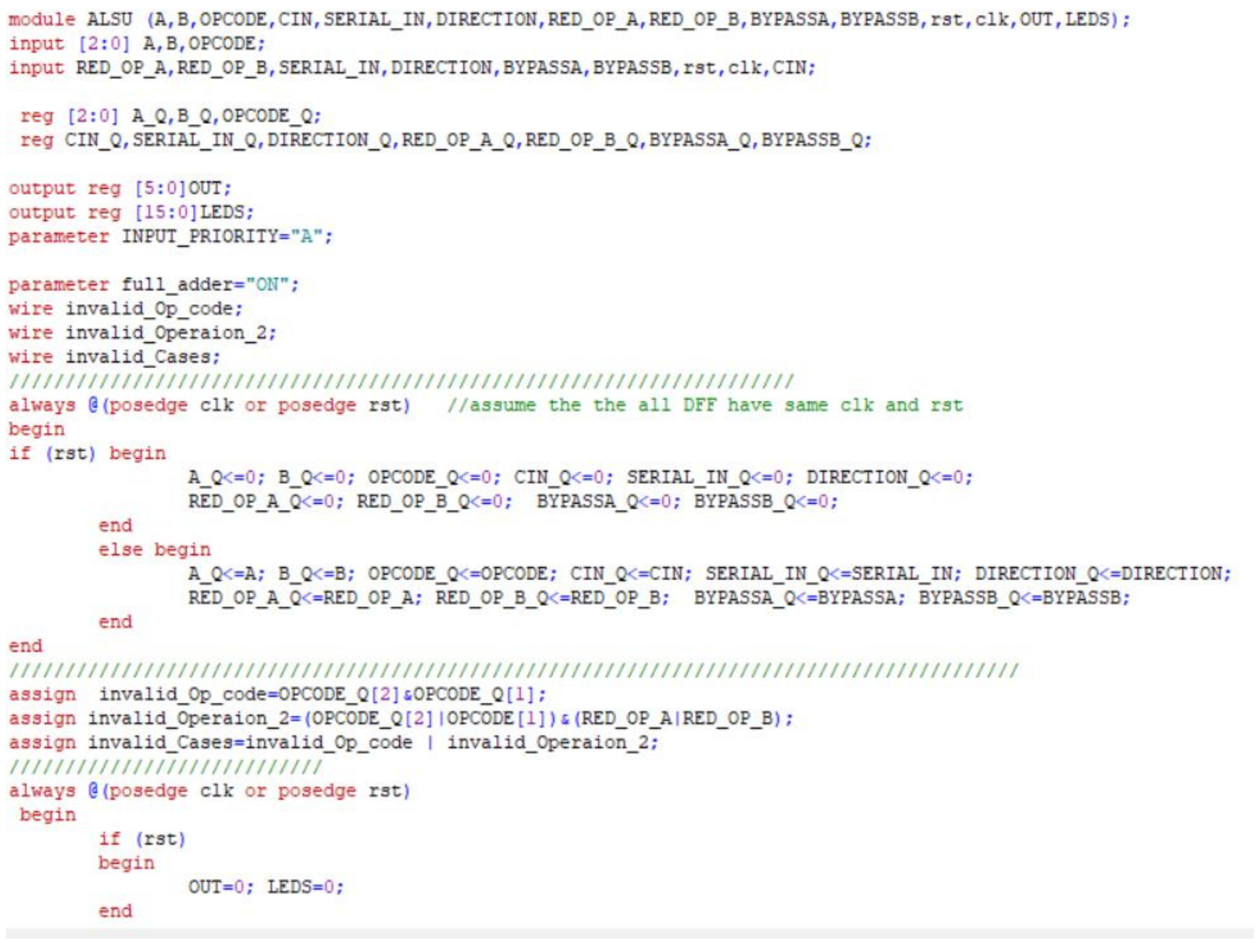
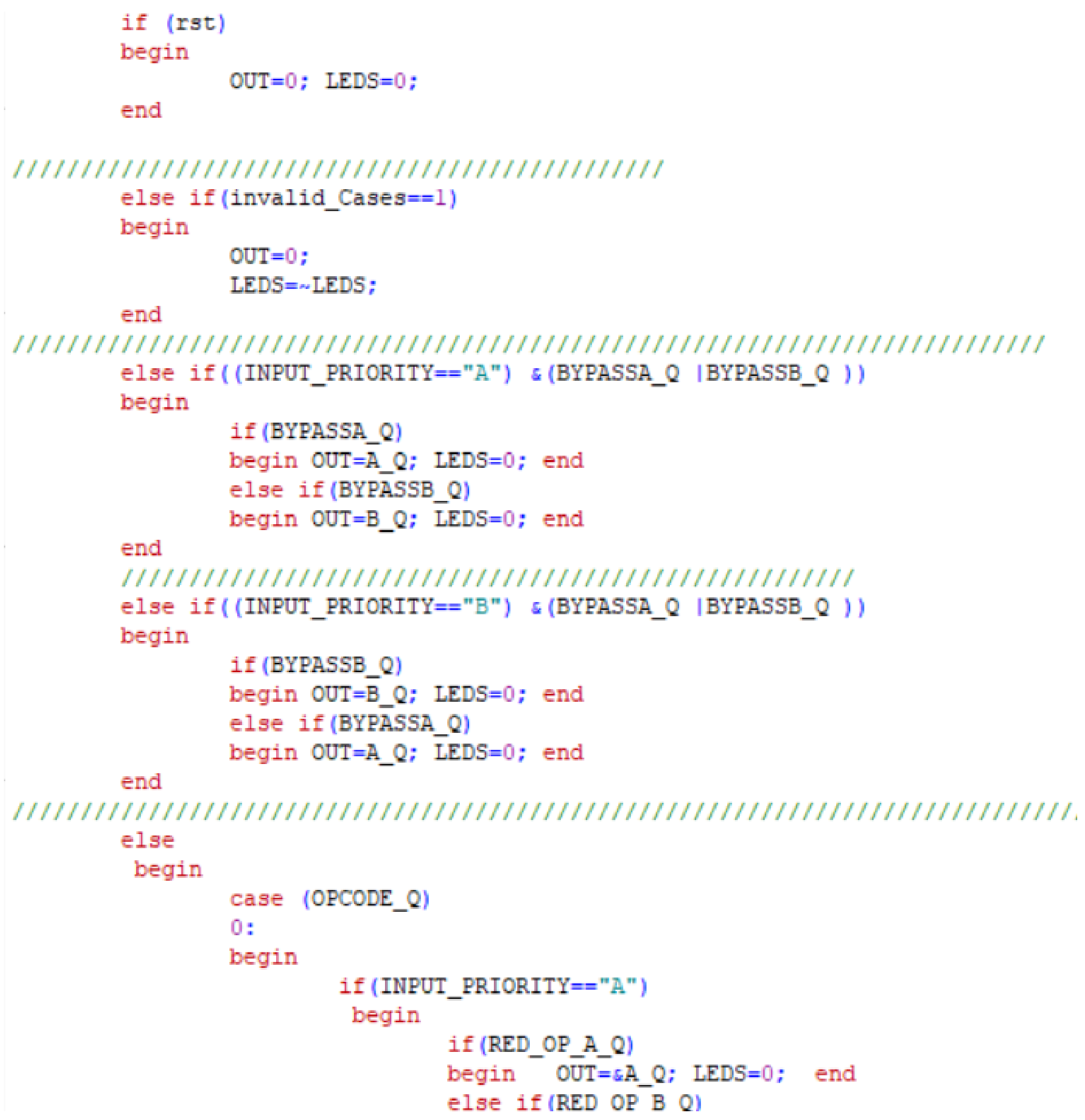
ALSU

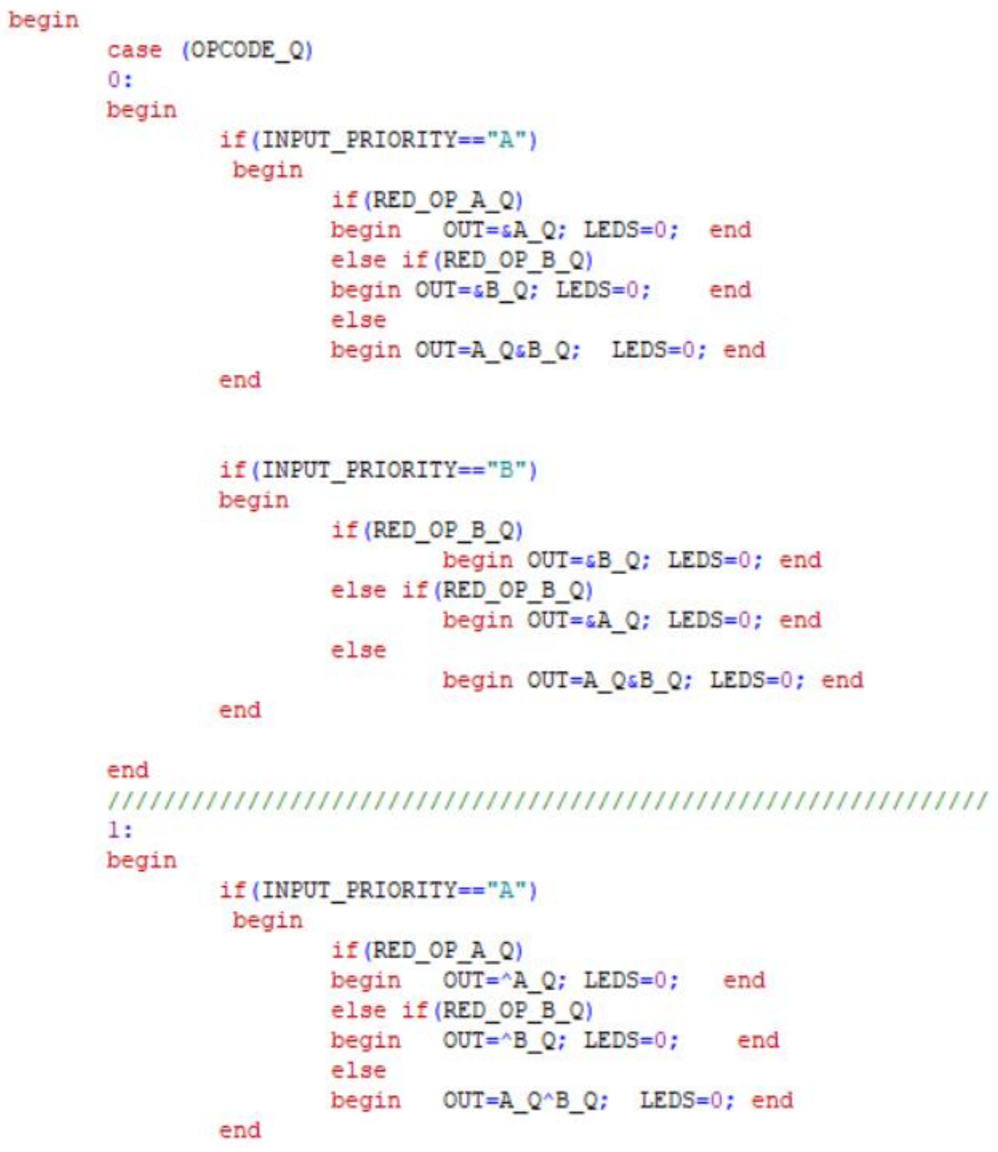
1-Design



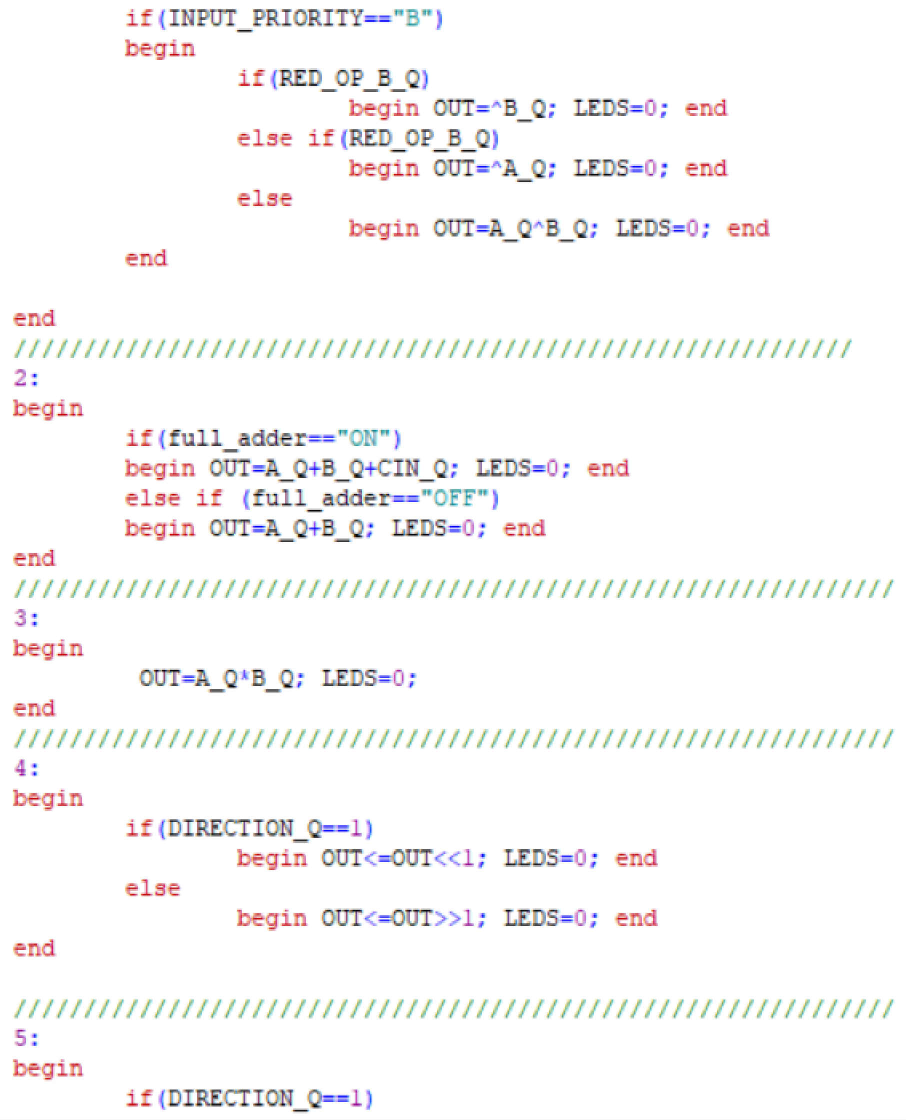
```
module ALSU (A,B,OPCODE,CIN,SERIAL_IN,DIRECTION,RED_OP_A,RED_OP_B,BYPASSA,BYPASSB,rst,clk,OUT,LEDS);
input [2:0] A,B,OPCODE;
input RED_OP_A,RED_OP_B,SERIAL_IN,DIRECTION,BYPASSA,BYPASSB,rst,clk,CIN;

 reg [2:0] A_Q,B_Q,OPCODE_Q;
 reg CIN_Q,SERIAL_IN_Q,DIRECTION_Q,RED_OP_A_Q,RED_OP_B_Q,BYPASSA_Q,BYPASSB_Q;

output reg [5:0]OUT;
output reg [15:0]LEDS;
parameter INPUT_PRIORITY="A";

parameter full_adder="ON";
wire invalid_Op_code;
wire invalid_Operaion_2;
wire invalid_Cases;
//////////////////////////////////////////////////////////////////////
always @(posedge clk or posedge rst)   //assume the the all DFF have same clk and rst
begin
if (rst) begin
                A_Q<=0; B_Q<=0; OPCODE_Q<=0; CIN_Q<=0; SERIAL_IN_Q<=0; DIRECTION_Q<=0;
                RED_OP_A_Q<=0; RED_OP_B_Q<=0;  BYPASSA_Q<=0; BYPASSB_Q<=0;
        end
        else begin
                A_Q<=A; B_Q<=B; OPCODE_Q<=OPCODE; CIN_Q<=CIN; SERIAL_IN_Q<=SERIAL_IN; DIRECTION_Q<=DIRECTION;
                RED_OP_A_Q<=RED_OP_A; RED_OP_B_Q<=RED_OP_B;  BYPASSA_Q<=BYPASSA; BYPASSB_Q<=BYPASSB;
        end
end
/////////////////////////////////////////////////////////////////////////////////////////
assign  invalid_Op_code=OPCODE_Q[2]&OPCODE_Q[1];
assign invalid_Operaion_2=(OPCODE_Q[2]|OPCODE[1])&(RED_OP_A|RED_OP_B);
assign invalid_Cases=invalid_Op_code | invalid_Operaion_2;
////////////////////////////
always @(posedge clk or posedge rst)
 begin
        if (rst)
        begin
                OUT=0; LEDS=0;
        end
```



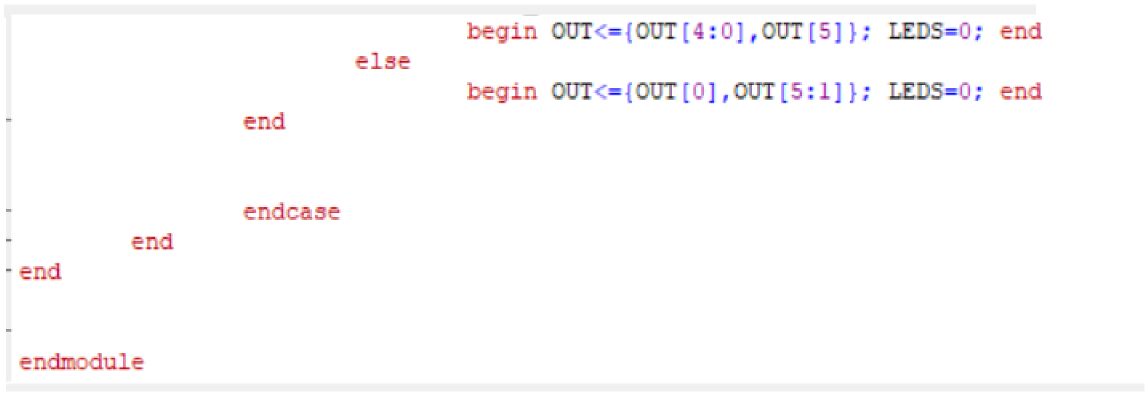
```
        if (rst)
        begin
                OUT=0; LEDS=0;
        end

////////////////////////////////////////////////
        else if(invalid_Cases==1)
        begin
                OUT=0;
                LEDS=~LEDS;
        end
////////////////////////////////////////////////////////////////////////////
        else if((INPUT_PRIORITY=="A") &(BYPASSA_Q |BYPASSB_Q ))
        begin
                if(BYPASSA_Q)
                begin OUT=A_Q; LEDS=0; end
                else if(BYPASSB_Q)
                begin OUT=B_Q; LEDS=0; end
        end
        //////////////////////////////////////////////////////
        else if((INPUT_PRIORITY=="B") &(BYPASSA_Q |BYPASSB_Q ))
        begin
                if(BYPASSB_Q)
                begin OUT=B_Q; LEDS=0; end
                else if(BYPASSA_Q)
                begin OUT=A_Q; LEDS=0; end
        end
//////////////////////////////////////////////////////////////////////////////.
        else
         begin
                case (OPCODE_Q)
                0:
                begin
                        if(INPUT_PRIORITY=="A")
                         begin
                                if(RED_OP_A_Q)
                                begin   OUT=&A_Q; LEDS=0;   end
                                else if(RED_OP_B_Q)
```



```
begin
       case (OPCODE_Q)
       0:
       begin
               if(INPUT_PRIORITY=="A")
                begin
                       if(RED_OP_A_Q)
                       begin   OUT=&A_Q; LEDS=0;  end
                       else if(RED_OP_B_Q)
                       begin OUT=&B_Q; LEDS=0;    end
                       else
                       begin OUT=A_Q&B_Q;  LEDS=0; end
               end


               if(INPUT_PRIORITY=="B")
               begin
                       if(RED_OP_B_Q)
                               begin OUT=&B_Q; LEDS=0; end
                       else if(RED_OP_B_Q)
                               begin OUT=&A_Q; LEDS=0; end
                       else
                               begin OUT=A_Q&B_Q; LEDS=0; end
               end

       end
       /////////////////////////////////////////////////////////////////
       1:
       begin
               if(INPUT_PRIORITY=="A")
                begin
                       if(RED_OP_A_Q)
                       begin   OUT=^A_Q; LEDS=0;   end
                       else if(RED_OP_B_Q)
                       begin   OUT=^B_Q; LEDS=0;    end
                       else
                       begin   OUT=A_Q^B_Q;  LEDS=0; end
               end
```

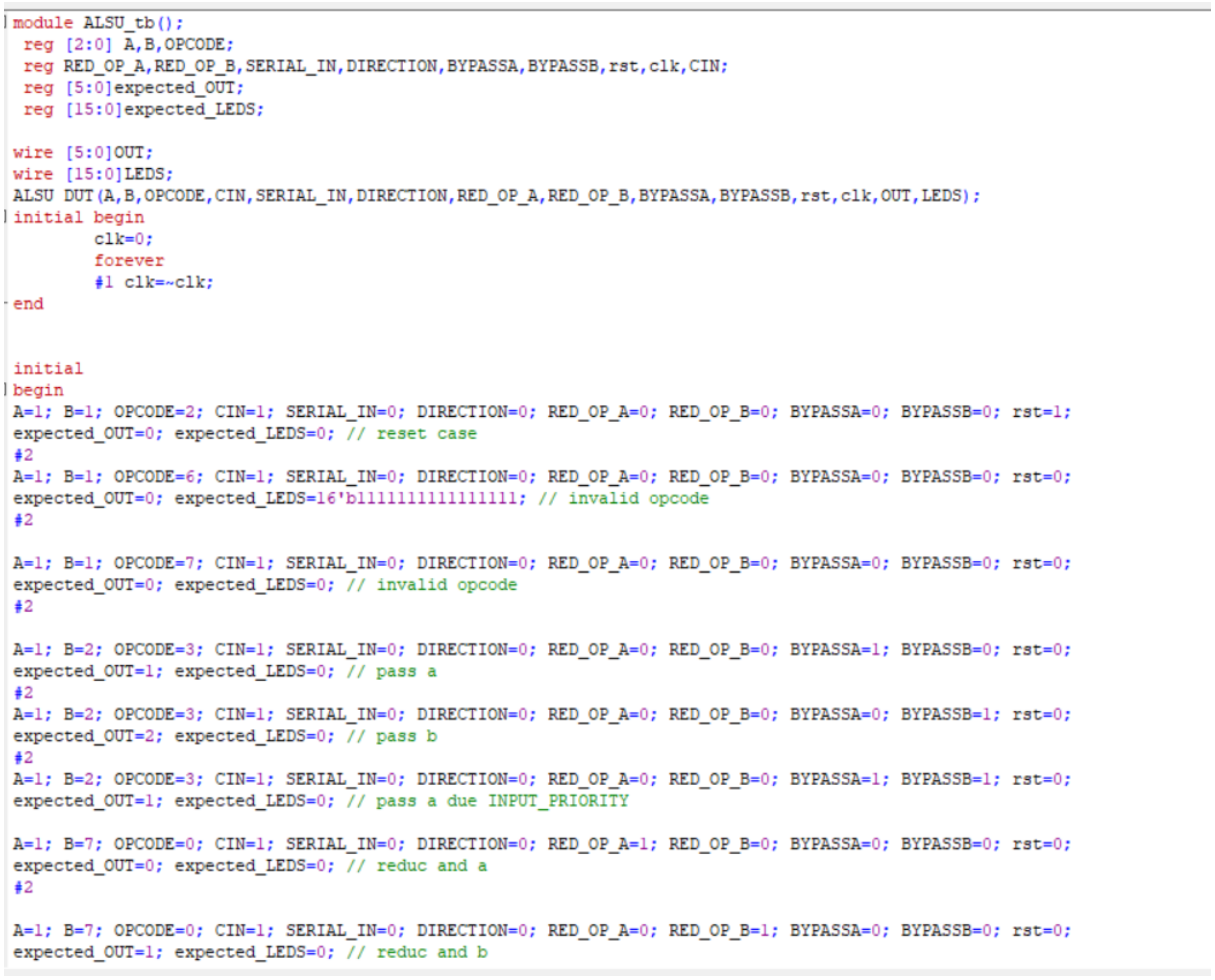


```
        if(INPUT_PRIORITY=="B")
        begin
                if(RED_OP_B_Q)
                        begin OUT=^B_Q; LEDS=0; end
                else if(RED_OP_B_Q)
                        begin OUT=^A_Q; LEDS=0; end
                else
                        begin OUT=A_Q^B_Q; LEDS=0; end
        end

end
/////////////////////////////////////////////////////////////
2:
begin
        if(full_adder=="ON")
        begin OUT=A_Q+B_Q+CIN_Q; LEDS=0; end
        else if (full_adder=="OFF")
        begin OUT=A_Q+B_Q; LEDS=0; end
end
////////////////////////////////////////////////////////////////
3:
begin
         OUT=A_Q*B_Q; LEDS=0;
end
////////////////////////////////////////////////////////////////
4:
begin
        if(DIRECTION_Q==1)
                begin OUT<=OUT<<1; LEDS=0; end
        else
                begin OUT<=OUT>>1; LEDS=0; end
end

////////////////////////////////////////////////////////////////
5:
begin
        if(DIRECTION_Q==1)
```

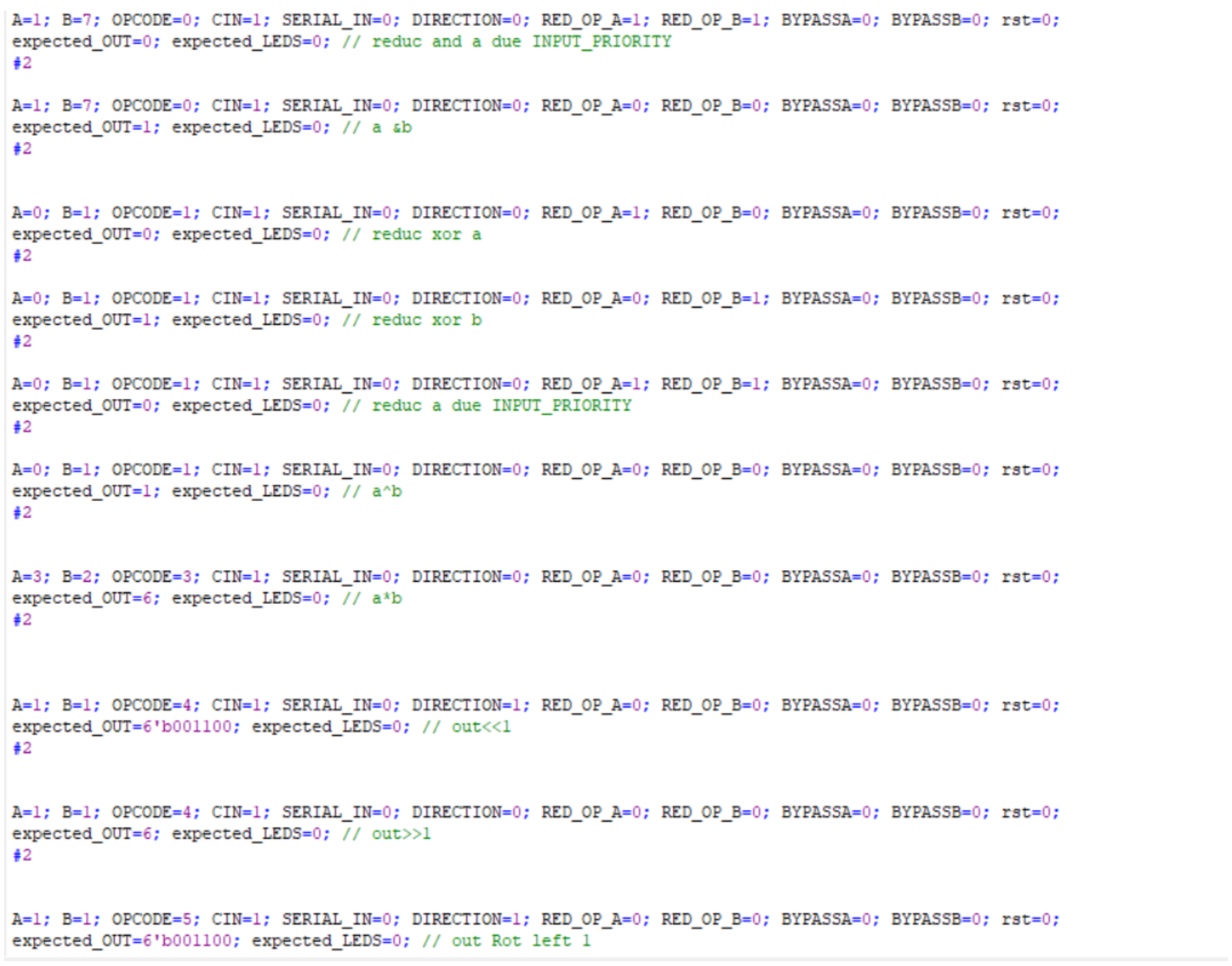


```
                                begin OUT<={OUT[4:0],OUT[5]}; LEDS=0; end
                        else
                                begin OUT<={OUT[0],OUT[5:1]}; LEDS=0; end
                end


                endcase
        end
end


endmodule
```

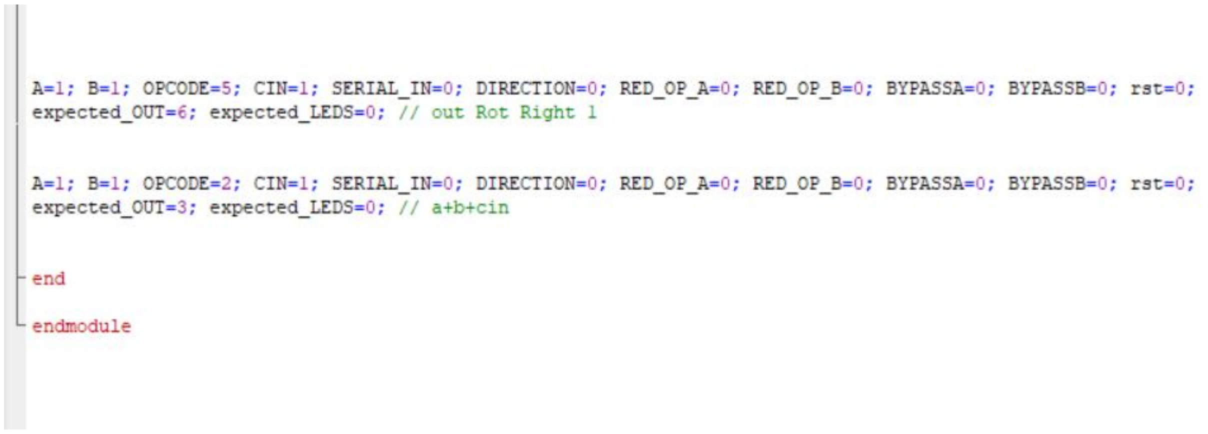
2-Test bench



```
module ALSU_tb();
 reg [2:0] A,B,OPCODE;
 reg RED_OP_A,RED_OP_B,SERIAL_IN,DIRECTION,BYPASSA,BYPASSB,rst,clk,CIN;
 reg [5:0]expected_OUT;
 reg [15:0]expected_LEDS;

wire [5:0]OUT;
wire [15:0]LEDS;
ALSU DUT(A,B,OPCODE,CIN,SERIAL_IN,DIRECTION,RED_OP_A,RED_OP_B,BYPASSA,BYPASSB,rst,clk,OUT,LEDS);
initial begin
        clk=0;
        forever
        #1 clk=~clk;
end


initial
begin
A=1; B=1; OPCODE=2; CIN=1; SERIAL_IN=0; DIRECTION=0; RED_OP_A=0; RED_OP_B=0; BYPASSA=0; BYPASSB=0; rst=1;
expected_OUT=0; expected_LEDS=0; // reset case
#2
A=1; B=1; OPCODE=6; CIN=1; SERIAL_IN=0; DIRECTION=0; RED_OP_A=0; RED_OP_B=0; BYPASSA=0; BYPASSB=0; rst=0;
expected_OUT=0; expected_LEDS=16'b1111111111111111; // invalid opcode
#2

A=1; B=1; OPCODE=7; CIN=1; SERIAL_IN=0; DIRECTION=0; RED_OP_A=0; RED_OP_B=0; BYPASSA=0; BYPASSB=0; rst=0;
expected_OUT=0; expected_LEDS=0; // invalid opcode
#2

A=1; B=2; OPCODE=3; CIN=1; SERIAL_IN=0; DIRECTION=0; RED_OP_A=0; RED_OP_B=0; BYPASSA=1; BYPASSB=0; rst=0;
expected_OUT=1; expected_LEDS=0; // pass a
#2
A=1; B=2; OPCODE=3; CIN=1; SERIAL_IN=0; DIRECTION=0; RED_OP_A=0; RED_OP_B=0; BYPASSA=0; BYPASSB=1; rst=0;
expected_OUT=2; expected_LEDS=0; // pass b
#2
A=1; B=2; OPCODE=3; CIN=1; SERIAL_IN=0; DIRECTION=0; RED_OP_A=0; RED_OP_B=0; BYPASSA=1; BYPASSB=1; rst=0;
expected_OUT=1; expected_LEDS=0; // pass a due INPUT_PRIORITY

A=1; B=7; OPCODE=0; CIN=1; SERIAL_IN=0; DIRECTION=0; RED_OP_A=1; RED_OP_B=0; BYPASSA=0; BYPASSB=0; rst=0;
expected_OUT=0; expected_LEDS=0; // reduc and a
#2

A=1; B=7; OPCODE=0; CIN=1; SERIAL_IN=0; DIRECTION=0; RED_OP_A=0; RED_OP_B=1; BYPASSA=0; BYPASSB=0; rst=0;
expected_OUT=1; expected_LEDS=0; // reduc and b
```

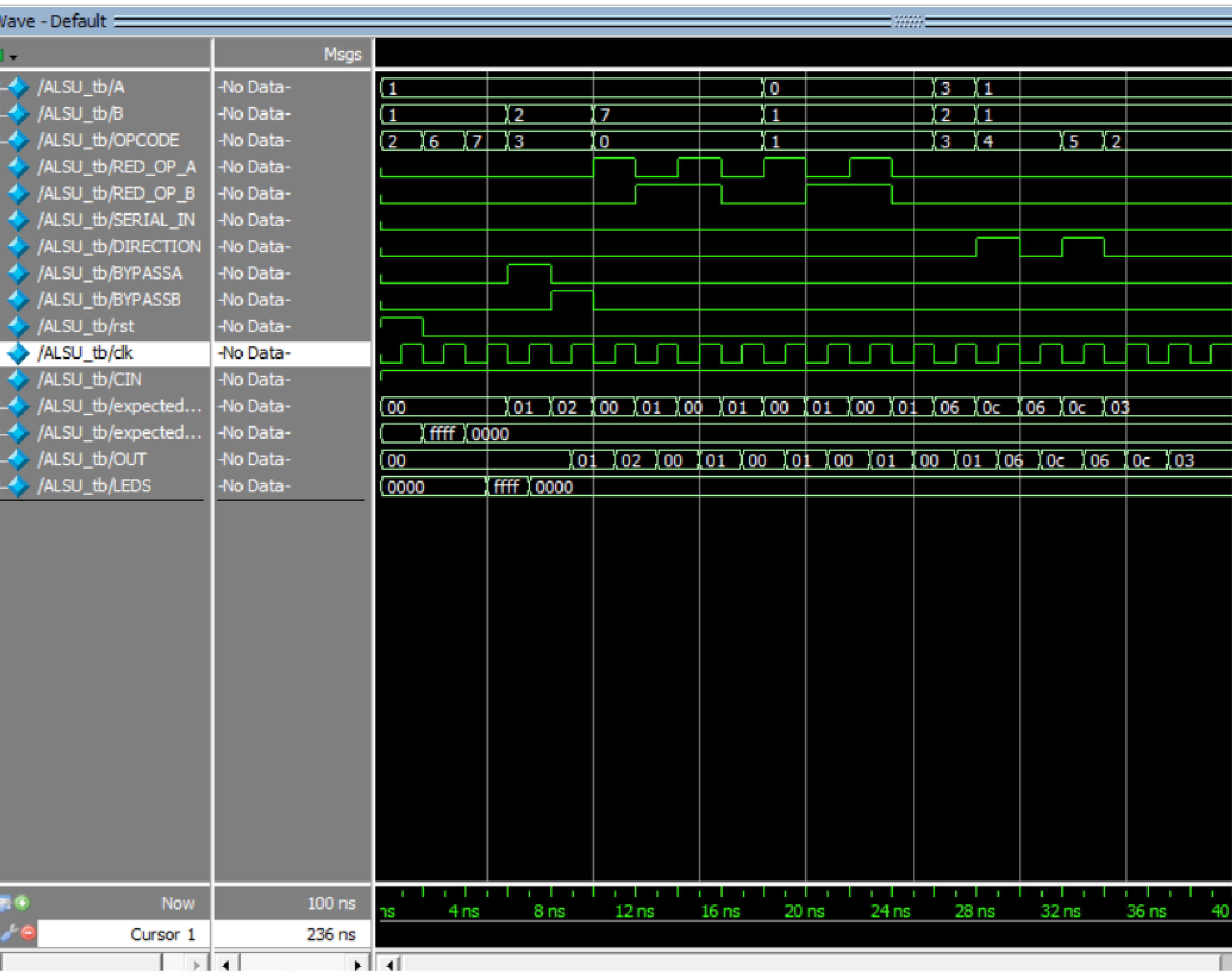


```
A=1; B=7; OPCODE=0; CIN=1; SERIAL_IN=0; DIRECTION=0; RED_OP_A=1; RED_OP_B=1; BYPASSA=0; BYPASSB=0; rst=0;
expected_OUT=0; expected_LEDS=0; // reduc and a due INPUT_PRIORITY
#2

A=1; B=7; OPCODE=0; CIN=1; SERIAL_IN=0; DIRECTION=0; RED_OP_A=0; RED_OP_B=0; BYPASSA=0; BYPASSB=0; rst=0;
expected_OUT=1; expected_LEDS=0; // a &b
#2


A=0; B=1; OPCODE=1; CIN=1; SERIAL_IN=0; DIRECTION=0; RED_OP_A=1; RED_OP_B=0; BYPASSA=0; BYPASSB=0; rst=0;
expected_OUT=0; expected_LEDS=0; // reduc xor a
#2

A=0; B=1; OPCODE=1; CIN=1; SERIAL_IN=0; DIRECTION=0; RED_OP_A=0; RED_OP_B=1; BYPASSA=0; BYPASSB=0; rst=0;
expected_OUT=1; expected_LEDS=0; // reduc xor b
#2

A=0; B=1; OPCODE=1; CIN=1; SERIAL_IN=0; DIRECTION=0; RED_OP_A=1; RED_OP_B=1; BYPASSA=0; BYPASSB=0; rst=0;
expected_OUT=0; expected_LEDS=0; // reduc a due INPUT_PRIORITY
#2

A=0; B=1; OPCODE=1; CIN=1; SERIAL_IN=0; DIRECTION=0; RED_OP_A=0; RED_OP_B=0; BYPASSA=0; BYPASSB=0; rst=0;
expected_OUT=1; expected_LEDS=0; // a^b
#2


A=3; B=2; OPCODE=3; CIN=1; SERIAL_IN=0; DIRECTION=0; RED_OP_A=0; RED_OP_B=0; BYPASSA=0; BYPASSB=0; rst=0;
expected_OUT=6; expected_LEDS=0; // a*b
#2



A=1; B=1; OPCODE=4; CIN=1; SERIAL_IN=0; DIRECTION=1; RED_OP_A=0; RED_OP_B=0; BYPASSA=0; BYPASSB=0; rst=0;
expected_OUT=6'b001100; expected_LEDS=0; // out<<1
#2


A=1; B=1; OPCODE=4; CIN=1; SERIAL_IN=0; DIRECTION=0; RED_OP_A=0; RED_OP_B=0; BYPASSA=0; BYPASSB=0; rst=0;
expected_OUT=6; expected_LEDS=0; // out>>1
#2


A=1; B=1; OPCODE=5; CIN=1; SERIAL_IN=0; DIRECTION=1; RED_OP_A=0; RED_OP_B=0; BYPASSA=0; BYPASSB=0; rst=0;
expected_OUT=6'b001100; expected_LEDS=0; // out Rot left 1
```



```
A=1; B=1; OPCODE=5; CIN=1; SERIAL_IN=0; DIRECTION=0; RED_OP_A=0; RED_OP_B=0; BYPASSA=0; BYPASSB=0; rst=0;
expected_OUT=6; expected_LEDS=0; // out Rot Right 1


A=1; B=1; OPCODE=2; CIN=1; SERIAL_IN=0; DIRECTION=0; RED_OP_A=0; RED_OP_B=0; BYPASSA=0; BYPASSB=0; rst=0;
expected_OUT=3; expected_LEDS=0; // a+b+cin


end

endmodule
```

3-Wave Form


Wave - Default
Msgs
/ALSU_tb/A -No Data-
/ALSU_tb/B -No Data-
/ALSU_tb/OPCODE -No Data-
/ALSU_tb/RED_OP_A -No Data-
/ALSU_tb/RED_OP_B -No Data-
/ALSU_tb/SERIAL_IN -No Data-
/ALSU_tb/DIRECTION -No Data-
/ALSU_tb/BYPASSA -No Data-
/ALSU_tb/BYPASSB -No Data-
/ALSU_tb/rst -No Data-
/ALSU_tb/clk -No Data-
/ALSU_tb/CIN -No Data-
/ALSU_tb/expected... -No Data-
/ALSU_tb/expected... -No Data-
/ALSU_tb/OUT -No Data-
/ALSU_tb/LEDS -No Data-
1 0 3 1
1 2 7 1 2 1
2 6 7 3 0 1 3 4 5 2
00 01 02 00 01 00 01 00 01 00 01 06 0c 06 0c 03
ffff 0000
00 01 02 00 01 00 01 00 01 00 01 06 0c 06 0c 03
0000 ffff 0000
Now 100 ns
Cursor 1 236 ns
ns 4 ns 8 ns 12 ns 16 ns 20 ns 24 ns 28 ns 32 ns 36 ns 40