```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 03/07/2023 10:02:05 AM
// Design Name:
// Module Name: ALSU
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module
ALSU(clk,rst,A,B,cin,serial_in,red_op_A,red_op_B,opcode,bypass_A,bypass_B,direction,leds,out);
input clk,rst,cin,serial_in,red_op_A,red_op_B,bypass_A,bypass_B,direction ;
input [2:0] A,B,opcode;
output reg [15:0]leds;
output reg [5:0]out;

parameter INPUT_PRIORITY="A";
parameter FULL_ADDER="ON";
```

```
always @(posedge clk or posedge rst) begin
        if(rst) begin
                out <= 0;
                leds<=0;
        end
        else begin
                 if (bypass_A || bypass_B) begin
                  if (bypass_A && bypass_B) begin
                        if (INPUT_PRIORITY=="A")
                                begin
                                leds<=0;
                                out<=A;
                                end
                        else if (INPUT_PRIORITY=="B")
                        begin
                        leds<=0;
                                out<=B;
                                end
                  end
                  else if (bypass_A) begin
                        leds<=0;
                                out<=A;
                  end
                  else begin
                        leds<=0;
                                out<=B;
                  end
```

```
end
else begin
        case (opcode)
        3'b000:if (red_op_A || red_op_B) begin
                                if (red_op_A && red_op_B) begin
                                if (INPUT_PRIORITY=="A")

                                                begin
                                                out<=&A;
                                                leds<=0;
                                                end


                                else if (INPUT_PRIORITY=="B")
                                                begin
                                                leds<=0;
                                                out<=&B;
                                                end
                                end
                                else if(red_op_A) begin
                                        out<=&A;
                                                leds<=0;
                                end
                                else begin
                                        leds<=0;
                                                out<=&B;
                                end
                        end
                        else
                        begin
                        leds<=0;
                        out<=A&B;
```

```
                end
3'b001:if (red_op_A || red_op_B) begin
                        if (red_op_A && red_op_B) begin
                        if (INPUT_PRIORITY=="A")

                                begin
                                out<=^A;
                                leds<=0;
                                end


                        else if (INPUT_PRIORITY=="B")
                                begin
                                leds<=0;
                                out<=^B;
                                end
                        end
                        else if(red_op_A) begin
                                out<=^A;
                                leds<=0;
                        end
                        else begin
                                leds<=0;
                                out<=^B;
                        end
                end
                else begin
                out<=A^B;
                leds<=0;
                end
3'b010:if (red_op_A || red_op_B) begin //invalid
                        out<=0;
```

```
                leds<=~leds;
            end
            else if (FULL_ADDER=="ON") begin
                out<=A+B+cin;
                leds<=0;
            end
            else if (FULL_ADDER=="OFF") begin
                out<=A+B;
                leds<=0;
            end
        3'b011:if (red_op_A || red_op_B) begin //invalid
                out<=0;
                leds<=~leds;
            end
            else begin
            out<=A*B;
            leds<=0;
            end
        3'b100:if (red_op_A || red_op_B) begin //invalid
                out<=0;
                leds<=~leds;
            end
            else begin
            if (direction) begin
                out<={out[4:0],serial_in};
                leds<=0;
            end
            else begin
                out<={serial_in,out[5:1]};
                leds<=0;
            end
```

```
                    end
                3'b101:if (red_op_A || red_op_B) begin //invalid
                            out<=0;
                            leds<=~leds;
                    end
                    else begin
                    if (direction) begin
                            out<={out[4:0],out[5]};
                            leds<=0;
                    end
                    else begin
                            out<={out[0],out[5:1]};
                            leds<=0;
                    end
                    end
                3'b110: begin //invalid
                            out<=0;
                            leds<=~leds;
                    end
                3'b111: begin //invalid
                            out<=0;
                            leds<=~leds;
                    end
                endcase
            end
        end
end

endmodule
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
```

```
// Company:
// Engineer:
//
// Create Date: 03/07/2023 02:21:06 PM
// Design Name:
// Module Name: ALSU_Bouns
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module ALSU_Bouns(clk,rst,A,B,cin,serial_in,red_op_A,red_op_B,opcode,bypass_A,
bypass_B,direction,leds,anode,cathode);
input clk,rst,cin,serial_in,red_op_A,red_op_B,bypass_A,bypass_B,direction ;
input [2:0] A,B,opcode;
output  [15:0]leds;
output reg [3:0]anode;
output [6:0]cathode;
wire [5:0]out;
reg cin_D,serial_in_D,red_op_A_D,red_op_B_D,bypass_A_D,bypass_B_D,direction_D ;
reg [2:0] A_D,B_D,opcode_D;
parameter INPUT_PRIORITY="A";
```

```
parameter FULL_ADDER="ON";
ALSU #(.INPUT_PRIORITY(INPUT_PRIORITY),.FULL_ADDER(FULL_ADDER))
DUT(clk,rst,A_D,B_D,cin_D,serial_in_D,red_op_A_D,red_op_B_D,opcode_D,bypass_A_D,bypass_B_
D,direction_D,leds,out);
wire [3:0]D1,D2,D3,D4;
reg [3:0] D;
convert C1(out,D1,D2,D3,D4);
convert_7seg C2(D,cathode);
integer counter=0;
always @(posedge clk or posedge rst) begin
   if(rst) begin
       anode<=0;
      end
    else begin
    A_D<=A;
    B_D<=B;
    cin_D<=cin;
    serial_in_D<=serial_in;
    red_op_A_D<=red_op_A;
    red_op_B_D<=red_op_B;
    opcode_D<=opcode;
    bypass_A_D<=bypass_A;
    bypass_B_D<=bypass_B;
    direction_D<=direction;
      if(counter==0)begin
         D<=D1;
         anode<=4'b0001;
         counter<=counter+1;
      end
      else if (counter==1) begin
         D<=D2;
```

```
            anode<=4'b0010;
            counter<=counter+1;
        end
        else if (counter==2) begin
            D<=D3;
            anode<=4'b0100;
            counter<=counter+1;
        end
        else if (counter==3) begin
            D<=D4;
            anode<=4'b1000;
            counter<=counter-3;
        end
    end
end
endmodule
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 03/07/2023 03:18:37 PM
// Design Name:
// Module Name: convert
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
```

```
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module convert(out,D1,D2,D3,D4);
input [5:0]out;
output [3:0]D1,D2,D3,D4;
assign D1=out%10;
assign D2=(out%100)/10;
assign D3=(out%1000)/100;
assign D4=out/1000;
endmodule

module convert_7seg(D,cathode);
input [3:0] D;
output reg [6:0]cathode;
always @(*)
begin
 case(D)
 4'b0000: cathode = 7'b0000001; // "0"
 4'b0001: cathode = 7'b1001111; // "1"
 4'b0010: cathode = 7'b0010010; // "2"
 4'b0011: cathode = 7'b0000110; // "3"
 4'b0100: cathode = 7'b1001100; // "4"
 4'b0101: cathode = 7'b0100100; // "5"
 4'b0110: cathode = 7'b0100000; // "6"
 4'b0111: cathode = 7'b0001111; // "7"
 4'b1000: cathode = 7'b0000000; // "8"
```

```
 4'b1001: cathode = 7'b0000100; // "9"
 default: cathode = 7'b0000001; // "0"
 endcase
end
endmodule
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 03/07/2023 03:51:34 PM
// Design Name:
// Module Name: ALSU_Bouns_tb
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module ALSU_Bouns_tb();

reg clk,rst,cin,serial_in,red_op_A,red_op_B,bypass_A,bypass_B,direction ;
reg [2:0] A,B,opcode;
```

```
wire [15:0]leds;
wire[3:0]anode;
wire [6:0]cathode;
parameter INPUT_PRIORITY="A";
parameter FULL_ADDER="ON";
ALSU_Bouns #(.INPUT_PRIORITY(INPUT_PRIORITY),.FULL_ADDER(FULL_ADDER)) DUT
(clk,rst,A,B,cin,serial_in,red_op_A,red_op_B,opcode,bypass_A,
bypass_B,direction,leds,anode,cathode);
integer i=0;
initial begin
clk=0;
forever
#2 clk=~clk;
end
initial begin
   rst=1;
   repeat(5)@(negedge clk);
   rst=0;
   repeat(5)@(negedge clk);
   bypass_A=1;
   for(i=0;i<10;i=i+1)begin
      {cin,serial_in,red_op_A,red_op_B,bypass_B,direction,opcode,A,B}=$random;
repeat(4)@(negedge clk);
   end
   repeat(5)@(negedge clk);
   bypass_B=1;
   for(i=0;i<10;i=i+1)begin
      {cin,serial_in,red_op_A,red_op_B,bypass_A,direction,opcode,A,B}=$random;
repeat(4)@(negedge clk);
   end
   repeat(5)@(negedge clk);
```

```
    bypass_B=0;
    bypass_A=0;
    for(i=0;i<10;i=i+1)begin
       {cin,serial_in,red_op_A,red_op_B,direction,opcode,A,B}=$random;
repeat(4)@(negedge clk);
    end
    repeat(5)@(negedge clk);
    red_op_A=1;
    for(i=0;i<10;i=i+1)begin
       {cin,serial_in,red_op_B,direction,opcode,A,B}=$random;
repeat(4)@(negedge clk);
    end
    repeat(5)@(negedge clk);
    red_op_B=1;
    for(i=0;i<10;i=i+1)begin
       {cin,serial_in,red_op_A,direction,opcode,A,B}=$random;
repeat(4)@(negedge clk);
    end
    repeat(5)@(negedge clk);
    red_op_A=0;
    red_op_B=0;
    for(i=0;i<100;i=i+1)begin
       {cin,serial_in,direction,opcode,A,B}=$random;
repeat(4)@(negedge clk);
    end
$stop;
end
endmodule
```