Part 1

Description:

- In this lab, we will create a simple calculator that will be able to add and subtract 4 bit signed binary numbers, and display the output as a hex digit on the seven-segment display only if the output is valid, and otherwise turn the display off.

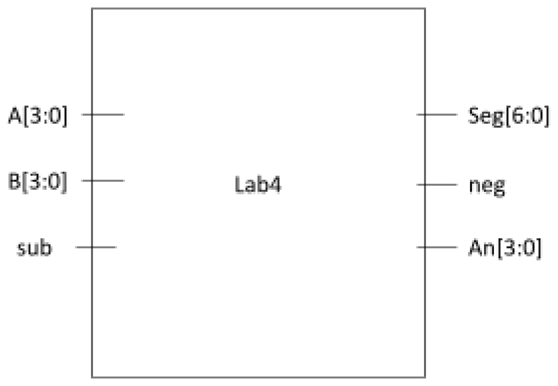

Figure 1.1.0: Top Level black box diagram

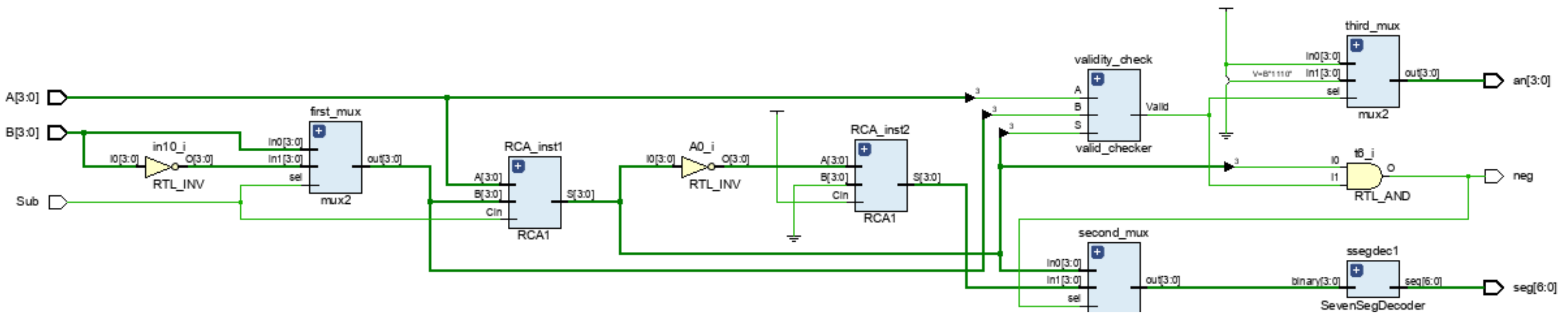

Figure 1.1.1: Top Level Structural Model/Schematic

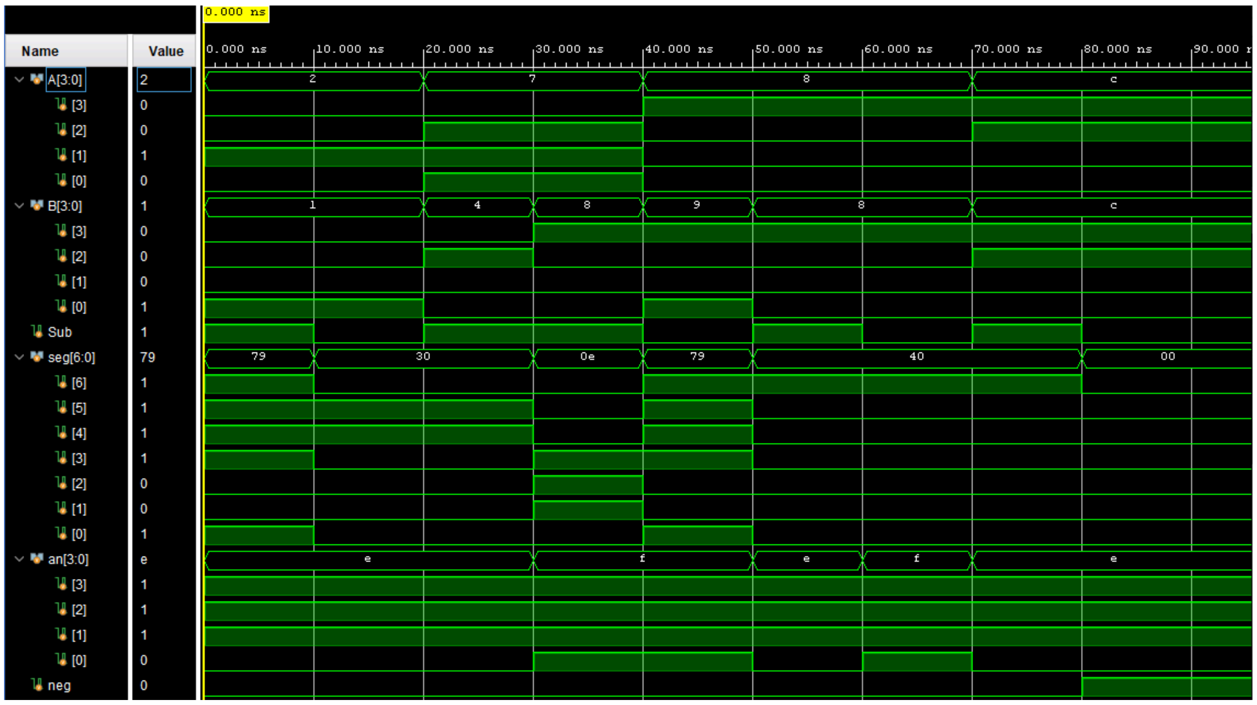

Figure 1.1.2: Top Level timing diagram of test cases

Figure 1.1.3: Timing diagram for the seven segment decoder. Showcases all possible inputs and their outputs.

Design Code for Top Level:

```
module Lab4 (
        input [3:0] A ,
        input [3:0] B ,
        input Sub,
        output [6:0] seg,
        output [3:0] an,
        output neg );

        logic [3:0] t1;
        logic [3:0] t3;
        logic [3:0] t4;
        logic [3:0] t5;
        logic t6;
        logic t7;
        assign t6 = t3[3] & t7;
        assign neg = t6;

        RCA1 RCA_inst1 (.A(A), .B(t1),.Cin(Sub), .S(t3), .C0());
        mux2 first_mux(.in0(B),.in1(~B),.sel(Sub),.out(t1));
        RCA1 RCA_inst2 (.A(~t3),.B(4'b0000),.Cin(1),.S(t4), .C0());
        mux2 second_mux(.in0(t3),.in1(t4),.sel(t6),.out(t5));
        valid_checker validity_check(.A(A[3]),.B(t1[3]),.S(t3[3]),.Valid(t7));
        mux2 third_mux(.in0(4'b1111),.in1(4'b1110),.sel(t7),.out(an));
        SevenSegDecoder ssegdec1(.binary(t5),.seg(seg));



endmodule
```

<u>Simulation Code for Top Level:</u>

```
module Lab4sim();
    logic [3:0]A,B;
    logic Sub;
    logic [6:0]seg;
    logic [3:0] an;
    logic neg;
    Lab4 Lab4_inst(.A(A), .B(B), .Sub(Sub), .seg(seg), .an(an), .neg(neg) );
   initial
   begin

    A = 2; B = 1; Sub = 1;
      #10
      //if(Sub!==1)$display("Error", A, B, Sub);
    A = 2; B = 1; Sub = 0;
      #10
     // if(Sub!==0)$display("Error", A, B, Sub);
    A = 7; B = 4; Sub = 1;
      #10
      //if(Sub!==0)$display("Error", A, B, Sub);
    A = 7; B = -8; Sub = 1;
      #10
    A = -8; B = -7; Sub = 0;
      #10
     // if(Sub!==0)$display("Error", A, B, Sub);
     A = -8; B = -8; Sub = 1;
      #10
      //if(Sub!==1)$display("Error", A, B, Sub);
```

```
  A = -8; B = -8; Sub = 0;
  #10
  //if(Sub!==0)$display("Error", A, B, Sub);
A = -4; B = -4; Sub = 1;
  #10
  //if(Sub!==1)$display("Error", A, B, Sub);
A = -4; B = -4; Sub = 0;
  #10
  //if(Sub!== 0)$display("Error", A, B, Sub);

$display("Finished");
end
```

Part 2

This part covers the design of the validity checker module that outputs a ‘1’ when the result is valid.

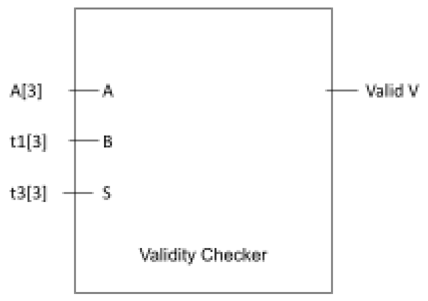

Figure 1.2.0: Black Box Diagram showing the validity checker’s 3 inputs and single output.

index	A[3]	t1[3]	t3[3]	Valid
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	1
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

Figure 1.2.1: Truth table representing the boolean logic present in the validity checker.

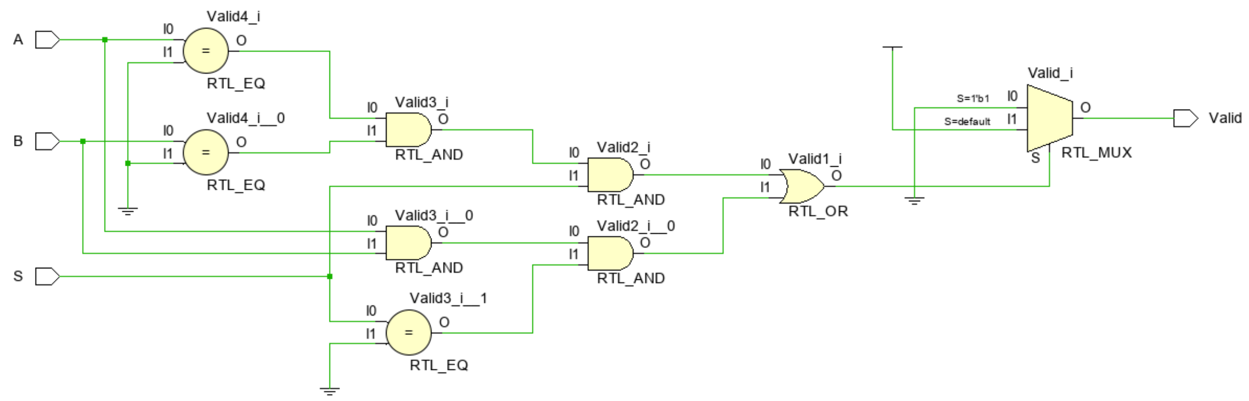

Figure 1.2.2: Circuit schematic of the validity checker

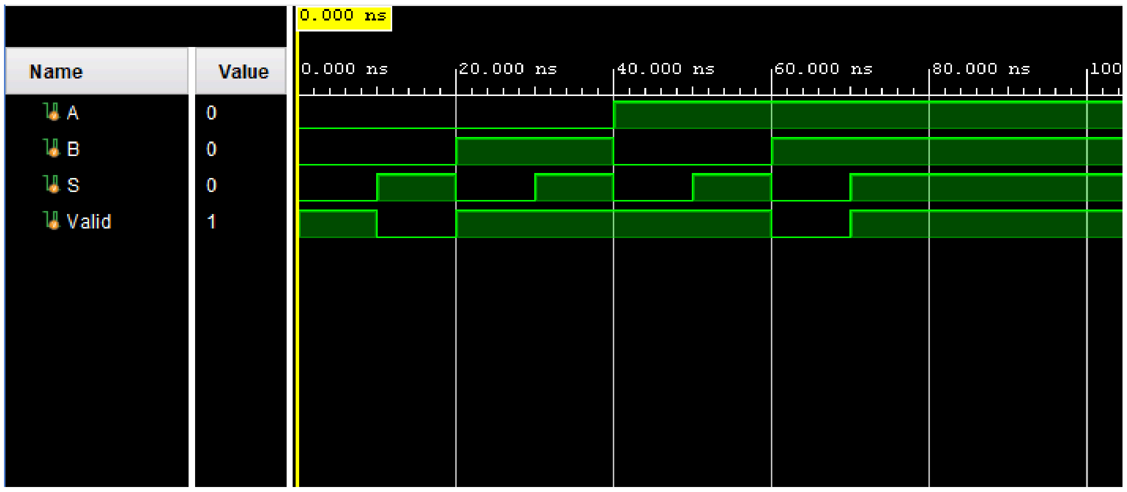

Figure 1.2.2: Timing diagram for the validity checker. Showcases all possible inputs and their outputs.

<u>Validity Checker Design Code:</u>

```
module valid_checker(
   input A,
   input B,
   input S,
   output logic Valid );
   always_comb
      begin
      if((A == 0 && B == 0 && S == 1) || (A==1 && B==1 && S == 0))
      Valid = 0;
      else
      Valid = 1;

   end
```

```
endmodule
```

Validity Checker Sim Code:

```
module validity_checksim();
    logic A,B,S,Valid;

    valid_checker validity_check_inst(.*);
initial
   begin

    A = 0; B = 0; S = 0;
      #10
      if(Valid!==1)$display("Error", A, B, S);
    A = 0; B = 0; S = 1;
      #10
      if(Valid!==0)$display("Error", A, B, S);
    A = 0; B = 1; S = 0;
      #10
      if(Valid!==1)$display("Error", A, B, S);
    A = 0; B = 1; S = 1;
      #10
      if(Valid!==1)$display("Error", A, B, S);
    A = 1; B = 0; S = 0;
      #10
      if(Valid!==1)$display("Error", A, B, S);
    A = 1; B = 0; S = 1;
      #10
      if(Valid!==1)$display("Error", A, B, S);
    A = 1; B = 1; S = 0;
      #10
      if(Valid!==0)$display("Error", A, B, S);
    A = 1; B = 1; S = 1;
      #10
      if(Valid!==1)$display("Error", A, B, S);

    $display("Finished");
    end

endmodule
```

Part 3

In this part of the lab, the RCA is updated with a new input.

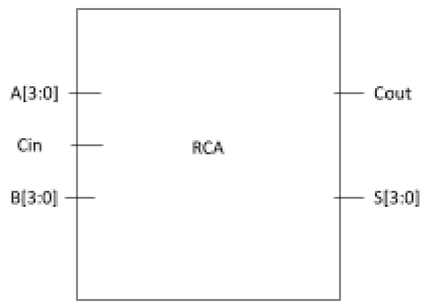

Figure 1.3.0: Black Box Diagram for the updated RCA showing the new input, Cin.

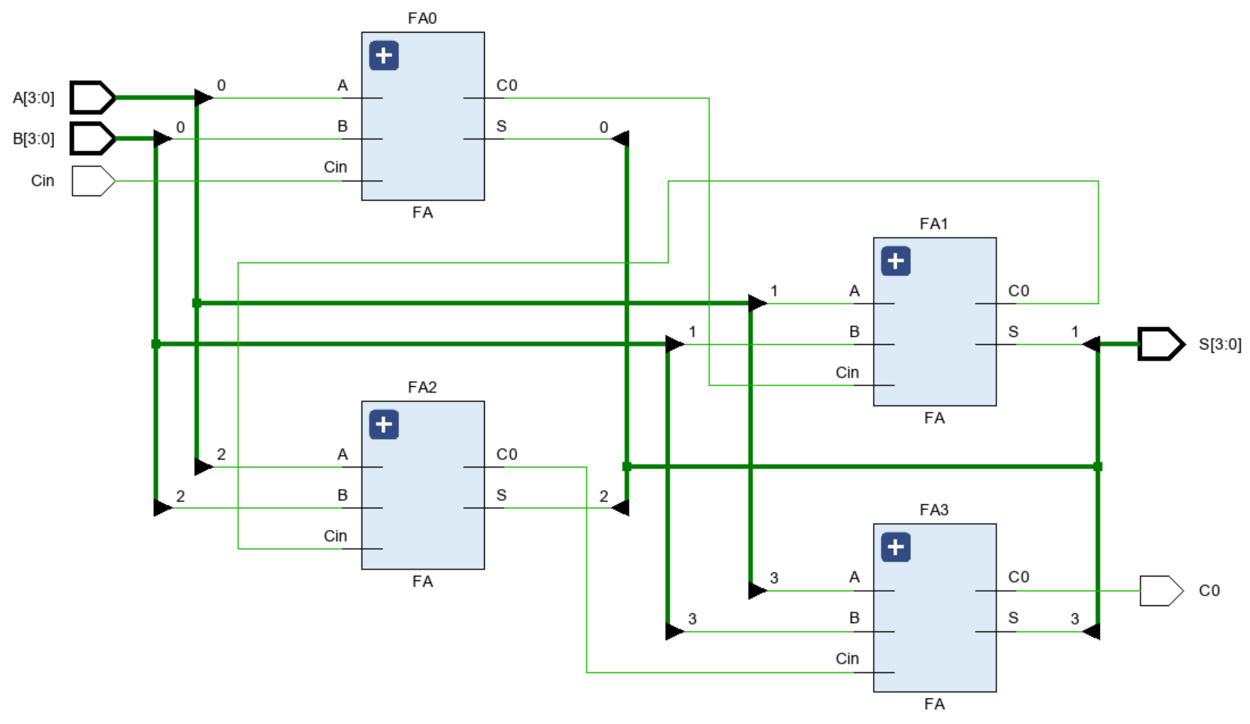

Figure 1.3.1: Schematic diagram of the RCA circuit.

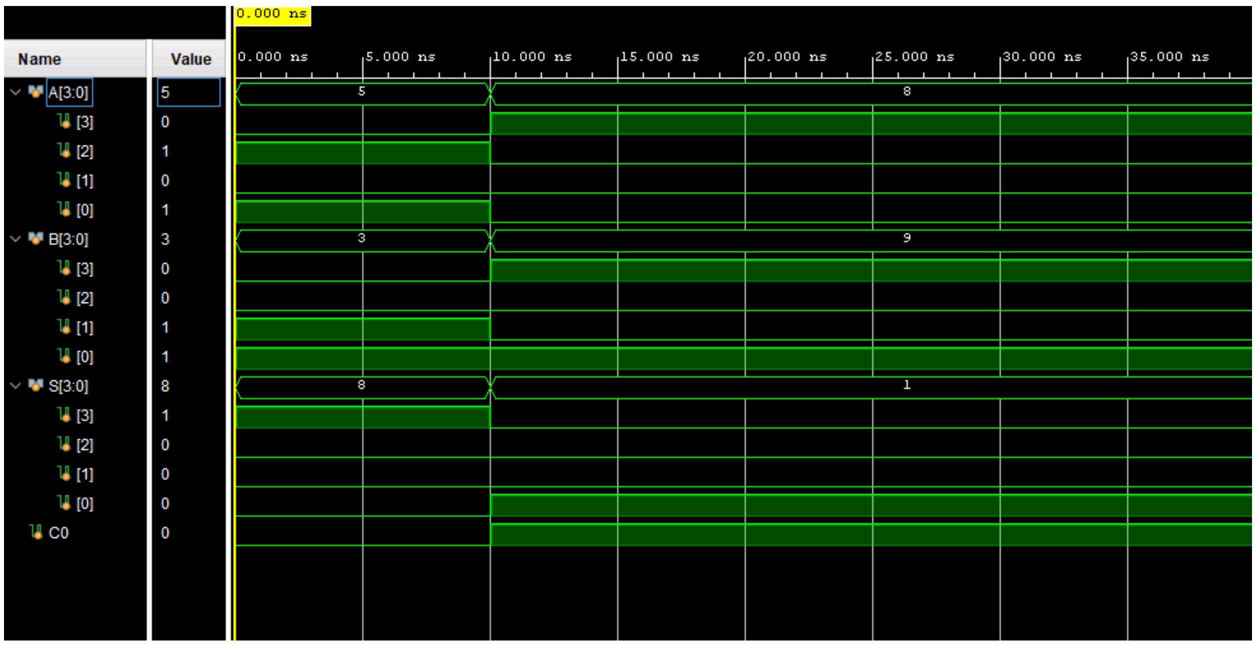

Figure 1.3.2: Timing diagram for the new, updated RCA.

<u>RCA Design Code:</u>

```
module RCA1(
    input [3:0] A,
    input [3:0] B,
    input  Cin,
    output [3:0] S,
    output C0
    );
    logic t1,t2,t3;

    FA FA0 (.A(A[0]), .B(B[0]), .Cin(Cin), .S(S[0]), .C0(t1));
    FA FA1 (.A(A[1]), .B(B[1]), .Cin(t1), .S(S[1]), .C0(t2));
    FA FA2 (.A(A[2]), .B(B[2]), .Cin(t2), .S(S[2]), .C0(t3));
    FA FA3 (.A(A[3]), .B(B[3]), .Cin(t3), .S(S[3]), .C0(C0));


endmodule
```