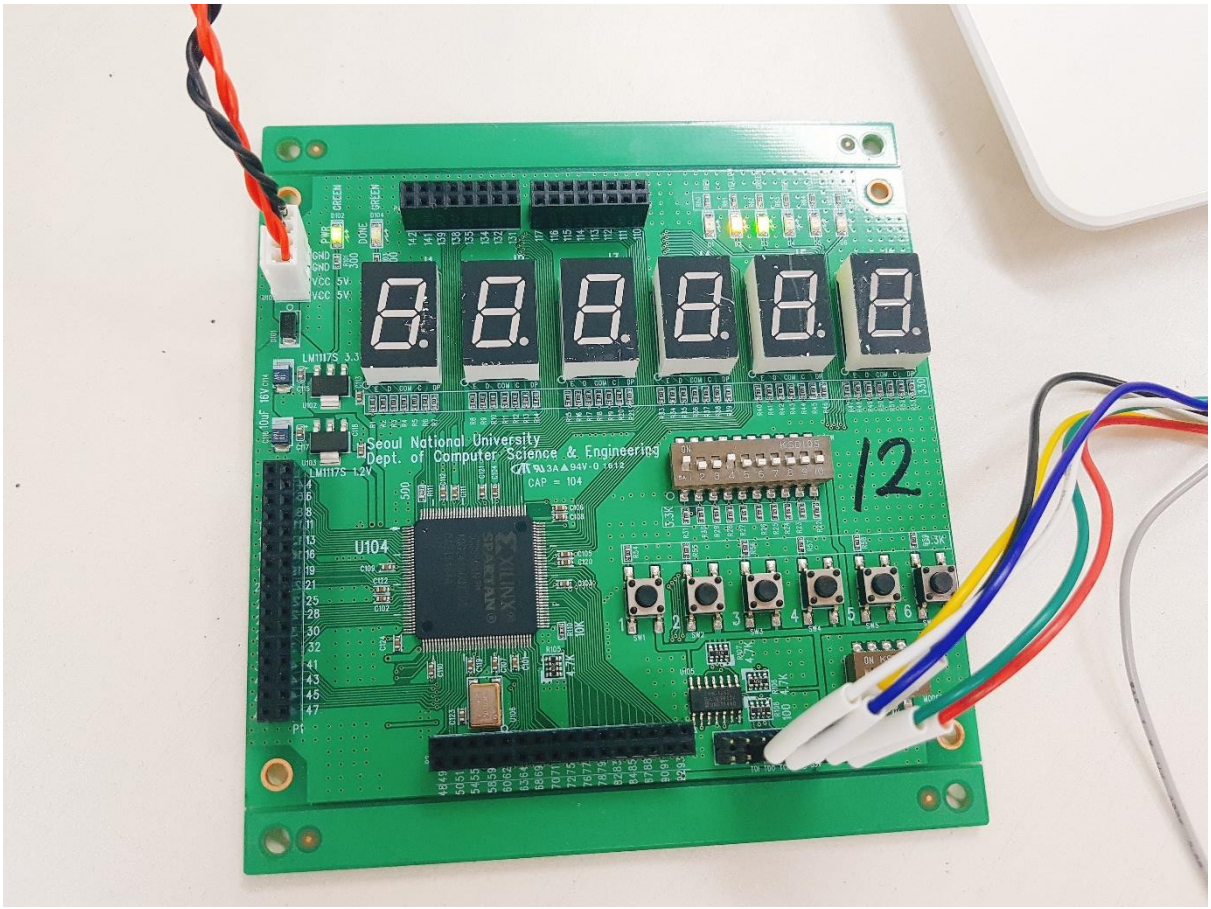
```
    FullAdder T1(.a(a[0]), .b(b[0]), .cin(c0), .sum(s[0]), .cout(c1));

    FullAdder T2(.a(a[1]), .b(b[1]), .cin(c1), .sum(s[1]), .cout(c2));


endmodule
```

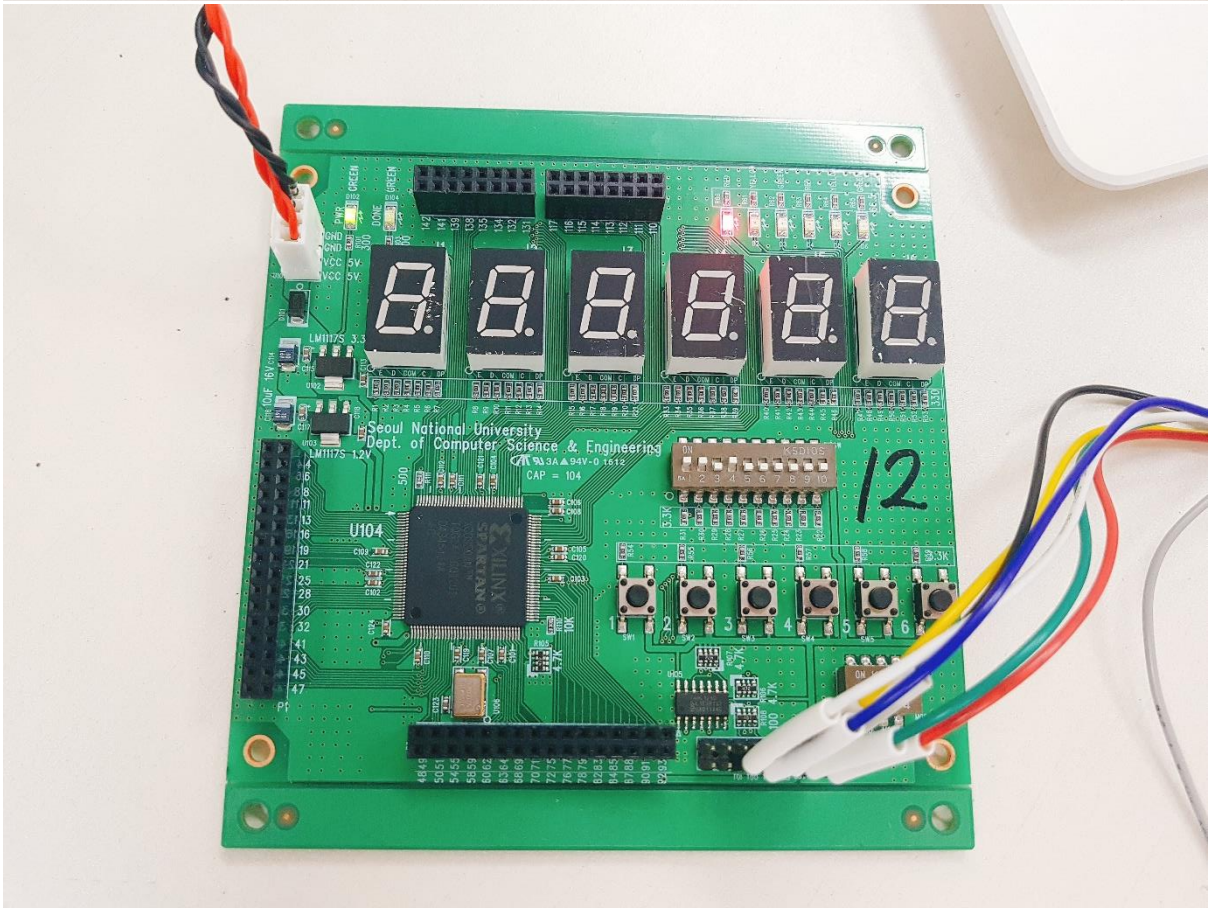
Result:

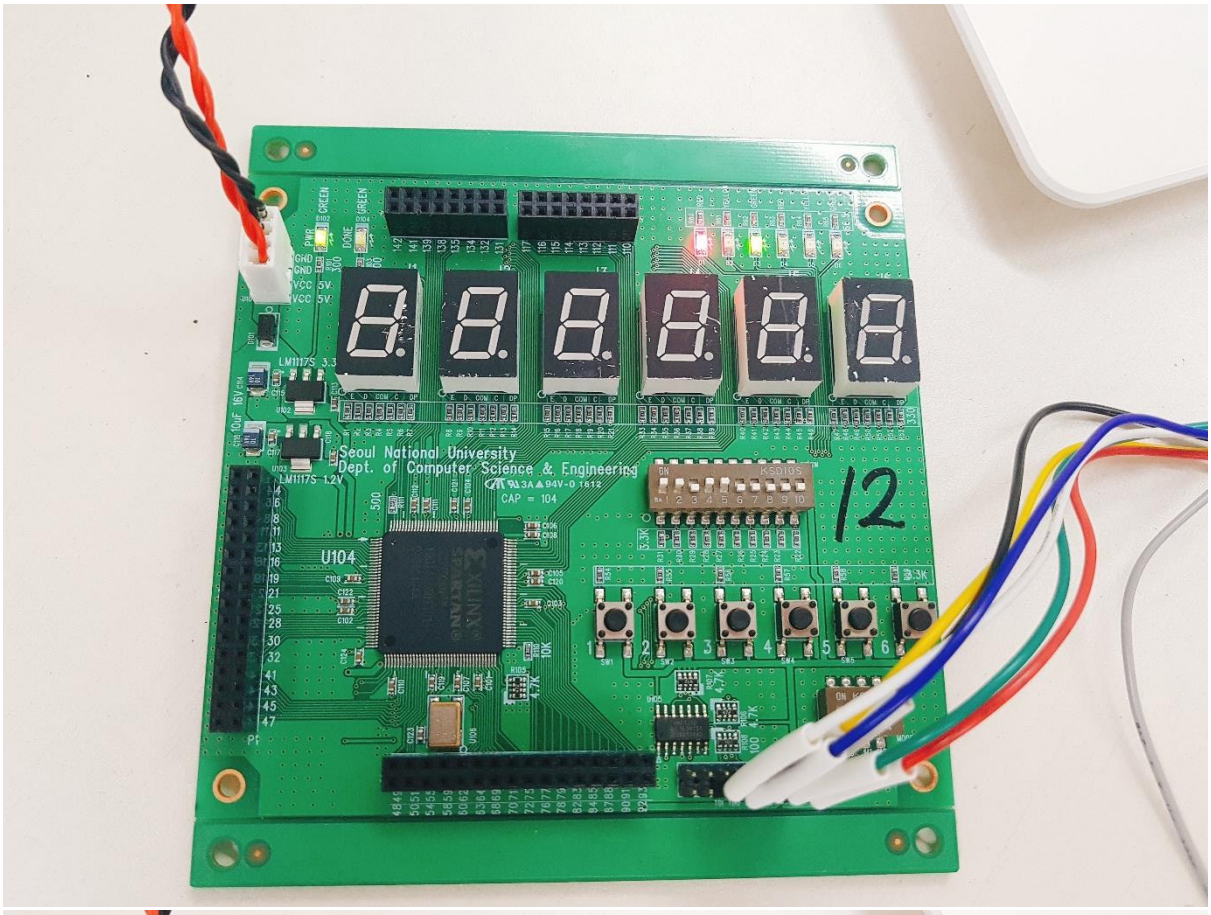


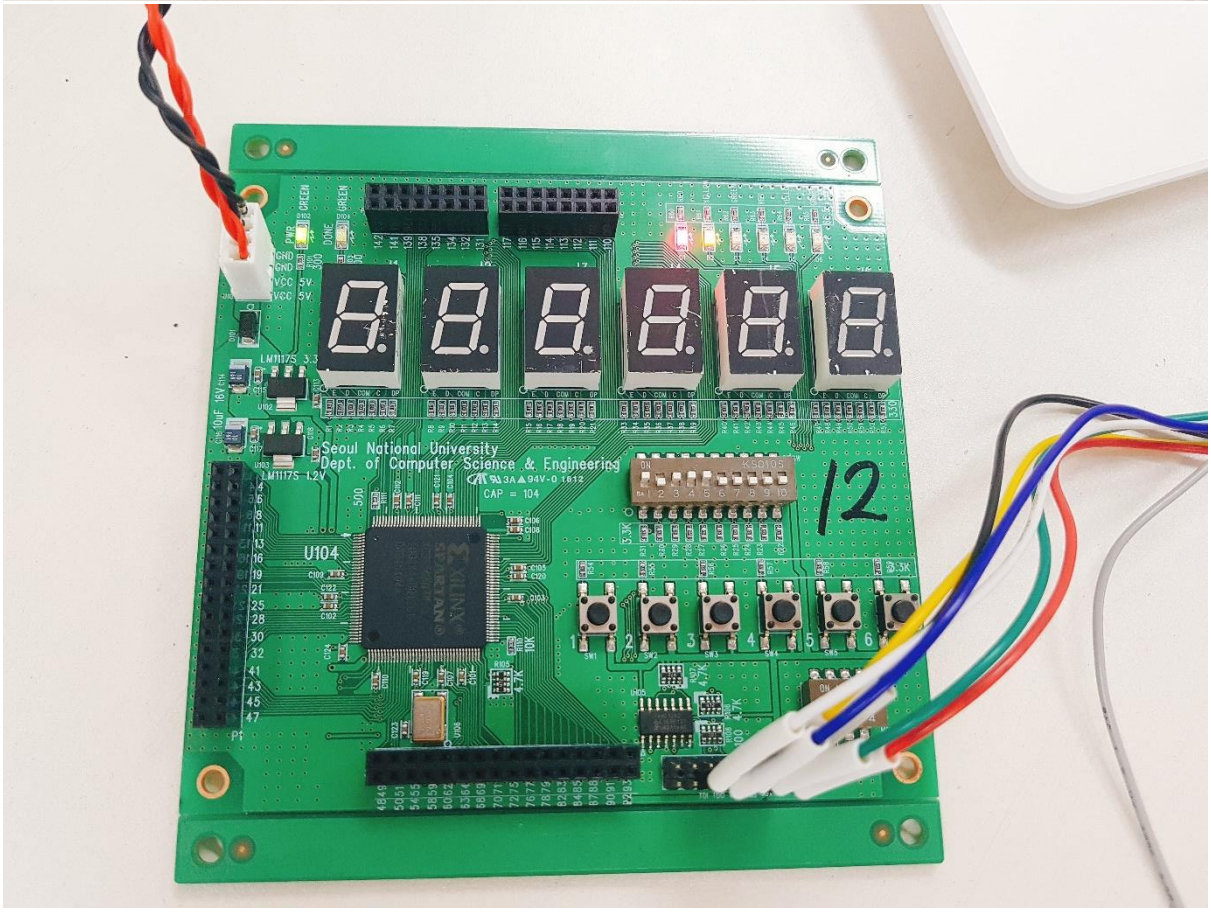

Seoul National University
Dept. of Computer Science & Engineering
U104
12

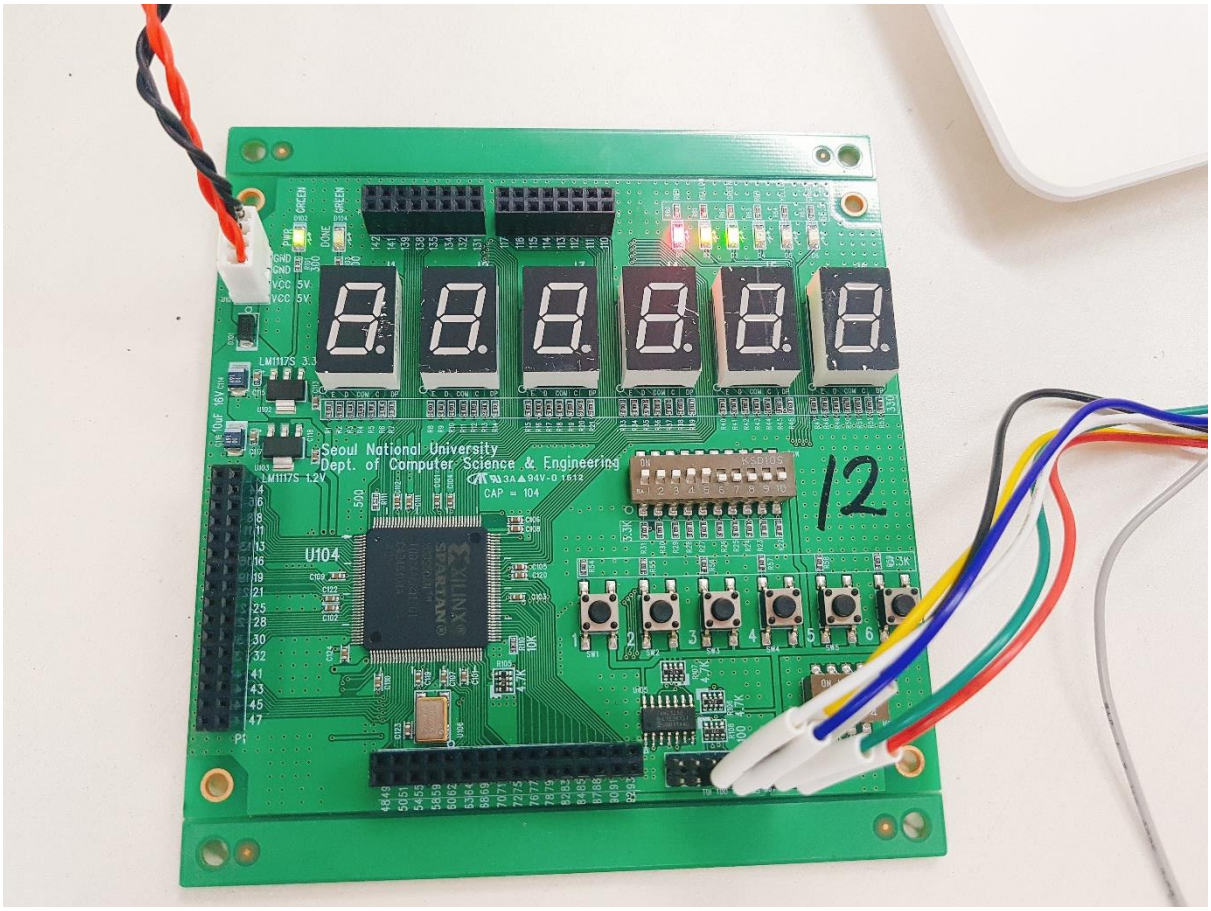

Seoul National University
Dept. of Computer Science & Engineering
U104
12


Seoul National University
Dept. of Computer Science & Engineering
U104
12


Seoul National University
Dept. of Computer Science & Engineering
U104
12


Seoul National University
Dept. of Computer Science & Engineering
U104
12


Seoul National University
Dept. of Computer Science & Engineering
U104
12


Seoul National University
Dept. of Computer Science & Engineering
U104
12

Homework: implement 1-bit Arithmetic Logic Unit (ALU) in Verilog and simulate all possible outputs using Verilog test bench.

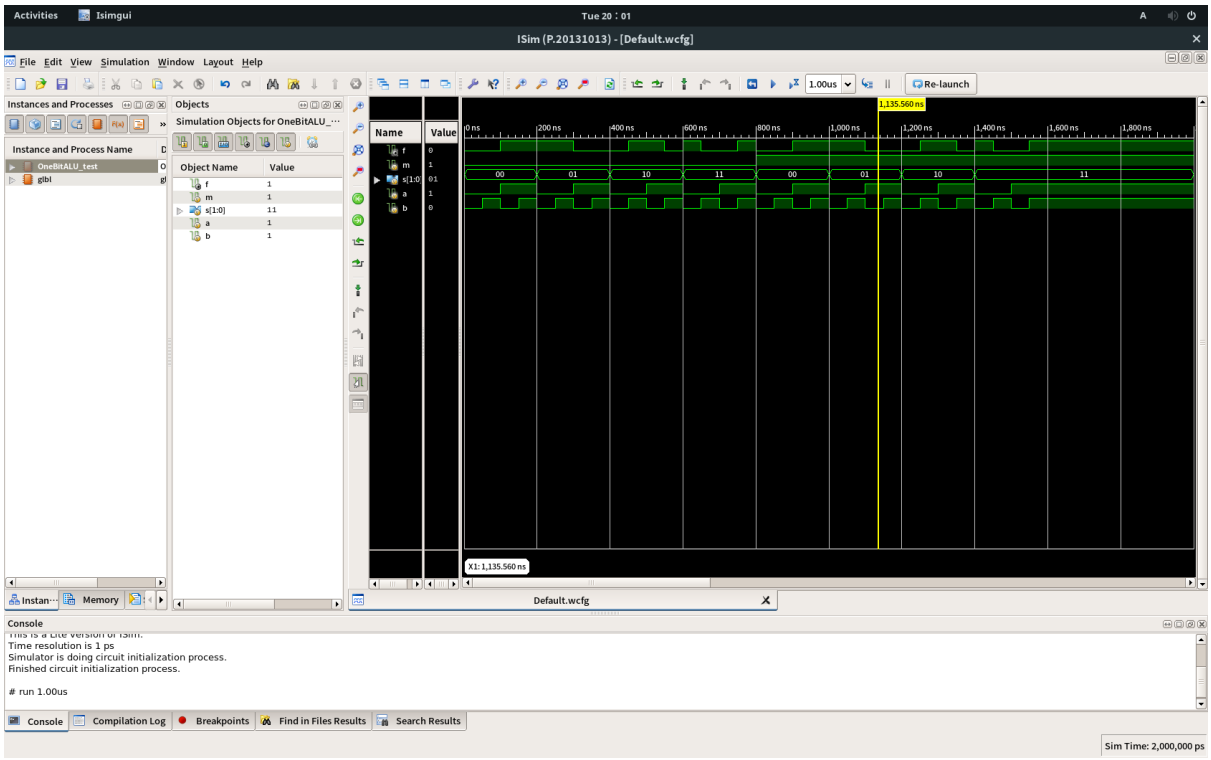
1-bit ALU (code):

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:       19:45:04 04/24/2018
// Design Name:
// Module Name:       OneBitALU
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module OneBitALU(
      input m,
```

```
input [1:0] s,
input a,
input b,
output f
);

	reg out;

	assign sel = {m, s[0]};
	assign f = out;

	always@(m or s or a or b)
		begin
			if(s[1]==1'b0)
				begin
					case(s[0])
						1'b0 : out = a;
						1'b1 : out = !a;
					endcase
				end
			else
				begin
					case(sel)
						2'b00 : out = (!a & b) | (!b & a);
						2'b01 : out = !((!a & b) | (!b & a));
						2'b00 : out = a | b;
```

```
                                        2'b00 : out = (!a) | b;

                                endcase

                        end

                end

endmodule
```

Result:



Test code for 1-bit ALU:

```
`timescale 1ns / 1ps


////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:     19:57:06 04/24/2018
// Design Name:     OneBitALU
// Module Name:     /csehome/ld15/OneBitALU/OneBitALU_test.v
// Project Name:    OneBitALU
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: OneBitALU
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////


module OneBitALU_test;
```

```
// Inputs
reg m;
reg [1:0] s;
reg a;
reg b;

// Outputs
wire f;

// Instantiate the Unit Under Test (UUT)
OneBitALU uut (
        .m(m),
        .s(s),
        .a(a),
        .b(b),
        .f(f)
);

initial begin
        // Initialize Inputs
        m = 0; s = 0; a = 0; b = 0; #50;
        b = 1; #50;
        a = 1; b = 0; #50;
        b = 1; #50;
        s = 1; a = 0; b = 0; #50;
```

```
b = 1; #50;

a = 1; b = 0; #50;

b = 1; #50;

s = 2; a = 0; b = 0; #50;

b = 1; #50;

a = 1; b = 0; #50;

b = 1; #50;

s = 3; a = 0; b = 0; #50;

b = 1; #50;

a = 1; b = 0; #50;

b = 1; #50;

m = 1; s = 0; a = 0; b = 0; #50;

b = 1; #50;

a = 1; b = 0; #50;

b = 1; #50;

s = 1; a = 0; b = 0; #50;

b = 1; #50;

a = 1; b = 0; #50;

b = 1; #50;

s = 2; a = 0; b = 0; #50;

b = 1; #50;

a = 1; b = 0; #50;

b = 1; #50;

s = 3; a = 0; b = 0; #50;

b = 1; #50;

a = 1; b = 0; #50;
```

```
                b = 1; #50;




        end


endmodule
```