```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 02/13/2018 01:44:43 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 btnD,
    input btnU,
    input [15:0] sw,
    output [15:0] led,
    output [3:0] an,
    output [6:0] seg
    );

    wire btnD_sync, btnU_sync, digsel, qsec;
    wire  ShowScore, ShowArgs, RTimer, cntL, cntR, FlashL, FlashR, RunLFSR, RScore,
alt, over;
    wire [3:0] sel, ScoreR, ScoreL, toDisplay;
    wire [7:0] p, rnd;
    wire [4:0] Q_timer, Time;

    assign Time = {5'b11000};   // time for each phase in quarters of a second

    lab5_clks myClks (.clkin(clkin), .greset(btnR), .clk(clk), .digsel(digsel),
.qsec(qsec));

    // syncronize inputs
```

```
    FDRE #(.INIT(1'b0) ) ff_btnD (.C(clk), .CE(1'b1), .D(btnD), .Q(btnD_sync));
    FDRE #(.INIT(1'b0) ) ff_btnU (.C(clk), .CE(1'b1), .D(btnU), .Q(btnU_sync));

    assign over = &(~(ScoreL^sw[14:11])) | &(~(ScoreR^sw[14:11]));

    State_Machine Big_Boss (.clk(clk), .Go(btnU_sync), .Ans(btnD_sync),
.correct(&(~(sw[7:0] ^ p))),
    .Over(over), .FourSec(&(~(Q_timer^Time))), .ShowScore(ShowScore),
    .ShowArgs(ShowArgs), .RTimer(RTimer), .cntL(cntL), .cntR(cntR), .FlashL(FlashL),
.FlashR(FlashR), .RunLFSR(RunLFSR),
    .RScore(RScore));

    Time_Counter Timer (clk, qsec, RTimer, Q_timer);

    CountUD4L LCount (.clk(clk), .up(cntL), .dw(1'b0), .ld(RScore), .Din({4'b0000}),
.Q(ScoreL));
    CountUD4L RCount (.clk(clk), .up(cntR), .dw(1'b0), .ld(RScore), .Din({4'b0000}),
.Q(ScoreR));

    Multiplier myMult (.m(rnd[3:0]), .q(rnd[7:4]), .p(p));

    LFSR rand (clk, RunLFSR, rnd);

    // Generates 7seg vals to display
    Ring_Counter Seg_Regulator (digsel, clk, sel);
    Selector Seg_Selector (sel, {ScoreL, rnd[7:4], rnd[3:0], ScoreR}, toDisplay);
    hex7seg my7Seg (toDisplay, seg);

    // Manage anodes
    assign an[2] = ~(ShowArgs&sel[2]);
    assign an[1] = ~(ShowArgs&sel[1]);

        // flashes segs 3 and 0
    FDRE #(.INIT(1'b0) ) alternate (.C(clk), .CE(qsec), .D(~alt), .Q(alt));

    assign an[3] = (~ShowScore|~alt|~sel[3]) & (~ShowScore|FlashL|~sel[3]);
    assign an[0] = (~ShowScore|~alt|~sel[0]) & (~ShowScore|FlashR|~sel[0]);

    // manage leds
    assign led[7:0] = sw[7:0];
    assign led[15:8] = (p & {8{sw[15]}}) & {8{ShowArgs}};

endmodule
```