```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: 
// Engineer: 
// 
// Create Date: 02/27/2018 03:17:34 PM
// Design Name: 
// Module Name: Top
// Project Name: 
// Target Devices: 
// Tool Versions: 
// Description: 
// 
// Dependencies: 
// 
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
// 
//////////////////////////////////////////////////////////////////////////////////


module Top(
    input clkin,
    input btnR,
    input btnL,
    input btnU,
    input btnD,
    input btnC,
    input [15:0] sw,
    output Vsync,
    output Hsync,
    output [3:0] vgaRed,
    output [3:0] vgaGreen,
    output [3:0] vgaBlue,
    output [6:0] seg,
    output [3:0] an,
    output dp,
    output [15:0] led
    );

    wire clk, digsel, active, R, G, B, HalfSec;
    wire Collision, FourSec, GameStart, Win;
    wire GameReset, MovePieces, MoveSkiier, SCcountU, SCcountD, FlashSkiier, 
FlashBorder, GateTop, RCounter;
    wire NewFrame, EightFrame;
    wire SSwitch, SSwitchEdge, SLeft, SRight, ScntU, ScntD;
    wire [9:0] Hadd, Vadd;
    wire [15:0] SHP, SVP, FQ;
    wire [7:0] rnd, Score;
    wire [6:0] Score2SC, ScoreC2SC, toseg;
    wire [3:0] GHP1, GHP2, GHP3, GHP4, GHP5, GHP6, GHP7, GHP8;
    wire [15:0] GVP1, GVP2, GVP3, GVP4, GVP5, GVP6, GVP7, GVP8;
    wire [3:0] plusminus, sel, tohex7seg;
    wire [2:0] gapSize;
    wire [16:0] DC;

    // temp var assignments

    lab7_clks not_so_slow (.clkin(clkin), .greset(sw[0]), .clk(clk), .digsel(digsel))

    VGAController MyVGA (.clk(clk), .Hsync(Hsync), .Vsync(Vsync), .Hadd(Hadd),
.Vadd(Vadd), .Active(active));

    // Frame Detection
    Edge_Detector NewFrameEdge (.clk(clk), .in(Vsync), .out(NewFrame));
    counterUD16L FrameCounter (.clk(clk), .Din(16'b0), .up(NewFrame), .dw(1'b0),
.r(1'b0), .ld(RCounter), .Q(FQ));
    Edge_Detector EightFrameEdge (.clk(clk), .in(FQ[2]), .out(EightFrame));

    // GameStateMachine
    assign Collision = R&G;
    assign FourSec = FQ == 10'd240;
    FDRE #(.INIT(1'b0)) btnCSync (.C(clk), .CE(1'b1), .D(btnC), .Q(GameStart)); //
synchronize btnC
    assign Win = SVP == 10'd400;
    GameStateMachine MyGame(.clk(clk), .Collision(Collision), .FourSec(FourSec),
.GameStart(GameStart), .Win(Win),
    .GameReset(GameReset), .MovePieces(MovePieces), .MoveSkiier(MoveSkiier),
.SCcountU(SCcountU), .SCcountD(SCcountD), .FlashSkiier(FlashSkiier),
    .FlashBorder, .GateTop(GateTop), .RCounter(RCounter));

    // Keep track of and display Score
    CountUDL8 ScoreCounter (clk, SCcountU, SCcountD, 1'b0, 8'b00000000, Score);

     Adder8 Addone({1'b0, ~Score[6:0]}, 8'b00000000, 1'b1, {DC[0], Score2SC[6:0]},
DC[1]);
     m2_1x8 SC2Mux({1'b0, Score[6:0]}, {1'b0, Score2SC[6:0]}, Score[7], 1'b1, {DC[2],
ScoreC2SC[6:0]});

     Ring_Counter MyCounter(digsel, clk, sel);
     Selector MySelector(sel, {4'b0000, 4'b0000, 1'b0, ScoreC2SC[6:0]}, tohex7seg);
     hex7seg myHex7Seg(tohex7seg, toseg);

     // inverts '0' to '-' when sel[2] is active
     assign seg[6:0] = toseg ^ {7{sel[2]}};

         // assign anodes
     assign an[1:0] = ~sel[1:0];
     assign an[2] = ~(sel[2]&Score[7]);
     assign an[3] = 1'b1;



    // Blue Border Control
    assign B = active && FlashBorder && HalfSec && (Hadd < 10'd8 || Hadd > 10'd631 ||
Vadd < 10'd8 || Vadd > 10'd471)
    || active && ~FlashBorder && (Hadd < 10'd8 || Hadd > 10'd631 || Vadd < 10'd8 ||
Vadd > 10'd471);

    // Skiier Control
    assign HalfSec = FQ[4]; // alternates between high and low every 32 frames ~half
a second

    assign SSwitch = G & B;
    Edge_Detector SwitchEdge (.clk(clk), .in(SSwitch), .out(SSwitchEdge)); // when

    FDRE #(.INIT(1'b0)) btnLSync (.C(clk), .CE(1'b1), .D(btnL), .Q(SLeft)); //
synchronize btnL
    FDRE #(.INIT(1'b0)) btnRSync (.C(clk), .CE(1'b1), .D(btnR), .Q(SRight)); //
synchronize btnR

    SkiStateMachine MySkiier (.clk(clk), .Switch(SSwitchEdge), .Left(SLeft),
.Right(SRight), .ScntU(ScntU), .ScntD(ScntD));
    counterUD16L SHCount (.clk(clk), .Din(10'd312), .up(NewFrame&ScntU&MoveSkiier),
.dw(NewFrame&ScntD&MoveSkiier), .r(1'b0), .ld(GameReset), .Q(SHP));
    counterUD16L SVCount (.clk(clk), .Din(10'd24), .up(EightFrame&MoveSkiier),
.dw(1'b0), .r(1'b0), .ld(GameReset), .Q(SVP));
    assign G = active && FlashSkiier && HalfSec && (Hadd>=SHP && Hadd<(SHP+10'd15) &&
Vadd>=SVP && Vadd<(SVP+10'd15))
    || active && ~FlashSkiier && (Hadd>=SHP && Hadd<(SHP+10'd15) && Vadd>=SVP &&
Vadd<(SVP+10'd15));

// Gate Control
// generate a random 8-bit value
LFSR random (.clk(clk), .run(1'b1), .reset(1'b0), .rnd(rnd));

//Map rnd 2,1,0 to -2,-1,0,1,2
assign plusminus = {rnd[3]&(rnd[1]|rnd[0]),
rnd[3]&(rnd[1]|rnd[0]),
rnd[3] | ~rnd[3]&rnd[1]&rnd[0],
rnd[3]&~rnd[1]&rnd[0] | ~rnd[3]&rnd[1]&~rnd[0]};

// Gates
Gate Gate1 (.clk(clk), .NewFrame(NewFrame), .GameReset(GameReset),
.plusminus(plusminus), .MovePieces(MovePieces), .GateTop(GateTop),
.prevgateH(GHP8), .initV(16'b0000000001100100), .initH(4'b0111), .GHP(GHP1),
.GVP(GVP1));
Gate Gate2 (.clk(clk), .NewFrame(NewFrame), .GameReset(GameReset),
.plusminus(plusminus), .MovePieces(MovePieces), .GateTop(GateTop),
.prevgateH(GHP1), .initV(16'b0000000010100000), .initH(4'b1000), .GHP(GHP2),
.GVP(GVP2));
Gate Gate3 (.clk(clk), .NewFrame(NewFrame), .GameReset(GameReset),
.plusminus(plusminus), .MovePieces(MovePieces), .GateTop(GateTop),
.prevgateH(GHP2), .initV(16'b0000000011011100), .initH(4'b1001), .GHP(GHP3),
.GVP(GVP3));
Gate Gate4 (.clk(clk), .NewFrame(NewFrame), .GameReset(GameReset),
.plusminus(plusminus), .MovePieces(MovePieces), .GateTop(GateTop),
.prevgateH(GHP3), .initV(16'b0000000100011000), .initH(4'b1010), .GHP(GHP4),
.GVP(GVP4));
Gate Gate5 (.clk(clk), .NewFrame(NewFrame), .GameReset(GameReset),
.plusminus(plusminus), .MovePieces(MovePieces), .GateTop(GateTop),
.prevgateH(GHP4), .initV(16'b0000000101010100), .initH(4'b1000), .GHP(GHP5),
.GVP(GVP5));
Gate Gate6 (.clk(clk), .NewFrame(NewFrame), .GameReset(GameReset),
.plusminus(plusminus), .MovePieces(MovePieces), .GateTop(GateTop),
.prevgateH(GHP5), .initV(16'b0000000110010000), .initH(4'b1001), .GHP(GHP6),
.GVP(GVP6));
Gate Gate7 (.clk(clk), .NewFrame(NewFrame), .GameReset(GameReset),
.plusminus(plusminus), .MovePieces(MovePieces), .GateTop(GateTop),
.prevgateH(GHP6), .initV(16'b0000000111001100), .initH(4'b1011), .GHP(GHP7),
.GVP(GVP7));
Gate Gate8 (.clk(clk), .NewFrame(NewFrame), .GameReset(GameReset),
.plusminus(plusminus), .MovePieces(MovePieces), .GateTop(GateTop),
.prevgateH(GHP7), .initV(16'b0000001000001000), .initH(4'b1101), .GHP(GHP8),
.GVP(GVP8));

// Find out when a pixel will be red based on gate positions and gate width
assign gapSize = sw[6:4];
assign R = active && ~B && (
Hadd >= 0 && Hadd <= 10'd80+10'd32*GHP1-(10'd4+10'd8*gapSize) && Vadd >= GVP1 && Vadd
<= GVP1+10'd7
|| Hadd <= 10'd639 && Hadd >= 10'd80+10'd32*GHP1+(10'd4+10'd8*gapSize) && Vadd >=
GVP1 && Vadd <= GVP1+10'd7
|| Hadd >= 0 && Hadd <= 10'd80+10'd32*GHP2-(10'd4+10'd8*gapSize) && Vadd >= GVP2 &&
Vadd <= GVP2+10'd7
|| Hadd <= 10'd639 && Hadd >= 10'd80+10'd32*GHP2+(10'd4+10'd8*gapSize) && Vadd >=
GVP2 && Vadd <= GVP2+10'd7
|| Hadd >= 0 && Hadd <= 10'd80+10'd32*GHP3-(10'd4+10'd8*gapSize) && Vadd >= GVP3 &&
Vadd <= GVP3+10'd7
|| Hadd <= 10'd639 && Hadd >= 10'd80+10'd32*GHP3+(10'd4+10'd8*gapSize) && Vadd >=
GVP3 && Vadd <= GVP3+10'd7
|| Hadd >= 0 && Hadd <= 10'd80+10'd32*GHP4-(10'd4+10'd8*gapSize) && Vadd >= GVP4 &&
Vadd <= GVP4+10'd7
|| Hadd <= 10'd639 && Hadd >= 10'd80+10'd32*GHP4+(10'd4+10'd8*gapSize) && Vadd >=
GVP4 && Vadd <= GVP4+10'd7
|| Hadd >= 0 && Hadd <= 10'd80+10'd32*GHP5-(10'd4+10'd8*gapSize) && Vadd >= GVP5 &&
Vadd <= GVP5+10'd7
|| Hadd <= 10'd639 && Hadd >= 10'd80+10'd32*GHP5+(10'd4+10'd8*gapSize) && Vadd >=
GVP5 && Vadd <= GVP5+10'd7
|| Hadd >= 0 && Hadd <= 10'd80+10'd32*GHP6-(10'd4+10'd8*gapSize) && Vadd >= GVP6 &&
Vadd <= GVP6+10'd7
|| Hadd <= 10'd639 && Hadd >= 10'd80+10'd32*GHP6+(10'd4+10'd8*gapSize) && Vadd >=
GVP6 && Vadd <= GVP6+10'd7
|| Hadd >= 0 && Hadd <= 10'd80+10'd32*GHP7-(10'd4+10'd8*gapSize) && Vadd >= GVP7 &&
Vadd <= GVP7+10'd7
|| Hadd <= 10'd639 && Hadd >= 10'd80+10'd32*GHP7+(10'd4+10'd8*gapSize) && Vadd >=
GVP7 && Vadd <= GVP7+10'd7
|| Hadd >= 0 && Hadd <= 10'd80+10'd32*GHP8-(10'd4+10'd8*gapSize) && Vadd >= GVP8 &&
Vadd <= GVP8+10'd7
|| Hadd <= 10'd639 && Hadd >= 10'd80+10'd32*GHP8+(10'd4+10'd8*gapSize) && Vadd >=
GVP8 && Vadd <= GVP8+10'd7
);



    // assign the color signals

    assign vgaBlue = {4{B}};
    assign vgaGreen = {4{G}};
    assign vgaRed = {4{R}};

    assign dp = 1'b1;
    assign led = 16'b0;

endmodule
```