​​<https://github.com/bencorn/Nexys3-Kitchen-Timer/blob/master/fast_clock.v>

`timescale 1ns / 1ps

module tb\_random;

reg clk;

reg reset;

reg gameover;

wire [4:0] rand;

random ra(

.clk(clk),

.reset(reset),

.rand(rand),

.gameover(gameover)

);

initial begin

gameover = 0;

clk =0;

reset = 1;

#200;

end

always begin

#5 clk =~clk;

end

endmodule

timescale 1ns / 1ps

module debouncer(

input clk,

input reset,

input bin,

output reg bout

);

reg [15:0] deb\_count;

reg output\_exist;

initial begin

bout = 0;

output\_exist = 0;

deb\_count = 16'b0000000000000000;

end

always @ (posedge clk) begin

bout <= 1'b0;

if(~reset) begin

output\_exist <= 1'b0;

deb\_count <= 16'b0000000000000000;

end

else if (bin && (~output\_exist)) begin

deb\_count <= deb\_count + 1'b1;

if(deb\_count == 16'b1111111111111111)begin

bout <= 1'b1;

output\_exist <= 1'b1;

deb\_count <= 16'b0000000000000000;

end

end

end

endmodule

## This file is a general .xdc for the Nexys4 DDR Rev. C

## To use it in a project:

## - uncomment the lines corresponding to used pins

## - rename the used ports (in each line, after get\_ports) according to the top level signal names in the project

## Clock signal

set\_property -dict { PACKAGE\_PIN E3 IOSTANDARD LVCMOS33 } [get\_ports { clk }]; #IO\_L12P\_T1\_MRCC\_35 Sch=clk100mhz

#create\_clock -add -name sys\_clk\_pin -period 10.00 -waveform {0 5} [get\_ports {CLK100MHZ}];

#set\_property SEVERITY {Warning} [get\_drc\_checks NSTD -1];

#set\_property SEVERITY {Warning} [get\_drc\_checks LUTLP-1];

#set\_property ALLOW\_COMBINATORIAL\_LOOPS true;

##Switches

#set\_property -dict { PACKAGE\_PIN J15 IOSTANDARD LVCMOS33 } [get\_ports { swt[0] }]; #IO\_L24N\_T3\_RS0\_15 Sch=sw[0]

#set\_property -dict { PACKAGE\_PIN L16 IOSTANDARD LVCMOS33 } [get\_ports { swt[1] }]; #IO\_L3N\_T0\_DQS\_EMCCLK\_14 Sch=sw[1]

#set\_property -dict { PACKAGE\_PIN M13 IOSTANDARD LVCMOS33 } [get\_ports { swt[2] }]; #IO\_L6N\_T0\_D08\_VREF\_14 Sch=sw[2]

#set\_property -dict { PACKAGE\_PIN R15 IOSTANDARD LVCMOS33 } [get\_ports { swt[3] }]; #IO\_L13N\_T2\_MRCC\_14 Sch=sw[3]

#set\_property -dict { PACKAGE\_PIN R17 IOSTANDARD LVCMOS33 } [get\_ports { swt[4] }]; #IO\_L12N\_T1\_MRCC\_14 Sch=sw[4]

#set\_property -dict { PACKAGE\_PIN T18 IOSTANDARD LVCMOS33 } [get\_ports { swt[5] }]; #IO\_L7N\_T1\_D10\_14 Sch=sw[5]

#set\_property -dict { PACKAGE\_PIN U18 IOSTANDARD LVCMOS33 } [get\_ports { swt[6] }]; #IO\_L17N\_T2\_A13\_D29\_14 Sch=sw[6]

#set\_property -dict { PACKAGE\_PIN R13 IOSTANDARD LVCMOS33 } [get\_ports { swt[7] }]; #IO\_L5N\_T0\_D07\_14 Sch=sw[7]

#set\_property -dict { PACKAGE\_PIN T8 IOSTANDARD LVCMOS18 } [get\_ports { SW[8] }]; #IO\_L24N\_T3\_34 Sch=sw[8]

#set\_property -dict { PACKAGE\_PIN U8 IOSTANDARD LVCMOS18 } [get\_ports { SW[9] }]; #IO\_25\_34 Sch=sw[9]

#set\_property -dict { PACKAGE\_PIN R16 IOSTANDARD LVCMOS33 } [get\_ports { SW[10] }]; #IO\_L15P\_T2\_DQS\_RDWR\_B\_14 Sch=sw[10]

#set\_property -dict { PACKAGE\_PIN T13 IOSTANDARD LVCMOS33 } [get\_ports { SW[11] }]; #IO\_L23P\_T3\_A03\_D19\_14 Sch=sw[11]

#set\_property -dict { PACKAGE\_PIN H6 IOSTANDARD LVCMOS33 } [get\_ports { SW[12] }]; #IO\_L24P\_T3\_35 Sch=sw[12]

#set\_property -dict { PACKAGE\_PIN U12 IOSTANDARD LVCMOS33 } [get\_ports { SW[13] }]; #IO\_L20P\_T3\_A08\_D24\_14 Sch=sw[13]

#set\_property -dict { PACKAGE\_PIN U11 IOSTANDARD LVCMOS33 } [get\_ports { SW[14] }]; #IO\_L19N\_T3\_A09\_D25\_VREF\_14 Sch=sw[14]

#set\_property -dict { PACKAGE\_PIN V10 IOSTANDARD LVCMOS33 } [get\_ports { SW[15] }]; #IO\_L21P\_T3\_DQS\_14 Sch=sw[15]

## LEDs

set\_property -dict { PACKAGE\_PIN H17 IOSTANDARD LVCMOS33 } [get\_ports { led[0] }]; #IO\_L18P\_T2\_A24\_15 Sch=led[0]

set\_property -dict { PACKAGE\_PIN K15 IOSTANDARD LVCMOS33 } [get\_ports { led[1] }]; #IO\_L24P\_T3\_RS1\_15 Sch=led[1]

set\_property -dict { PACKAGE\_PIN J13 IOSTANDARD LVCMOS33 } [get\_ports { led[2] }]; #IO\_L17N\_T2\_A25\_15 Sch=led[2]

set\_property -dict { PACKAGE\_PIN N14 IOSTANDARD LVCMOS33 } [get\_ports { led[3] }]; #IO\_L8P\_T1\_D11\_14 Sch=led[3]

set\_property -dict { PACKAGE\_PIN R18 IOSTANDARD LVCMOS33 } [get\_ports { led[4] }]; #IO\_L7P\_T1\_D09\_14 Sch=led[4]

#set\_property -dict { PACKAGE\_PIN V17 IOSTANDARD LVCMOS33 } [get\_ports { led[5] }]; #IO\_L18N\_T2\_A11\_D27\_14 Sch=led[5]

#set\_property -dict { PACKAGE\_PIN U17 IOSTANDARD LVCMOS33 } [get\_ports { led[6] }]; #IO\_L17P\_T2\_A14\_D30\_14 Sch=led[6]

#set\_property -dict { PACKAGE\_PIN U16 IOSTANDARD LVCMOS33 } [get\_ports { led[7] }]; #IO\_L18P\_T2\_A12\_D28\_14 Sch=led[7]

#set\_property -dict { PACKAGE\_PIN V16 IOSTANDARD LVCMOS33 } [get\_ports { LED[8] }]; #IO\_L16N\_T2\_A15\_D31\_14 Sch=led[8]

#set\_property -dict { PACKAGE\_PIN T15 IOSTANDARD LVCMOS33 } [get\_ports { LED[9] }]; #IO\_L14N\_T2\_SRCC\_14 Sch=led[9]

#set\_property -dict { PACKAGE\_PIN U14 IOSTANDARD LVCMOS33 } [get\_ports { LED[10] }]; #IO\_L22P\_T3\_A05\_D21\_14 Sch=led[10]

#set\_property -dict { PACKAGE\_PIN T16 IOSTANDARD LVCMOS33 } [get\_ports { LED[11] }]; #IO\_L15N\_T2\_DQS\_DOUT\_CSO\_B\_14 Sch=led[11]

#set\_property -dict { PACKAGE\_PIN V15 IOSTANDARD LVCMOS33 } [get\_ports { LED[12] }]; #IO\_L16P\_T2\_CSI\_B\_14 Sch=led[12]

#set\_property -dict { PACKAGE\_PIN V14 IOSTANDARD LVCMOS33 } [get\_ports { LED[13] }]; #IO\_L22N\_T3\_A04\_D20\_14 Sch=led[13]

#set\_property -dict { PACKAGE\_PIN V12 IOSTANDARD LVCMOS33 } [get\_ports { LED[14] }]; #IO\_L20N\_T3\_A07\_D23\_14 Sch=led[14]

#set\_property -dict { PACKAGE\_PIN V11 IOSTANDARD LVCMOS33 } [get\_ports { LED[15] }]; #IO\_L21N\_T3\_DQS\_A06\_D22\_14 Sch=led[15]

#set\_property -dict { PACKAGE\_PIN R12 IOSTANDARD LVCMOS33 } [get\_ports { LED16\_B }]; #IO\_L5P\_T0\_D06\_14 Sch=led16\_b

#set\_property -dict { PACKAGE\_PIN M16 IOSTANDARD LVCMOS33 } [get\_ports { LED16\_G }]; #IO\_L10P\_T1\_D14\_14 Sch=led16\_g

#set\_property -dict { PACKAGE\_PIN N15 IOSTANDARD LVCMOS33 } [get\_ports { LED16\_R }]; #IO\_L11P\_T1\_SRCC\_14 Sch=led16\_r

#set\_property -dict { PACKAGE\_PIN G14 IOSTANDARD LVCMOS33 } [get\_ports { LED17\_B }]; #IO\_L15N\_T2\_DQS\_ADV\_B\_15 Sch=led17\_b

#set\_property -dict { PACKAGE\_PIN R11 IOSTANDARD LVCMOS33 } [get\_ports { LED17\_G }]; #IO\_0\_14 Sch=led17\_g

#set\_property -dict { PACKAGE\_PIN N16 IOSTANDARD LVCMOS33 } [get\_ports { LED17\_R }]; #IO\_L11N\_T1\_SRCC\_14 Sch=led17\_r

##7 segment display

set\_property -dict { PACKAGE\_PIN T10 IOSTANDARD LVCMOS33 } [get\_ports { display[0] }]; #IO\_L24N\_T3\_A00\_D16\_14 Sch=ca

set\_property -dict { PACKAGE\_PIN R10 IOSTANDARD LVCMOS33 } [get\_ports { display[1] }]; #IO\_25\_14 Sch=cb

set\_property -dict { PACKAGE\_PIN K16 IOSTANDARD LVCMOS33 } [get\_ports { display[2] }]; #IO\_25\_15 Sch=cc

set\_property -dict { PACKAGE\_PIN K13 IOSTANDARD LVCMOS33 } [get\_ports { display[3] }]; #IO\_L17P\_T2\_A26\_15 Sch=cd

set\_property -dict { PACKAGE\_PIN P15 IOSTANDARD LVCMOS33 } [get\_ports { display[4] }]; #IO\_L13P\_T2\_MRCC\_14 Sch=ce

set\_property -dict { PACKAGE\_PIN T11 IOSTANDARD LVCMOS33 } [get\_ports { display[5] }]; #IO\_L19P\_T3\_A10\_D26\_14 Sch=cf

set\_property -dict { PACKAGE\_PIN L18 IOSTANDARD LVCMOS33 } [get\_ports { display[6] }]; #IO\_L4P\_T0\_D04\_14 Sch=cg

#set\_property -dict { PACKAGE\_PIN H15 IOSTANDARD LVCMOS33 } [get\_ports { DP }]; #IO\_L19N\_T3\_A21\_VREF\_15 Sch=dp

set\_property -dict { PACKAGE\_PIN J17 IOSTANDARD LVCMOS33 } [get\_ports { AN[0] }]; #IO\_L23P\_T3\_FOE\_B\_15 Sch=an[0]

set\_property -dict { PACKAGE\_PIN J18 IOSTANDARD LVCMOS33 } [get\_ports { AN[1] }]; #IO\_L23N\_T3\_FWE\_B\_15 Sch=an[1]

set\_property -dict { PACKAGE\_PIN T9 IOSTANDARD LVCMOS33 } [get\_ports { AN[2] }]; #IO\_L24P\_T3\_A01\_D17\_14 Sch=an[2]

set\_property -dict { PACKAGE\_PIN J14 IOSTANDARD LVCMOS33 } [get\_ports { AN[3] }]; #IO\_L19P\_T3\_A22\_15 Sch=an[3]

set\_property -dict { PACKAGE\_PIN P14 IOSTANDARD LVCMOS33 } [get\_ports { AN[4] }]; #IO\_L8N\_T1\_D12\_14 Sch=an[4]

set\_property -dict { PACKAGE\_PIN T14 IOSTANDARD LVCMOS33 } [get\_ports { AN[5] }]; #IO\_L14P\_T2\_SRCC\_14 Sch=an[5]

#set\_property -dict { PACKAGE\_PIN K2 IOSTANDARD LVCMOS33 } [get\_ports { AN[6] }]; #IO\_L23P\_T3\_35 Sch=an[6]

#set\_property -dict { PACKAGE\_PIN U13 IOSTANDARD LVCMOS33 } [get\_ports { AN[7] }]; #IO\_L23N\_T3\_A02\_D18\_14 Sch=an[7]

##Buttons

set\_property -dict { PACKAGE\_PIN C12 IOSTANDARD LVCMOS33 } [get\_ports { reset }]; #IO\_L3P\_T0\_DQS\_AD1P\_15 Sch=cpu\_resetn

set\_property -dict { PACKAGE\_PIN N17 IOSTANDARD LVCMOS33 } [get\_ports { button[0] }]; #IO\_L9P\_T1\_DQS\_14 Sch=btnc

set\_property -dict { PACKAGE\_PIN M18 IOSTANDARD LVCMOS33 } [get\_ports { button[1] }]; #IO\_L4N\_T0\_D05\_14 Sch=btnu

set\_property -dict { PACKAGE\_PIN P17 IOSTANDARD LVCMOS33 } [get\_ports { button[2] }]; #IO\_L12P\_T1\_MRCC\_14 Sch=btnl

set\_property -dict { PACKAGE\_PIN M17 IOSTANDARD LVCMOS33 } [get\_ports { button[3] }]; #IO\_L10N\_T1\_D15\_14 Sch=btnr

set\_property -dict { PACKAGE\_PIN P18 IOSTANDARD LVCMOS33 } [get\_ports { button[4] }]; #IO\_L9N\_T1\_DQS\_D13\_14 Sch=btnd

##Pmod Headers

##Pmod Header JA

#set\_property -dict { PACKAGE\_PIN C17 IOSTANDARD LVCMOS33 } [get\_ports { JA[1] }]; #IO\_L20N\_T3\_A19\_15 Sch=ja[1]

#set\_property -dict { PACKAGE\_PIN D18 IOSTANDARD LVCMOS33 } [get\_ports { JA[2] }]; #IO\_L21N\_T3\_DQS\_A18\_15 Sch=ja[2]

#set\_property -dict { PACKAGE\_PIN E18 IOSTANDARD LVCMOS33 } [get\_ports { JA[3] }]; #IO\_L21P\_T3\_DQS\_15 Sch=ja[3]

#set\_property -dict { PACKAGE\_PIN G17 IOSTANDARD LVCMOS33 } [get\_ports { JA[4] }]; #IO\_L18N\_T2\_A23\_15 Sch=ja[4]

#set\_property -dict { PACKAGE\_PIN D17 IOSTANDARD LVCMOS33 } [get\_ports { JA[7] }]; #IO\_L16N\_T2\_A27\_15 Sch=ja[7]

#set\_property -dict { PACKAGE\_PIN E17 IOSTANDARD LVCMOS33 } [get\_ports { JA[8] }]; #IO\_L16P\_T2\_A28\_15 Sch=ja[8]

#set\_property -dict { PACKAGE\_PIN F18 IOSTANDARD LVCMOS33 } [get\_ports { JA[9] }]; #IO\_L22N\_T3\_A16\_15 Sch=ja[9]

#set\_property -dict { PACKAGE\_PIN G18 IOSTANDARD LVCMOS33 } [get\_ports { JA[10] }]; #IO\_L22P\_T3\_A17\_15 Sch=ja[10]

##Pmod Header JB

#set\_property -dict { PACKAGE\_PIN D14 IOSTANDARD LVCMOS33 } [get\_ports { JB[1] }]; #IO\_L1P\_T0\_AD0P\_15 Sch=jb[1]

#set\_property -dict { PACKAGE\_PIN F16 IOSTANDARD LVCMOS33 } [get\_ports { JB[2] }]; #IO\_L14N\_T2\_SRCC\_15 Sch=jb[2]

#set\_property -dict { PACKAGE\_PIN G16 IOSTANDARD LVCMOS33 } [get\_ports { JB[3] }]; #IO\_L13N\_T2\_MRCC\_15 Sch=jb[3]

#set\_property -dict { PACKAGE\_PIN H14 IOSTANDARD LVCMOS33 } [get\_ports { JB[4] }]; #IO\_L15P\_T2\_DQS\_15 Sch=jb[4]

#set\_property -dict { PACKAGE\_PIN E16 IOSTANDARD LVCMOS33 } [get\_ports { JB[7] }]; #IO\_L11N\_T1\_SRCC\_15 Sch=jb[7]

#set\_property -dict { PACKAGE\_PIN F13 IOSTANDARD LVCMOS33 } [get\_ports { JB[8] }]; #IO\_L5P\_T0\_AD9P\_15 Sch=jb[8]

#set\_property -dict { PACKAGE\_PIN G13 IOSTANDARD LVCMOS33 } [get\_ports { JB[9] }]; #IO\_0\_15 Sch=jb[9]

#set\_property -dict { PACKAGE\_PIN H16 IOSTANDARD LVCMOS33 } [get\_ports { JB[10] }]; #IO\_L13P\_T2\_MRCC\_15 Sch=jb[10]

##Pmod Header JC

#set\_property -dict { PACKAGE\_PIN K1 IOSTANDARD LVCMOS33 } [get\_ports { JC[1] }]; #IO\_L23N\_T3\_35 Sch=jc[1]

#set\_property -dict { PACKAGE\_PIN F6 IOSTANDARD LVCMOS33 } [get\_ports { JC[2] }]; #IO\_L19N\_T3\_VREF\_35 Sch=jc[2]

#set\_property -dict { PACKAGE\_PIN J2 IOSTANDARD LVCMOS33 } [get\_ports { JC[3] }]; #IO\_L22N\_T3\_35 Sch=jc[3]

#set\_property -dict { PACKAGE\_PIN G6 IOSTANDARD LVCMOS33 } [get\_ports { JC[4] }]; #IO\_L19P\_T3\_35 Sch=jc[4]

#set\_property -dict { PACKAGE\_PIN E7 IOSTANDARD LVCMOS33 } [get\_ports { JC[7] }]; #IO\_L6P\_T0\_35 Sch=jc[7]

#set\_property -dict { PACKAGE\_PIN J3 IOSTANDARD LVCMOS33 } [get\_ports { JC[8] }]; #IO\_L22P\_T3\_35 Sch=jc[8]

#set\_property -dict { PACKAGE\_PIN J4 IOSTANDARD LVCMOS33 } [get\_ports { JC[9] }]; #IO\_L21P\_T3\_DQS\_35 Sch=jc[9]

#set\_property -dict { PACKAGE\_PIN E6 IOSTANDARD LVCMOS33 } [get\_ports { JC[10] }]; #IO\_L5P\_T0\_AD13P\_35 Sch=jc[10]

##Pmod Header JD

#set\_property -dict { PACKAGE\_PIN H4 IOSTANDARD LVCMOS33 } [get\_ports { JD[1] }]; #IO\_L21N\_T3\_DQS\_35 Sch=jd[1]

#set\_property -dict { PACKAGE\_PIN H1 IOSTANDARD LVCMOS33 } [get\_ports { JD[2] }]; #IO\_L17P\_T2\_35 Sch=jd[2]

#set\_property -dict { PACKAGE\_PIN G1 IOSTANDARD LVCMOS33 } [get\_ports { JD[3] }]; #IO\_L17N\_T2\_35 Sch=jd[3]

#set\_property -dict { PACKAGE\_PIN G3 IOSTANDARD LVCMOS33 } [get\_ports { JD[4] }]; #IO\_L20N\_T3\_35 Sch=jd[4]

#set\_property -dict { PACKAGE\_PIN H2 IOSTANDARD LVCMOS33 } [get\_ports { JD[7] }]; #IO\_L15P\_T2\_DQS\_35 Sch=jd[7]

#set\_property -dict { PACKAGE\_PIN G4 IOSTANDARD LVCMOS33 } [get\_ports { JD[8] }]; #IO\_L20P\_T3\_35 Sch=jd[8]

#set\_property -dict { PACKAGE\_PIN G2 IOSTANDARD LVCMOS33 } [get\_ports { JD[9] }]; #IO\_L15N\_T2\_DQS\_35 Sch=jd[9]

#set\_property -dict { PACKAGE\_PIN F3 IOSTANDARD LVCMOS33 } [get\_ports { JD[10] }]; #IO\_L13N\_T2\_MRCC\_35 Sch=jd[10]

##Pmod Header JXADC

#set\_property -dict { PACKAGE\_PIN A14 IOSTANDARD LVDS } [get\_ports { XA\_N[1] }]; #IO\_L9N\_T1\_DQS\_AD3N\_15 Sch=xa\_n[1]

#set\_property -dict { PACKAGE\_PIN A13 IOSTANDARD LVDS } [get\_ports { XA\_P[1] }]; #IO\_L9P\_T1\_DQS\_AD3P\_15 Sch=xa\_p[1]

#set\_property -dict { PACKAGE\_PIN A16 IOSTANDARD LVDS } [get\_ports { XA\_N[2] }]; #IO\_L8N\_T1\_AD10N\_15 Sch=xa\_n[2]

#set\_property -dict { PACKAGE\_PIN A15 IOSTANDARD LVDS } [get\_ports { XA\_P[2] }]; #IO\_L8P\_T1\_AD10P\_15 Sch=xa\_p[2]

#set\_property -dict { PACKAGE\_PIN B17 IOSTANDARD LVDS } [get\_ports { XA\_N[3] }]; #IO\_L7N\_T1\_AD2N\_15 Sch=xa\_n[3]

#set\_property -dict { PACKAGE\_PIN B16 IOSTANDARD LVDS } [get\_ports { XA\_P[3] }]; #IO\_L7P\_T1\_AD2P\_15 Sch=xa\_p[3]

#set\_property -dict { PACKAGE\_PIN A18 IOSTANDARD LVDS } [get\_ports { XA\_N[4] }]; #IO\_L10N\_T1\_AD11N\_15 Sch=xa\_n[4]

#set\_property -dict { PACKAGE\_PIN B18 IOSTANDARD LVDS } [get\_ports { XA\_P[4] }]; #IO\_L10P\_T1\_AD11P\_15 Sch=xa\_p[4]

##VGA Connector

#set\_property -dict { PACKAGE\_PIN A3 IOSTANDARD LVCMOS33 } [get\_ports { VGA\_R[0] }]; #IO\_L8N\_T1\_AD14N\_35 Sch=vga\_r[0]

#set\_property -dict { PACKAGE\_PIN B4 IOSTANDARD LVCMOS33 } [get\_ports { VGA\_R[1] }]; #IO\_L7N\_T1\_AD6N\_35 Sch=vga\_r[1]

#set\_property -dict { PACKAGE\_PIN C5 IOSTANDARD LVCMOS33 } [get\_ports { VGA\_R[2] }]; #IO\_L1N\_T0\_AD4N\_35 Sch=vga\_r[2]

#set\_property -dict { PACKAGE\_PIN A4 IOSTANDARD LVCMOS33 } [get\_ports { VGA\_R[3] }]; #IO\_L8P\_T1\_AD14P\_35 Sch=vga\_r[3]

#set\_property -dict { PACKAGE\_PIN C6 IOSTANDARD LVCMOS33 } [get\_ports { VGA\_G[0] }]; #IO\_L1P\_T0\_AD4P\_35 Sch=vga\_g[0]

#set\_property -dict { PACKAGE\_PIN A5 IOSTANDARD LVCMOS33 } [get\_ports { VGA\_G[1] }]; #IO\_L3N\_T0\_DQS\_AD5N\_35 Sch=vga\_g[1]

#set\_property -dict { PACKAGE\_PIN B6 IOSTANDARD LVCMOS33 } [get\_ports { VGA\_G[2] }]; #IO\_L2N\_T0\_AD12N\_35 Sch=vga\_g[2]

#set\_property -dict { PACKAGE\_PIN A6 IOSTANDARD LVCMOS33 } [get\_ports { VGA\_G[3] }]; #IO\_L3P\_T0\_DQS\_AD5P\_35 Sch=vga\_g[3]

#set\_property -dict { PACKAGE\_PIN B7 IOSTANDARD LVCMOS33 } [get\_ports { VGA\_B[0] }]; #IO\_L2P\_T0\_AD12P\_35 Sch=vga\_b[0]

#set\_property -dict { PACKAGE\_PIN C7 IOSTANDARD LVCMOS33 } [get\_ports { VGA\_B[1] }]; #IO\_L4N\_T0\_35 Sch=vga\_b[1]

#set\_property -dict { PACKAGE\_PIN D7 IOSTANDARD LVCMOS33 } [get\_ports { VGA\_B[2] }]; #IO\_L6N\_T0\_VREF\_35 Sch=vga\_b[2]

#set\_property -dict { PACKAGE\_PIN D8 IOSTANDARD LVCMOS33 } [get\_ports { VGA\_B[3] }]; #IO\_L4P\_T0\_35 Sch=vga\_b[3]

#set\_property -dict { PACKAGE\_PIN B11 IOSTANDARD LVCMOS33 } [get\_ports { VGA\_HS }]; #IO\_L4P\_T0\_15 Sch=vga\_hs

#set\_property -dict { PACKAGE\_PIN B12 IOSTANDARD LVCMOS33 } [get\_ports { VGA\_VS }]; #IO\_L3N\_T0\_DQS\_AD1N\_15 Sch=vga\_vs

##Micro SD Connector

#set\_property -dict { PACKAGE\_PIN E2 IOSTANDARD LVCMOS33 } [get\_ports { SD\_RESET }]; #IO\_L14P\_T2\_SRCC\_35 Sch=sd\_reset

#set\_property -dict { PACKAGE\_PIN A1 IOSTANDARD LVCMOS33 } [get\_ports { SD\_CD }]; #IO\_L9N\_T1\_DQS\_AD7N\_35 Sch=sd\_cd

#set\_property -dict { PACKAGE\_PIN B1 IOSTANDARD LVCMOS33 } [get\_ports { SD\_SCK }]; #IO\_L9P\_T1\_DQS\_AD7P\_35 Sch=sd\_sck

#set\_property -dict { PACKAGE\_PIN C1 IOSTANDARD LVCMOS33 } [get\_ports { SD\_CMD }]; #IO\_L16N\_T2\_35 Sch=sd\_cmd

#set\_property -dict { PACKAGE\_PIN C2 IOSTANDARD LVCMOS33 } [get\_ports { SD\_DAT[0] }]; #IO\_L16P\_T2\_35 Sch=sd\_dat[0]

#set\_property -dict { PACKAGE\_PIN E1 IOSTANDARD LVCMOS33 } [get\_ports { SD\_DAT[1] }]; #IO\_L18N\_T2\_35 Sch=sd\_dat[1]

#set\_property -dict { PACKAGE\_PIN F1 IOSTANDARD LVCMOS33 } [get\_ports { SD\_DAT[2] }]; #IO\_L18P\_T2\_35 Sch=sd\_dat[2]

#set\_property -dict { PACKAGE\_PIN D2 IOSTANDARD LVCMOS33 } [get\_ports { SD\_DAT[3] }]; #IO\_L14N\_T2\_SRCC\_35 Sch=sd\_dat[3]

##Accelerometer

#set\_property -dict { PACKAGE\_PIN E15 IOSTANDARD LVCMOS33 } [get\_ports { ACL\_MISO }]; #IO\_L11P\_T1\_SRCC\_15 Sch=acl\_miso

#set\_property -dict { PACKAGE\_PIN F14 IOSTANDARD LVCMOS33 } [get\_ports { ACL\_MOSI }]; #IO\_L5N\_T0\_AD9N\_15 Sch=acl\_mosi

#set\_property -dict { PACKAGE\_PIN F15 IOSTANDARD LVCMOS33 } [get\_ports { ACL\_SCLK }]; #IO\_L14P\_T2\_SRCC\_15 Sch=acl\_sclk

#set\_property -dict { PACKAGE\_PIN D15 IOSTANDARD LVCMOS33 } [get\_ports { ACL\_CSN }]; #IO\_L12P\_T1\_MRCC\_15 Sch=acl\_csn

#set\_property -dict { PACKAGE\_PIN B13 IOSTANDARD LVCMOS33 } [get\_ports { ACL\_INT[1] }]; #IO\_L2P\_T0\_AD8P\_15 Sch=acl\_int[1]

#set\_property -dict { PACKAGE\_PIN C16 IOSTANDARD LVCMOS33 } [get\_ports { ACL\_INT[2] }]; #IO\_L20P\_T3\_A20\_15 Sch=acl\_int[2]

##Temperature Sensor

#set\_property -dict { PACKAGE\_PIN C14 IOSTANDARD LVCMOS33 } [get\_ports { TMP\_SCL }]; #IO\_L1N\_T0\_AD0N\_15 Sch=tmp\_scl

#set\_property -dict { PACKAGE\_PIN C15 IOSTANDARD LVCMOS33 } [get\_ports { TMP\_SDA }]; #IO\_L12N\_T1\_MRCC\_15 Sch=tmp\_sda

#set\_property -dict { PACKAGE\_PIN D13 IOSTANDARD LVCMOS33 } [get\_ports { TMP\_INT }]; #IO\_L6N\_T0\_VREF\_15 Sch=tmp\_int

#set\_property -dict { PACKAGE\_PIN B14 IOSTANDARD LVCMOS33 } [get\_ports { TMP\_CT }]; #IO\_L2N\_T0\_AD8N\_15 Sch=tmp\_ct

##Omnidirectional Microphone

#set\_property -dict { PACKAGE\_PIN J5 IOSTANDARD LVCMOS33 } [get\_ports { M\_CLK }]; #IO\_25\_35 Sch=m\_clk

#set\_property -dict { PACKAGE\_PIN H5 IOSTANDARD LVCMOS33 } [get\_ports { M\_DATA }]; #IO\_L24N\_T3\_35 Sch=m\_data

#set\_property -dict { PACKAGE\_PIN F5 IOSTANDARD LVCMOS33 } [get\_ports { M\_LRSEL }]; #IO\_0\_35 Sch=m\_lrsel

##PWM Audio Amplifier

#set\_property -dict { PACKAGE\_PIN A11 IOSTANDARD LVCMOS33 } [get\_ports { AUD\_PWM }]; #IO\_L4N\_T0\_15 Sch=aud\_pwm

#set\_property -dict { PACKAGE\_PIN D12 IOSTANDARD LVCMOS33 } [get\_ports { AUD\_SD }]; #IO\_L6P\_T0\_15 Sch=aud\_sd

##USB-RS232 Interface

#set\_property -dict { PACKAGE\_PIN C4 IOSTANDARD LVCMOS33 } [get\_ports { UART\_TXD\_IN }]; #IO\_L7P\_T1\_AD6P\_35 Sch=uart\_txd\_in

#set\_property -dict { PACKAGE\_PIN D4 IOSTANDARD LVCMOS33 } [get\_ports { UART\_RXD\_OUT }]; #IO\_L11N\_T1\_SRCC\_35 Sch=uart\_rxd\_out

#set\_property -dict { PACKAGE\_PIN D3 IOSTANDARD LVCMOS33 } [get\_ports { UART\_CTS }]; #IO\_L12N\_T1\_MRCC\_35 Sch=uart\_cts

#set\_property -dict { PACKAGE\_PIN E5 IOSTANDARD LVCMOS33 } [get\_ports { UART\_RTS }]; #IO\_L5N\_T0\_AD13N\_35 Sch=uart\_rts

##USB HID (PS/2)

#set\_property -dict { PACKAGE\_PIN F4 IOSTANDARD LVCMOS33 } [get\_ports { PS2\_CLK }]; #IO\_L13P\_T2\_MRCC\_35 Sch=ps2\_clk

#set\_property -dict { PACKAGE\_PIN B2 IOSTANDARD LVCMOS33 } [get\_ports { PS2\_DATA }]; #IO\_L10N\_T1\_AD15N\_35 Sch=ps2\_data

##SMSC Ethernet PHY

#set\_property -dict { PACKAGE\_PIN C9 IOSTANDARD LVCMOS33 } [get\_ports { ETH\_MDC }]; #IO\_L11P\_T1\_SRCC\_16 Sch=eth\_mdc

#set\_property -dict { PACKAGE\_PIN A9 IOSTANDARD LVCMOS33 } [get\_ports { ETH\_MDIO }]; #IO\_L14N\_T2\_SRCC\_16 Sch=eth\_mdio

#set\_property -dict { PACKAGE\_PIN B3 IOSTANDARD LVCMOS33 } [get\_ports { ETH\_RSTN }]; #IO\_L10P\_T1\_AD15P\_35 Sch=eth\_rstn

#set\_property -dict { PACKAGE\_PIN D9 IOSTANDARD LVCMOS33 } [get\_ports { ETH\_CRSDV }]; #IO\_L6N\_T0\_VREF\_16 Sch=eth\_crsdv

#set\_property -dict { PACKAGE\_PIN C10 IOSTANDARD LVCMOS33 } [get\_ports { ETH\_RXERR }]; #IO\_L13N\_T2\_MRCC\_16 Sch=eth\_rxerr

#set\_property -dict { PACKAGE\_PIN C11 IOSTANDARD LVCMOS33 } [get\_ports { ETH\_RXD[0] }]; #IO\_L13P\_T2\_MRCC\_16 Sch=eth\_rxd[0]

#set\_property -dict { PACKAGE\_PIN D10 IOSTANDARD LVCMOS33 } [get\_ports { ETH\_RXD[1] }]; #IO\_L19N\_T3\_VREF\_16 Sch=eth\_rxd[1]

#set\_property -dict { PACKAGE\_PIN B9 IOSTANDARD LVCMOS33 } [get\_ports { ETH\_TXEN }]; #IO\_L11N\_T1\_SRCC\_16 Sch=eth\_txen

#set\_property -dict { PACKAGE\_PIN A10 IOSTANDARD LVCMOS33 } [get\_ports { ETH\_TXD[0] }]; #IO\_L14P\_T2\_SRCC\_16 Sch=eth\_txd[0]

#set\_property -dict { PACKAGE\_PIN A8 IOSTANDARD LVCMOS33 } [get\_ports { ETH\_TXD[1] }]; #IO\_L12N\_T1\_MRCC\_16 Sch=eth\_txd[1]

#set\_property -dict { PACKAGE\_PIN D5 IOSTANDARD LVCMOS33 } [get\_ports { ETH\_REFCLK }]; #IO\_L11P\_T1\_SRCC\_35 Sch=eth\_refclk

#set\_property -dict { PACKAGE\_PIN B8 IOSTANDARD LVCMOS33 } [get\_ports { ETH\_INTN }]; #IO\_L12P\_T1\_MRCC\_16 Sch=eth\_intn

##Quad SPI Flash

#set\_property -dict { PACKAGE\_PIN K17 IOSTANDARD LVCMOS33 } [get\_ports { QSPI\_DQ[0] }]; #IO\_L1P\_T0\_D00\_MOSI\_14 Sch=qspi\_dq[0]

#set\_property -dict { PACKAGE\_PIN K18 IOSTANDARD LVCMOS33 } [get\_ports { QSPI\_DQ[1] }]; #IO\_L1N\_T0\_D01\_DIN\_14 Sch=qspi\_dq[1]

#set\_property -dict { PACKAGE\_PIN L14 IOSTANDARD LVCMOS33 } [get\_ports { QSPI\_DQ[2] }]; #IO\_L2P\_T0\_D02\_14 Sch=qspi\_dq[2]

#set\_property -dict { PACKAGE\_PIN M14 IOSTANDARD LVCMOS33 } [get\_ports { QSPI\_DQ[3] }]; #IO\_L2N\_T0\_D03\_14 Sch=qspi\_dq[3]

#set\_property -dict { PACKAGE\_PIN L13 IOSTANDARD LVCMOS33 } [get\_ports { QSPI\_CSN }]; #IO\_L6P\_T0\_FCS\_B\_14 Sch=qspi\_csn

`timescale 1ns / 1ps

module binary\_to\_bcd(

input clk,

input [7:0] eight\_bit\_value,

output reg[3:0] ones =0,

output reg[3:0] tens =0,

output reg[3:0] hundreds =0

);

reg[3:0] i = 0;

reg[19:0] shift\_register =0;

reg[3:0] temp\_hundreds =0;

reg[3:0] temp\_tens =0;

reg[3:0] temp\_ones =0;

reg[7:0] old\_eight\_bit\_value =0;

always@(posedge clk)

begin

if(i==0 & (old\_eight\_bit\_value !=eight\_bit\_value))

begin

shift\_register = 20'd0;

old\_eight\_bit\_value = eight\_bit\_value;

shift\_register[7:0] = eight\_bit\_value;

temp\_hundreds = shift\_register[19:16];

temp\_tens = shift\_register[15:12];

temp\_ones = shift\_register[11:8];

i=i+1;

end

if(i<9 & i>0) begin

if(temp\_hundreds >=5) temp\_hundreds = temp\_hundreds +3;

if(temp\_tens >=5) temp\_tens = temp\_tens+3;

if(temp\_ones >=5) temp\_ones = temp\_ones+3;

shift\_register [19:8] = {temp\_hundreds, temp\_tens, temp\_ones};

shift\_register = shift\_register <<1;

i = i+1;

end

if(i==9) begin

i=0;

hundreds = temp\_hundreds;

tens = temp\_tens;

ones = temp\_ones;

end

end

endmodule

`timescale 1ns / 1ps

module bcdconverter(

input [7:0] binary,

output [3:0] ones, tens,

output [1:0] hundreds

);

wire [3:0] c1,c2,c3,c4,c5,c6,c7;

wire [3:0] d1,d2,d3,d4,d5,d6,d7;

assign d1 = {1'b0,binary[7:5]};

assign d2 = {c1[2:0],binary[4]};

assign d3 = {c2[2:0],binary[3]};

assign d4 = {c3[2:0],binary[2]};

assign d5 = {c4[2:0],binary[1]};

assign d6 = {1'b0,c1[3],c2[3],c3[3]};

assign d7 = {c6[2:0],c4[3]};

add3 m1(d1,c1);

add3 m2(d2,c2);

add3 m3(d3,c3);

add3 m4(d4,c4);

add3 m5(d5,c5);

add3 m6(d6,c6);

add3 m7(d7,c7);

assign ones = {c5[2:0],binary[0]};

assign tens = {c7[2:0],c5[3]};

assign hundreds = {c6[3],c7[3]};

endmodule

`timescale 1ns / 1ps

module add3(in,out);

input [3:0] in;

output [3:0] out;

reg [3:0] out;

always @ (in)

case (in)

4'b0000: out <= 4'b0000;

4'b0001: out <= 4'b0001;

4'b0010: out <= 4'b0010;

4'b0011: out <= 4'b0011;

4'b0100: out <= 4'b0100;

4'b0101: out <= 4'b1000;

4'b0110: out <= 4'b1001;

4'b0111: out <= 4'b1010;

4'b1000: out <= 4'b1011;

4'b1001: out <= 4'b1100;

default: out <= 4'b0000;

endcase

endmodule

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 12/07/2021 09:39:54 PM

// Design Name:

// Module Name: bcd\_tb

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module bcd\_tb(

);

reg [7:0] binary;

reg clk;

wire [1:0] hundreds;

wire [3:0] tens;

wire [3:0] ones;

bcdconverter btcd(.binary(binary),.hundreds(hundreds),.tens(tens),.ones(ones));

initial begin

clk = 0;

binary = 8'b00010011;

#200;

end

always begin

#5 clk =~clk;

#20 binary = binary - 1;

end

endmodule

`timescale 1ns / 1ps

module top(

input clk,

input[4:0] button,

input reset,

output [6:0] display,

output [5:0] AN,

output [4:0] led

);

wire clk1hz, clk1khz, gameover;

wire [4:0] bout;

wire [4:0] timecount;

wire [5:0] score\_count;

wire [3:0] num;

wire [9:0] bcdtime;

wire [9:0] bcdscore;

slow\_clk clk1 (.in\_clk(clk), .out\_clk(clk1hz));

fast\_clk clk2 (.in\_clk(clk), .out\_clk(clk1khz));

debouncer deb(.bout(bout), .clk(clk), .reset(reset) , .bin(button));

random ran(.clk(clk1hz), .reset(reset), .gameover(gameover), .rand(led));

timer30 tim(.clk(clk1hz),.reset(reset),.count(timecount),.gameover(gameover));

score\_counter score(.clk(clk1hz), .reset(reset), .button(bout),.gameover(gameover), .ledin(led), .score\_count(score\_count));

bcdconverter bcd1(.binary({3'b000, timecount}), .hundreds(bcdtime[9:8]),.tens(bcdtime[7:4]),.ones(bcdtime[3:0]));

bcdconverter bcd2(.binary({2'b00,score\_count}),.hundreds(bcdscore[9:8]),.tens(bcdscore[7:4]),.ones(bcdscore[3:0]));

display\_control dispctrl(.anode(AN),.num(num), .onesa(bcdtime[3:0]),.onesb(bcdscore[3:0]),.tensa(bcdtime[7:4]),.tensb(bcdscore[7:4]),.clk(clk1khz),.reset(reset));

seven\_segment seg(.sev(display),.num(num));

endmodule

`timescale 1ns / 1ps

module score\_counter(

input clk,

input[4:0] button,

input reset,

input gameover,

input[4:0] ledin,

output reg [5:0] score\_count=0

);

always @(posedge button) begin

if(~gameover) begin

if(~reset)begin

score\_count <=0;

end

else if((button==ledin)&&(reset)) begin

if(score\_count <= 6'b111111)

score\_count<= score\_count + 1;

end

end

end

endmodule

`timescale 1ns / 1ps

module timer30(

input clk,

input reset,

output reg [4:0] count, //30 second timer

output reg gameover = 0

);

reg[4:0] current\_count=5;

reg five = 1;

always @ (posedge clk) begin

if(~reset)begin

five = 1;

current\_count=5;

end

if(five && (current\_count==0))begin

current\_count<=30;

five =0;

end

else if(current\_count == 0)begin

current\_count<=0;

gameover = 1;

end

else if(current\_count >= 1)begin

gameover = 0;

current\_count<=current\_count -1;

end

else

current\_count <= current\_count;

count<=current\_count;

end

endmodule

`timescale 1ns / 1ps

module display\_control(

output reg[5:0] anode,

output reg [3:0] num,

input [3:0] onesa,

input [3:0] tensa,

input [3:0] onesb,

input [3:0] tensb,

input clk,

input reset

);

reg[1:0] counter =0;

always@(posedge clk)

begin

if(~reset)

begin

anode=0;

counter=0;

end

else

begin

counter=counter+1'b1;

case(counter)

2'b00: begin

anode= 6'b111110;

num = onesb;

end

3'b01: begin

anode= 6'b111101;

num = tensb;

end

3'b10: begin

anode= 6'b101111;

num = onesa;

end

3'b11: begin

anode= 6'b011111;

num = tensa;

end

endcase

end

end

endmodule

`timescale 1ns / 1ps

module debouncer(

output reg [4:0] bout = 5'b00000,

input clk,

input reset,

input [4:0] bin //glitchy button

//////////

);

reg[31:0] debouncer\_counter =1'b0;

reg bout\_exists=0;

always @ (posedge clk) begin

if(~reset) begin

debouncer\_counter <=1'b0;

bout\_exists <=0;

end

if((bin == 1)&& reset)

begin

if(bout\_exists==0)

begin

if(debouncer\_counter ==1000000)

begin

bout <=4'b0001;

bout\_exists = 1'b1;

debouncer\_counter=32'b0;

end

else

begin

debouncer\_counter <= debouncer\_counter +1'b1;

end

end

else

begin

bout=4'b0000;

end

end

else if((bin == 4)&& reset)

begin

if(bout\_exists==0)

begin

if(debouncer\_counter ==1000000)

begin

bout <=4'b0010;

bout\_exists = 1'b1;

debouncer\_counter=32'b0;

end

else

begin

debouncer\_counter <= debouncer\_counter +1'b1;

end

end

else

begin

bout=4'b0000;

end

end

if((bin == 8)&& reset)

begin

if(bout\_exists==0)

begin

if(debouncer\_counter ==1000000)

begin

bout <=4'b0100;

bout\_exists = 1'b1;

debouncer\_counter=32'b0;

end

else

begin

debouncer\_counter <= debouncer\_counter +1'b1;

end

end

else

begin

bout=4'b0000;

end

end

if((bin == 16)&& reset)

begin

if(bout\_exists==0)

begin

if(debouncer\_counter ==1000000)

begin

bout <=4'b1000;

bout\_exists = 1'b1;

debouncer\_counter=32'b0;

end

else

begin

debouncer\_counter <= debouncer\_counter +1'b1;

end

end

else

begin

bout=4'b0000;

end

end

else

begin

bout\_exists=1'b0;

debouncer\_counter = 32'b0;

end

end

endmodule

`timescale 1ns / 1ps

module random(

input clk,

input reset,

input gameover,

output reg [4:0] rand = 5'b11111

);

reg [2:0] ran=3'b111;

reg [2:0] randh=3'b111;

wire tap = randh[2]^randh[1];

always@(posedge clk)

begin

if (gameover)

rand =5'b00000;

else if(~reset)begin

rand = 5'b00000;

randh = 3'b111;

end

randh = {randh[1:0], tap};

if ((randh<6)&&(randh>0)&& (reset))

ran <= randh-1;

case(ran)

0: rand = 5'b00001;

1: rand = 5'b00010;

2: rand = 5'b00100;

3: rand = 5'b01000;

4: rand = 5'b10000;

endcase

end

endmodule

`timescale 1ns / 1ps

module Whack\_a\_Mole(

input clk,

input[4:0] button,

input reset,

output reg [4:0] led,

output reg [5:0] score\_count);

localparam

S000000= 0,

S000001= 1,

S000010= 2,

S000011= 3,

S000100= 4,

S000101= 5,

S000110= 6,

S000111= 7,

S001000= 8,

S001001= 9,

S001010= 10,

S001011= 11,

S001100= 12,

S001101= 13,

S001110 = 14,

S001111= 15,

S010000= 16,

S010001= 17,

S010010= 18,

S010011= 19,

S010100= 20,

S010101= 21,

S010110= 22,

S010111= 23,

S011000= 24,

S011001= 25,

S011010= 26,

S011011= 27,

S011100= 28,

S011101= 29,

S011110= 30,

S011111= 31,

S100000= 32,

S100001= 33,

S100010= 34,

S100011= 35,

S100100= 36,

S100101= 37,

S100110= 38,

S100111= 39,

S101000= 40,

S101001= 41,

S101010= 42,

S101011= 43,

S101100= 44,

S101101= 45,

S101110= 46,

S101111= 47,

S110000= 48,

S110001= 49,

S110010= 50,

S110011= 51,

S110100= 52,

S110101= 53,

S110110= 54,

S110111= 55,

S111000= 56,

S111001= 57,

S111010= 58,

S111011= 59,

S111100= 60,

S111101= 61,

S111110= 62,

S111111= 63;

reg[5:0] current\_state =0;

reg[5:0] next\_state=0;

always @ (posedge clk) begin

if(reset)

current\_state <= S000000;

else

current\_state<=next\_state;

end

always @ (current\_state, button[4:0]) begin

case(current\_state)

S000000: begin //0

next\_state<= S000000;

//button[4:0] <=0;

score\_count <= 6'b000000;

led[4:0] <=0;

led[2] <=1;

if(button[2])

next\_state <=S000001;

else

next\_state <= S000000;

end

S000001: begin //1

score\_count <= 6'b000001;

led[4:0] <=0;

led[0] <=1;

if(button[0])

next\_state <=S000010;

else

next\_state <= S000001;

end

S000010: begin

score\_count <= 6'b000010;

led[4:0] <=0;

led[2] <=1;

if(button[2])

next\_state <=S000011;

else

next\_state <= S000010;

end

S000011: begin

score\_count <= 6'b000011;

led[4:0] <=0;

led[4] <=1;

if(button[4])

next\_state <=S000100;

else

next\_state <= S000011;

end

S000100: begin

score\_count <= 6'b000100;

led[4:0] <=0;

led[3] <=1;

if(button[3])

next\_state <=S000101;

else

next\_state <= S000100;

end

S000101: begin

score\_count <= 6'b000101;

led[4:0] <=0;

led[0] <=1;

if(button[0])

next\_state <=S000110;

else

next\_state <= S000101;

end

S000110: begin

score\_count <= 6'b000110;

led[4:0] <=0;

led[1] <=1;

if(button[1])

next\_state <=S000111;

else

next\_state <= S000110;

end

S000111: begin

score\_count <= 6'b000111;

led[4:0] <=0;

led[3] <=1;

if(button[3])

next\_state <=S001000;

else

next\_state <= S000111;

end

S001000: begin

score\_count <= 6'b001000;

led[4:0] <=0;

led[4] <=1;

if(button[4])

next\_state <=S001001;

else

next\_state <= S001000;

end

S001001: begin

score\_count <= 6'b001001;

led[4:0] <=0;

led[0] <=1;

if(button[0])

next\_state <=S001010;

else

next\_state <= S001001;

end

S001010: begin

score\_count <= 6'b001010;

led[4:0] <=0;

led[2] <=1;

if(button[2])

next\_state <=S001011;

else

next\_state <= S001010;

end

S001011: begin

score\_count <= 6'b001011;

led[4:0] <=0;

led[1] <=1;

if(button[1])

next\_state <=S001100;

else

next\_state <= S001011;

end

S001100: begin

score\_count <= 6'b001100;

led[4:0] <=0;

led[0] <=1;

if(button[0])

next\_state <=S001101;

else

next\_state <= S001100;

end

S001101: begin

score\_count <= 6'b001101;

led[4:0] <=0;

led[3] <=1;

if(button[3])

next\_state <=S001110;

else

next\_state <= S001101;

end

S001110: begin

score\_count <= 6'b001110;

led[4:0] <=0;

led[1] <=1;

if(button[1])

next\_state <=S001111;

else

next\_state <= S001110;

end

S001111: begin

score\_count <= 6'b001111;

led[4:0] <=0;

led[4] <=1;

if(button[4])

next\_state <=S010000;

else

next\_state <= S001111;

end

S010000: begin

score\_count <= 6'b010000;

led[4:0] <=0;

led[2] <=1;

if(button[2])

next\_state <=S010001;

else

next\_state <= S010000;

end

S010001: begin

score\_count <= 6'b010001;

led[4:0] <=0;

led[0] <=1;

if(button[0])

next\_state <=S010010;

else

next\_state <= S010001;

end

S010010: begin

score\_count <= 6'b010010;

led[4:0] <=0;

led[4] <=1;

if(button[4])

next\_state <=S010011;

else

next\_state <= S010010;

end

S010011: begin

score\_count <= 6'b010011;

led[4:0] <=0;

led[1] <=1;

if(button[1])

next\_state <=S010100;

else

next\_state <= S010011;

end

S010100: begin

score\_count <= 6'b010100;

led[4:0] <=0;

led[3] <=1;

if(button[3])

next\_state <=S010101;

else

next\_state <= S010100;

end

S010101: begin

score\_count <= 6'b010101;

led[4:0] <=0;

led[1] <=1;

if(button[1])

next\_state <=S010110;

else

next\_state <= S010101;

end

S010110: begin

score\_count <= 6'b010110;

led[4:0] <=0;

led[4] <=1;

if(button[4])

next\_state <=S010111;

else

next\_state <= S010110;

end

S010111: begin

score\_count <= 6'b010111;

led[4:0] <=0;

led[2] <=1;

if(button[2])

next\_state <=S011000;

else

next\_state <= S010111;

end

S011000: begin

score\_count <= 6'b011000;

led[4:0] <=0;

led[4] <=1;

if(button[4])

next\_state <=S011001;

else

next\_state <= S011000;

end

S011001: begin

score\_count <= 6'b011001;

led[4:0] <=0;

led[0] <=1;

if(button[0])

next\_state <=S011010;

else

next\_state <= S011001;

end

S011010: begin

score\_count <= 6'b011010;

led[4:0] <=0;

led[1] <=1;

if(button[1])

next\_state <=S011011;

else

next\_state <= S011010;

end

S011011: begin

score\_count <= 6'b011011;

led[4:0] <=0;

led[4] <=1;

if(button[4])

next\_state <=S011100;

else

next\_state <= S011011;

end

S011100: begin

score\_count <= 6'b011100;

led[4:0] <=0;

led[2] <=1;

if(button[2])

next\_state <=S011101;

else

next\_state <= S011100;

end

S011101: begin

score\_count <= 6'b011101;

led[4:0] <=0;

led[0] <=1;

if(button[0])

next\_state <=S011110;

else

next\_state <= S011101;

end

S011110: begin

score\_count <= 6'b011110;

led[4:0] <=0;

led[1] <=1;

if(button[1])

next\_state <=S011111;

else

next\_state <= S011110;

end

S011111: begin

score\_count <= 6'b011111;

led[4:0] <=0;

led[3] <=1;

if(button[3])

next\_state <=S100000;

else

next\_state <= S011111;

end

S100000: begin

score\_count <= 6'b100000;

led[4:0] <=0;

led[4] <=1;

if(button[4])

next\_state <=S100001;

else

next\_state <= S100000;

end

S100001: begin

score\_count <= 6'b100001;

led[4:0] <=0;

led[2] <=1;

if(button[2])

next\_state <=S100010;

else

next\_state <= S100001;

end

S100010: begin

score\_count <= 6'b100010;

led[4:0] <=0;

led[4] <=1;

if(button[4])

next\_state <=S100011;

else

next\_state <= S100010;

end

S100011: begin

score\_count <= 6'b100011;

led[4:0] <=0;

led[1] <=1;

if(button[1])

next\_state <=S100100;

else

next\_state <= S100011;

end

S100100: begin

score\_count <= 6'b100100;

led[4:0] <=0;

led[0] <=1;

if(button[0])

next\_state <=S100101;

else

next\_state <= S100100;

end

S100101: begin

score\_count <= 6'b100101;

led[4:0] <=0;

led[2] <=1;

if(button[2])

next\_state <=S100110;

else

next\_state <= S100101;

end

S100110: begin

score\_count <= 6'b100110;

led[4:0] <=0;

led[3] <=1;

if(button[3])

next\_state <=S100111;

else

next\_state <= S100110;

end

S100111: begin

score\_count <= 6'b100111;

led[4:0] <=0;

led[2] <=1;

if(button[2])

next\_state <=S101000;

else

next\_state <= S100111;

end

S101000: begin

score\_count <= 6'b101000;

led[4:0] <=0;

led[0] <=1;

if(button[0])

next\_state <=S101001;

else

next\_state <= S101000;

end

S101001: begin

score\_count <= 6'b101001;

led[4:0] <=0;

led[1] <=1;

if(button[1])

next\_state <=S101010;

else

next\_state <= S101001;

end

S101010: begin

score\_count <= 6'b101010;

led[4:0] <=0;

led[0] <=1;

if(button[0])

next\_state <=S101011;

else

next\_state <= S101010;

end

S101011: begin

score\_count <= 6'b000000;

led[4:0] <=0;

led[4] <=1;

if(button[4])

next\_state <=S101100;

else

next\_state <= S101011;

end

S101100: begin

score\_count <= 6'b101100;

led[4:0] <=0;

led[3] <=1;

if(button[3])

next\_state <=S101101;

else

next\_state <= S101100;

end

S101101: begin

score\_count <= 6'b101101;

led[4:0] <=0;

led[0] <=1;

if(button[0])

next\_state <=S101110;

else

next\_state <= S101101;

end

S101110: begin

score\_count <= 6'b101110;

led[4:0] <=0;

led[3] <=1;

if(button[3])

next\_state <=S101111;

else

next\_state <= S101110;

end

S101111: begin

score\_count <= 6'b101111;

led[4:0] <=0;

led[0] <=1;

if(button[0])

next\_state <=S110000;

else

next\_state <= S101111;

end

S110000: begin

score\_count <= 6'b110000;

led[4:0] <=0;

led[2] <=1;

if(button[2])

next\_state <=S110001;

else

next\_state <= S110000;

end

S110001: begin

score\_count <= 6'b110001;

led[4:0] <=0;

led[1] <=1;

if(button[1])

next\_state <=S110010;

else

next\_state <= S110001;

end

S110010: begin

score\_count <= 6'b110010;

led[4:0] <=0;

led[2] <=1;

if(button[2])

next\_state <=S110011;

else

next\_state <= S110010;

end

S110011: begin

score\_count <= 6'b110011;

led[4:0] <=0;

led[1] <=1;

if(button[1])

next\_state <=S110100;

else

next\_state <= S110011;

end

S110100: begin

score\_count <= 6'b110100;

led[4:0] <=0;

led[4] <=1;

if(button[4])

next\_state <=S110101;

else

next\_state <= S110100;

end

S110101: begin

score\_count <= 6'b110101;

led[4:0] <=0;

led[2] <=1;

if(button[2])

next\_state <=S110110;

else

next\_state <= S110101;

end

S110110: begin

score\_count <= 6'b110110;

led[4:0] <=0;

led[3] <=1;

if(button[3])

next\_state <=S110111;

else

next\_state <= S110110;

end

S110111: begin

score\_count <= 6'b110111;

led[4:0] <=0;

led[0] <=1;

if(button[0])

next\_state <=111000;

else

next\_state <= S110111;

end

111000: begin

score\_count <= 6'b111000;

led[4:0] <=0;

led[1] <=1;

if(button[1])

next\_state <=S111001;

else

next\_state <= S111000;

end

S111001: begin

score\_count <= 6'b111001;

led[4:0] <=0;

led[4] <=1;

if(button[4])

next\_state <=S111010;

else

next\_state <= S111001;

end

S111010: begin

score\_count <= 6'b111010;

led[4:0] <=0;

led[2] <=1;

if(button[2])

next\_state <=S111011;

else

next\_state <= S111010;

end

S111011: begin

score\_count <= 6'b111011;

led[4:0] <=0;

led[3] <=1;

if(button[3])

next\_state <=S111100;

else

next\_state <= S111011;

end

S111100: begin

score\_count <= 6'b111100;

led[4:0] <=0;

led[0] <=1;

if(button[0])

next\_state <=S111101;

else

next\_state <= S111100;

end

S111101: begin

score\_count <= 6'b111101;

led[4:0] <=0;

led[3] <=1;

if(button[3])

next\_state <=S111110;

else

next\_state <= S111101;

end

S111110: begin

score\_count <= 6'b111110;

led[4:0] <=0;

led[4] <=1;

if(button[4])

next\_state <=S111111;

else

next\_state <= S111110;

end

S111111: begin

next\_state <= S111111;

score\_count <= 63;

led[4:0] <=0;

end

default: begin

next\_state = S000000;

end

endcase

end

endmodule

`timescale 1ns / 1ps

module timer30(

input clk,

input reset,

output reg [4:0] count, //30 second timer

output reg gameover = 0

);

reg[4:0] current\_count=5;

reg five = 1;

always @ (posedge clk) begin

if(~reset)begin

five = 1;

current\_count=5;

end

if(five && (current\_count==0))begin

current\_count<=30;

five =0;

end

else if(current\_count == 0)begin

current\_count<=0;

gameover = 1;

end

else if(current\_count >= 1)begin

gameover = 0;

current\_count<=current\_count -1;

end

else

current\_count <= current\_count;

count<=current\_count;

end

endmodule

`timescale 1ns / 1ps

module display\_control(

output reg[5:0] anode,

output reg [3:0] num,

input [3:0] onesa,

input [3:0] tensa,

input [3:0] onesb,

input [3:0] tensb,

input clk,

input reset

);

reg[1:0] counter =0;

always@(posedge clk)

begin

if(~reset)

begin

anode=0;

counter=0;

end

else

begin

counter=counter+1'b1;

case(counter)

2'b00: begin

anode= 6'b111110;

num = onesb;

end

3'b01: begin

anode= 6'b111101;

num = tensb;

end

3'b10: begin

anode= 6'b101111;

num = onesa;

end

3'b11: begin

anode= 6'b011111;

num = tensa;

end

endcase

end

end

endmodule

`timescale 1ns / 1ps

module slow\_clk(in\_clk, out\_clk); //1 Hz clock

input in\_clk;

output reg out\_clk;

reg [25:0] count=0;

always@(posedge in\_clk)

begin

count <= count + 1'b1;

if(count == 50000000)

begin

out\_clk <=~out\_clk;

count <= 0;

end

end

endmodule

`timescale 1ns / 1ps

`timescale 1ns / 1ps

module fast\_clk(in\_clk, out\_clk);

input in\_clk;

output reg out\_clk;

reg [8:0] count=0;

always@(posedge in\_clk)

begin

count <= count + 1'b1;

if(count == 500)

begin

out\_clk <=~out\_clk;

count <= 0;

end

end

endmodule

`timescale 1ns / 1ps

module seven\_segment(

output reg [6:0]sev,

input [3:0] num

);

always @(num)

begin

case (num)

0: sev = 7'b1000000;

1: sev = 7'b1111001;

2: sev = 7'b0100100;

3: sev = 7'b0110000;

4: sev = 7'b0011001;

5: sev = 7'b0010010;

6: sev = 7'b0000010;

7: sev = 7'b1111000;

8: sev = 7'b0000000;

9: sev = 7'b0011000;

endcase

end

endmodule

**BCD Counter?**

module BCDcountmod(

Input Clock,

Input Clear,

Input E,

Output reg [3:0] BCD1,

Output reg [3:0] BCD0);

always @(posedge Clock)begin

if (Clear) begin

BCD1 <= 0;

BCD0 <= 0;

End else if (E)

if (BCD0 == 4’b1001) begin

BCD0 <= 0;

if (BCD1 == 4’b1001)

BCD1 <= 0;

Else

BCD1 <= BCD1 + 1;

End else

BCD0 <= BCD0 + 1;

end

endmodule

Option 2:

module BCDCount (CLK, clear, load, a0, a1);  
 input CLK, reset, in;  
 output reg a0, a1;

reg [1:0] state; // state variables  
reg [1:0] next\_state;

always @(posedge CLK) begin

state = next\_state;

end

always @(state or clear or load) begin

case (state)

2’b00: next\_state = 2’b01;

2’b01: next\_state = 2’b10;  
 2’b10: next\_state = 2’b11;  
 2’b11: next\_state = 2’b00;

endcase  
 if (clear) next\_state = 2’b00;

// handle load

end

end

assign a0 = state[0];

assign a1 = state[1];

endmodule