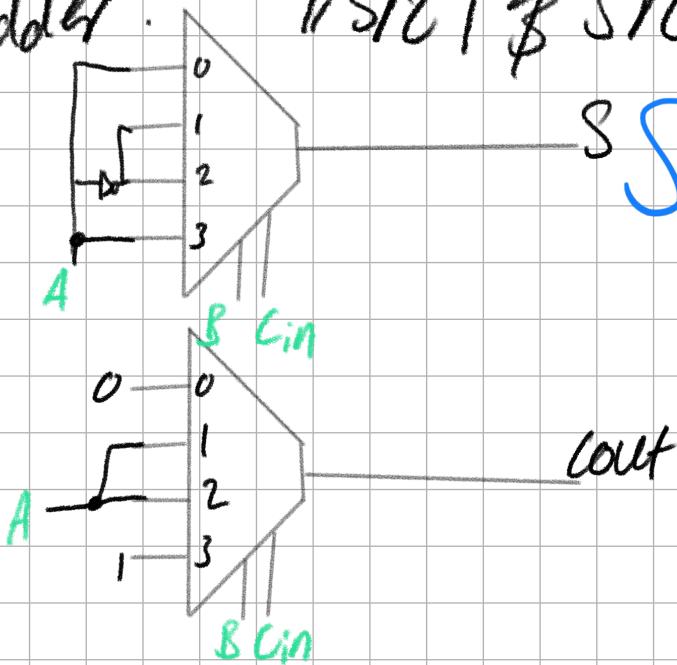


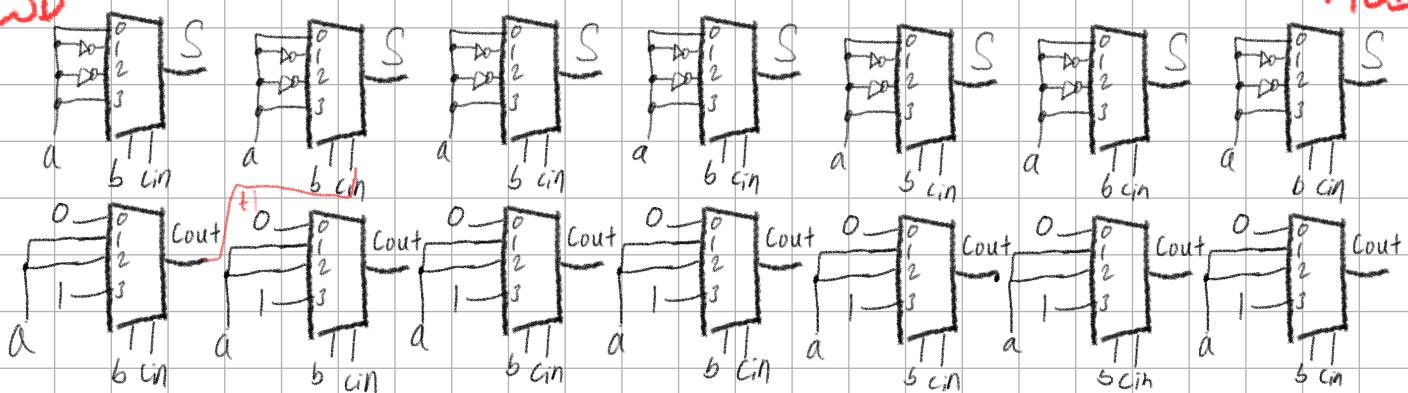
$$0000100 + 01$$

Full Adder: //SRC1 \$ SRC2 Lab



S SUPPLEMENTARY

FA1 FA2 FA3 FA4 FA5 FA6 FA7 MSB  
LSB



$$11111100 \text{ } 0 \text{ } \text{Cin}$$

$$\begin{array}{r}
 + 00000011 \\
 \hline
 \end{array}$$

0 0 0 0 0 0 1 1  
 b7 b6 b5 b4 b3 b2 b1 b0

$$\begin{array}{r}
 \hline
 11111101010 \\
 \hline
 S7 S6 S5 S4 S3 S2 S1 S0
 \end{array}$$

# 7 Seg Disp Converter ~~A~~ FROM lab 2

$n_3$	$n_2$	$n_1$	$n_0$	A	B	C	D	E	F	G
0	0	0	0	1	1	1	1	1	1	0
1	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0
3	0	0	1	1	1	1	1	0	0	1
4	0	1	0	0	1	1	0	0	1	1
5	0	1	0	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1
7	0	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1
9	1	0	0	1	1	1	0	0	1	1
a	1	0	1	0	1	1	1	0	1	1
b	1	0	1	1	0	0	1	1	1	1
c	1	1	0	0	1	0	0	1	1	0
d	1	1	0	1	0	1	1	1	0	1
e	1	1	1	0	1	0	0	1	1	1
f	1	1	1	1	0	0	0	1	1	1

15β2

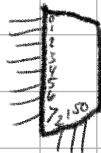
# hex>seg Mod CLK



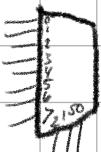
A



B



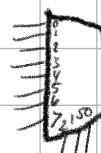
C



D



E



F



G

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

module m4_1( //m4_1(.in[3:0], .sel[1:0],
.out)
    input [3:0] in,
    input [1:0] sel,
    output out
);
    assign out = (!sel[0] & !sel[1]) ?
in[0] : //00
                (!sel[0] & sel[1]) ?
in[1] : //01
                (sel[0] & !sel[1]) ?
in[2] : //10
                in[3]
;
endmodule
```

```
`timescale 1ns / 1ps
///////////////////////////////
/////////////////////////////
// Company:
// Engineer:
//coming in is a 4 bit number to display
as a hex value
module hex7seg(//hex7seg(n [3:0]. seg
[7:0])
    input [3:0] n,
    output [6:0] seg //seg[0] = a, seg[1]
= b...etc
);
//m8_1(in [7:0], [2:0] sel, [7:0] out)
wire [7:0] inputs0, inputs1, inputs2,
inputs3, inputs4, inputs5, inputs6;
wire [2:0] selectors;
assign selectors[0] = n[0],
selectors[1] = n[1], selectors[2] = n[2];
wire W; assign W = n[3];

assign inputs0[0] = 1, inputs0[1] = 1,
inputs0[2] = !W, inputs0[3] = W,
inputs0[4] = 1, inputs0[5] = !W,
```

```
inputs0[6] = W, inputs0[7] = 0; //my: B,  
vid: A
```

```
    m8_1 mux0 (.in(inputs0),  
.sel(selectors), .out(seg[1]));
```

```
assign inputs1[0] = 1, inputs1[1] = W,  
inputs1[2] = 1, inputs1[3] = 1, inputs1[4]  
= 1, inputs1[5] = 1, inputs1[6] = W,  
inputs1[7] = 0; //my C, vid: B
```

```
    m8_1 mux1 (.in(inputs1),  
.sel(selectors), .out(seg[2]));
```

```
assign inputs2[0] = !W, inputs2[1] =  
1, inputs2[2] = W, inputs2[3] = !W,  
inputs2[4] = 1, inputs2[5] = W, inputs2[6]  
= 1, inputs2[7] = !W; //my D, Vid: C
```

```
    m8_1 mux2 (.in(inputs2),  
.sel(selectors), .out(seg[3]));
```

```
assign inputs3[0] = !W, inputs3[1] =  
!W, inputs3[2] = 0, inputs3[3] = !W,  
inputs3[4] = !W, inputs3[5] = 1,  
inputs3[6] = 1, inputs3[7] = 1; //my E,  
vid: D
```

```
m8_1 mux3 (.in(inputs3),
.sel(selectors), .out(seg[4]));

assign inputs4[0] = !W, inputs4[1] =
0, inputs4[2] = 1, inputs4[3] = !W,
inputs4[4] = 1, inputs4[5] = 1, inputs4[6]
= !W, inputs4[7] = 1; //my F, vid: E

m8_1 mux4 (.in(inputs4),
.sel(selectors), .out(seg[5]));

assign inputs5[0] = !W, inputs5[1] = 1,
inputs5[2] = W, inputs5[3] = 1, inputs5[4]
= 1, inputs5[5] = !W, inputs5[6] = !W,
inputs5[7] = 1; //my A, vid: F

m8_1 mux5 (.in(inputs5),
.sel(selectors), .out(seg[0]));

assign inputs6[0] = 0, inputs6[1] = 1,
inputs6[2] = 1, inputs6[3] = !W,
inputs6[4] = 1, inputs6[5] = 1, inputs6[6]
= W, inputs6[7] = 1; //my G, vid: G

m8_1 mux6 (.in(inputs6),
.sel(selectors), .out(seg[6]));
endmodule
```

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

`timescale 1ps/1ps

module clk_wiz_0

// Clock in ports
// Clock out ports
output      clk_out1,
// Status and control signals
input       reset,
output      locked,
input       clk_in1
);

// Input buffering
//-----
wire clk_in1_clk_wiz_0;
wire clk_in2_clk_wiz_0;
IBUF clkin1_ibufg
(.O (clk_in1_clk_wiz_0),
```

```
.I (clk_in1));  
  
// Clocking PRIMITIVE  
//-----  
  
// Instantiation of the MMCM PRIMITIVE  
//      * Unused inputs are tied off  
//      * Unused outputs are labeled unused  
  
wire          clk_out1_clk_wiz_0;  
wire          clk_out2_clk_wiz_0;  
wire          clk_out3_clk_wiz_0;  
wire          clk_out4_clk_wiz_0;  
wire          clk_out5_clk_wiz_0;  
wire          clk_out6_clk_wiz_0;  
wire          clk_out7_clk_wiz_0;  
  
wire [15:0] do_unused;  
wire          drdy_unused;  
wire          psdone_unused;  
wire          locked_int;  
wire          clkfbout_clk_wiz_0;  
wire          clkfbout_buf_clk_wiz_0;
```

```
wire clkfboutb_unused;
wire clkout0b_unused;
wire clkout1_unused;
wire clkout1b_unused;
wire clkout2_unused;
wire clkout2b_unused;
wire clkout3_unused;
wire clkout3b_unused;
wire clkout4_unused;
wire clkout5_unused;
wire clkout6_unused;
wire clkfbstopped_unused;
wire clkinstopped_unused;
wire reset_high;
```

## MMCME2\_ADV

```
# (.BANDWIDTH      ("OPTIMIZED") ,
  .CLKOUT4 CASCADE ("FALSE") ,
  .COMPENSATION    ("ZHOLD") ,
  .STARTUP_WAIT   ("FALSE") ,
  .DIVCLK_DIVIDE  (1) ,
  .CLKFBOUT_MULT_F (9.125) ,
  .CLKFBOUT_PHASE (0.000) ,
  .CLKFBOUT_USE_FINE_PS ("FALSE") ,
```

```
.CLKOUT0_DIVIDE_F      (36.500),
.CLKOUT0_PHASE         (0.000),
.CLKOUT0_DUTY_CYCLE    (0.500),
.CLKOUT0_USE_FINE_PS   ("FALSE"),
.CLKIN1_PERIOD          (10.0))

mmcm_adv_inst
// Output clocks
(
    .CLKFBOUT
    (clkfbout_clk_wiz_0),
    .CLKFBOUTB
    (clkfboutb_unused),
    .CLKOUT0
    (clk_out1_clk_wiz_0),
    .CLKOUT0B             (clkout0b_unused),
    .CLKOUT1              (clkout1_unused),
    .CLKOUT1B             (clkout1b_unused),
    .CLKOUT2              (clkout2_unused),
    .CLKOUT2B             (clkout2b_unused),
    .CLKOUT3              (clkout3_unused),
    .CLKOUT3B             (clkout3b_unused),
    .CLKOUT4              (clkout4_unused),
    .CLKOUT5              (clkout5_unused),
    .CLKOUT6              (clkout6_unused),
```

```
// Input clock control
.CLKFBIN
(clkfbout_buf_clk_wiz_0),
.CLKIN1
(clk_in1_clk_wiz_0),
.CLKIN2          (1'b0),
// Tied to always select the primary
input clock
.CLKINSEL        (1'b1),
// Ports for dynamic reconfiguration
.DADDR          (7'h0),
.DCLK            (1'b0),
.DEN             (1'b0),
.DI              (16'h0),
.DO              (do_unused),
.DRDY            (drdy_unused),
.DWE             (1'b0),
// Ports for dynamic phase shift
.PSCLK           (1'b0),
.PSEN             (1'b0),
.PSINCDEC        (1'b0),
.PSDONE          (psdone_unused),
// Other control and status signals
.LOCKED          (locked_int),
```

```
.CLKINSTOPPED
(clkinstopped_unused),
.CLKFBSTOPPED
(clkfbstopped_unused),
.PWRDWN          (1'b0),
.RST            (reset_high));
assign reset_high = reset;

assign locked = locked_int;
// Clock Monitor clock assigning
//-----
// Output buffering
//-----
```

```
BUFG clkf_buf
(.O (clkfbout_buf_clk_wiz_0),
.I (clkfbout_clk_wiz_0));
```

```
BUFG clkout1_buf
(.O  (clk_out1),
.I  (clk_out1_clk_wiz_0));
```

```
endmodule
```

```
module clkcntrl4(clkin,  
                  //clkb2,  
                  seldig);
```

```
    input clkin;  
    // output clkb2;  
    output seldig;
```

```
//wire XLXN_38;  
//wire XLXN_39;  
wire XLXN_44;  
wire XLXN_47;  
wire XLXN_70;  
wire XLXN_71;  
wire XLXN_72;  
wire XLXN_73;  
wire XLXN_74;  
wire XLXN_75;  
wire XLXN_76;
```

```
wire clk2_DUMMY;  
  
GND    XLXI_24  (.G(XLXN_44));  
  
(* HU_SET = "XLXI_37_73" *)  
CB4CE_MXILINX_clkctrl4  XLXI_37  
(.C(clk2_DUMMY),  
 .CE(XLXN_73),  
 .CLR(XLXN_76),  
 .CEO(XLXN_72),  
           .Q0(),  
           .Q1(),  
 .Q2(XLXN_74),  
           .Q3(),  
           .TC());  
  
(* HU_SET = "XLXI_38_74" *)  
CB4CE_MXILINX_clkctrl4  XLXI_38  
(.C(clk2_DUMMY),  
 .CE(XLXN_72),
```

```
.CLR(XLXN_76),  
  
.CEO(XLXN_70),  
    .Q0(),  
    .Q1(),  
    .Q2(),  
    .Q3(),  
    .TC());  
  
(* HU_SET = "XLXI_39_75" *)  
CB4CE_MXILINX_clkctrl4  XLXI_39  
(.C(clkb2_DUMMY),  
  
.CE(XLXN_70),  
  
.CLR(XLXN_76),  
  
.CEO(XLXN_71),  
    .Q0(),  
    .Q1(),  
    .Q2(),  
    .Q3(),  
    .TC());  
  
(* HU_SET = "XLXI_40_76" *)
```

```
CB4CE_MXILINX_clkctrl4  XLXI_40
(.C(clkb2_DUMMY) ,
.CE(XLXN_71) ,
.CLR(XLXN_76) ,
.CEO() ,
.Q0(XLXN_75) ,
.Q1() ,
.Q2() ,
.Q3() ,
.TC() ) ;
VCC  XLXI_41 (.P(XLXN_73)) ;
GND  XLXI_43 (.G(XLXN_76)) ;
BUF  XLXI_328 (.I(clkin) ,
.O(clkb2_DUMMY)) ;
`ifdef XILINX_SIMULATOR
BUF  XLXI_336 (.I(XLXN_74) ,
`else   BUF  XLXI_336 (.I(XLXN_75) ,
`endif
.O(seldig)) ;
endmodule
```

```
module lab3_digsel (
    input clkin,
    input greset, //btnR
    output digsel);

    clk_wiz_0 my_clk_inst
(.clk_out1(clk), .reset(greset),
.locked(), .clk_in1(clkin));

    clkcntrl4 slowit (.clkin(clk),
.seldig(digsel));

    STARTUPE2 #(.PROG_USR("FALSE"), //
Activate program event security feature.
Requires encrypted bitstreams.

        .SIM_CCLK_FREQ(0.0)
// Set the Configuration Clock
Frequency(ns) for simulation.

)

    STARTUPE2_inst (.CFGCLK(),
// 1-bit output: Configuration main clock
```

```
output .CFGMCLK() ,  
// 1-bit output: Configuration internal oscillator clock output .EOS() ,  
// 1-bit output: Active high output signal indicating the End Of Startup. .PREQ() , // 1-bit output: PROGRAM request to fabric output .CLK() , // 1-bit input: User start-up clock input .GSR(greset) , // 1-bit input: Global Set/Reset input (GSR cannot be used for the port name) .GTS() , // 1-bit input: Global 3-state input (GTS cannot be used for the port name) .KEYCLEARB() , // 1-bit input: Clear AES Decrypter Key input from Battery-Backed RAM (BBRAM) .PACK() , //
```

```
1-bit input: PROGRAM acknowledge input
                .USRCCCLKO(),
// 1-bit input: User CCLK input

                .USRCCCLKTS(), // 1-bit input: User CCLK
3-state enable input

                .USRDONEO(), // 1-bit input: User DONE pin
output control

                .USRDONETS() // 1-bit input: User DONE
3-state enable output
                ); // End of
STARTUPE2_inst instantiation
endmodule
```

```
module FTCE_MXILINX_clkcntrl4(C,  
                                CE,  
                                CLR,  
                                T,  
                                Q);
```

```
parameter INIT = 1'b0;

input C;
input CE;
input CLR;
input T;
output Q;

wire TQ;
wire Q_DUMMY;

assign Q = Q_DUMMY;
XOR2 I_36_32 (.I0(T),
               .I1(Q_DUMMY),
               .O(TQ));
/// (* RLOC = "X0Y0" *)
FDCE I_36_35 (.C(C),
               .CE(CE),
               .CLR(CLR),
               .D(TQ),
               .Q(Q_DUMMY));
endmodule
`timescale 1ns / 1ps
```

```
module CB4CE_MXILINX_clkcntrl4(C,
                                  CE,
                                  CLR,
                                  CEO,
                                  Q0,
                                  Q1,
                                  Q2,
                                  Q3,
                                  TC) ;
```

```
input C;
input CE;
input CLR;
output CEO;
output Q0;
output Q1;
output Q2;
output Q3;
output TC;
```

```
wire T2;
wire T3;
wire XLXN_1;
```

```
wire Q0_DUMMY;
wire Q1_DUMMY;
wire Q2_DUMMY;
wire Q3_DUMMY;
wire TC_DUMMY;

assign Q0 = Q0_DUMMY;
assign Q1 = Q1_DUMMY;
assign Q2 = Q2_DUMMY;
assign Q3 = Q3_DUMMY;
assign TC = TC_DUMMY;
(* HU_SET = "I_Q0_69" *)
FTCE_MXILINX_clkctrl4 #(.INIT(1'b0))
I_Q0 (.C(C),
       .CE(CE),
       .CLR(CLR),
       .T(XLXN_1),
       .Q(Q0_DUMMY));
(* HU_SET = "I_Q1_70" *)
FTCE_MXILINX_clkctrl4 #(.INIT(1'b0))
I_Q1 (.C(C),
       .CE(CE),
```

```
.T(Q0_DUMMY) ,
.Q(Q1_DUMMY) ;
(* HU_SET = "I_Q2_71" *)
FTCE_MXILINX_clkcntrl4 #( .INIT(1'b0) )
I_Q2 (.C(C),
       .CE(CE),
       .CLR(CLR),
       .T(T2),
.Q(Q2_DUMMY) ;
(* HU_SET = "I_Q3_72" *)
FTCE_MXILINX_clkcntrl4 #( .INIT(1'b0) )
I_Q3 (.C(C),
       .CE(CE),
       .CLR(CLR),
       .T(T3),
.Q(Q3_DUMMY) ;
AND4 I_36_31 (.I0(Q3_DUMMY),
               .I1(Q2_DUMMY),
               .I2(Q1_DUMMY),
               .O(Q3_DUMMY))
```

```
        .I3 (Q0_DUMMY) ,  
        .O (TC_DUMMY) ) ;  
  
AND3    I_36_32  (.I0 (Q2_DUMMY) ,  
                    .I1 (Q1_DUMMY) ,  
                    .I2 (Q0_DUMMY) ,  
                    .O (T3) ) ;  
  
AND2    I_36_33  (.I0 (Q1_DUMMY) ,  
                    .I1 (Q0_DUMMY) ,  
                    .O (T2) ) ;  
  
VCC     I_36_58  (.P (XLXN_1) ) ;  
  
AND2    I_36_67  (.I0 (CE) ,  
                    .I1 (TC_DUMMY) ,  
                    .O (CEO) ) ;  
  
endmodule
```

```
`timescale 1ns / 1ps
///////////////////////////////
/////////////////////////////
// Company:
// Engineer:
//
module m2_1x8( //m2_1x8(.in0 [7:0], .in1
[7:0], .sel, out [7:0])
    input [7:0] in0,
    input [7:0] in1,
    input sel,
    output [7:0] out
);
    assign out = sel ? in1 : in0; //if
sel=1, then out = in1, else in0
endmodule
```

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

module m8_1(      //m8_1(in [7:0], [2:0]
sel, [7:0] out)
    input [7:0] in,
    input [2:0] sel,
    output out
);
    assign out = (!sel[0] & !sel[1] &
!sel[2]) ? !in[0] :          //000
                (!sel[0] & !sel[1] &
sel[2]) ? !in[1] :          //001
                (!sel[0] & sel[1] &
!sel[2]) ? !in[2] :          //010
                (!sel[0] & sel[1] &
sel[2]) ? !in[3] :          //011
                (sel[0] & !sel[1] &
!sel[2]) ? !in[4] :          //100
                (sel[0] & !sel[1] &
```

```
sel[2]) ? !in[5] :          //101
           (sel[0] & sel[1] &
!sel[2]) ? !in[6] :          //110
           !in[7]
;
           //111
endmodule
```

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

module Incrementer( //Incrementer(a [7:0],
[1:0] b, [7:0] s)
    input [7:0] a,
    input [1:0] b,
    output [7:0] s
);
    //Full_Adder(cin, [1:0] selct, s, cout)
    wire [7:0] b_convert;
    assign b_convert[7:2] = 6'b000000;
    assign b_convert[0] = b[1];
    assign b_convert[1] = b[0]; //00000b[
    wire [7:0] w;
    wire [1:0] t0, t1, t2, t3, t4, t5, t6,
t7;
    assign t0[0] = b_convert[0], t0[1] = 0;
    Full_Adder fa1 (.cin(a[0]),
    .cout(s),
    .t1(t1),
    .t2(t2),
    .t3(t3),
    .t4(t4),
    .t5(t5),
    .t6(t6),
    .t7(t7));
endmodule
```

```
.select(t0), .s(s[7]), .cout(w[0]));  
  
assign t1[0] = b_convert[1], t1[1] =  
w[0];  
Full_Adder fa2 (.cin(a[1]),  
.select(t1), .s(s[6]), .cout(w[1]));  
  
assign t2[0] = b_convert[2], t2[1] =  
w[1];  
Full_Adder fa3 (.cin(a[2]),  
.select(t2), .s(s[5]), .cout(w[2]));  
  
assign t3[0] = b_convert[3], t3[1] =  
w[2];  
Full_Adder fa4 (.cin(a[3]),  
.select(t3), .s(s[4]), .cout(w[3]));  
  
assign t4[0] = b_convert[4], t4[1] =  
w[3];  
Full_Adder fa5 (.cin(a[4]),  
.select(t4), .s(s[3]), .cout(w[4]));  
  
assign t5[0] = b_convert[5], t5[1] =  
w[4];
```

```
Full_Adder fa6 (.cin(a[5]),  
.select(t5), .s(s[2]), .cout(w[5]));  
  
assign t6[0] = b_convert[6], t6[1] =  
w[5];  
Full_Adder fa7 (.cin(a[6]),  
.select(t6), .s(s[1]), .cout(w[6]));  
  
assign t7[0] = b_convert[7], t7[1] =  
w[6];  
Full_Adder fa8 (.cin(a[7]),  
.select(t7), .s(s[0]), .cout(w[7]));  
endmodule
```

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

module Top_Module_Main(
    input [7:0] sw,
    input btnL,
    input btnC,
    input btnR,
    input clkin,
    output [6:0] seg,
    output dp,
    output [3:0] an
);
    wire [1:0] button;
    wire [7:0] sum_output;
    wire [6:0] left_disp, right_disp;
    assign button[0] = btnL, button[1] =
btnC, an[3] = 1, an[2] = 1, dp = 1;

    //Incrementer(a [7:0], [1:0] b, [7:0]
```

```
s)
    Incrementer inc (.a(sw), .b(button),
.s(sum_output)); //sum_output bus 4 to
hex7seg1 and hex7seg2

hex7seg sgmt_cnvtr1
(.n(sum_output[3:0]), .seg(left_disp));
//left display

hex7seg sgmt_cnvtr2
(.n(sum_output[7:4]), .seg(right_disp));
//right display

wire dig_sel;
lab3_digsel lab3_digselect
(.clkin(clkin), .greset(btnR),
.digsel(dig_sel));
assign an[0] = dig_sel, an[1] =
!dig_sel;
//m2_1x8(.in0 [7:0], .in1 [7:0], .sel,
out [7:0])
m2_1x8 mux8_1(.in0(right_disp),
.in1(left_disp), .sel(dig_sel),
.out(seg)); //seg is the output of the
display
```