`timescale 1ns / 1ps

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

// Company: Ratner Surf Designs

// Engineer: James Ratner

//

// Create Date: 11/04/2018 09:56:49 AM

// Design Name:

// Module Name: seq\_detect\_driver

// Project Name:

// Target Devices:

// Tool Versions:

// Description: Driver for sequence detector experiment

//

// Dependencies: This module requires the following modules:

//

// cntr\_udclr\_nb (provided)

// clk\_divder\_nbit (provided)

// mux\_8t1\_nb (provided)

// mux\_2t1\_nb (provided)

// stand\_dcdr\_2t4\_1cold (provided)

// fsm\_module (you must provide)

//

// Revision:

// Revision 1.00 - File Created (11-04-2018)

// 1.01 - added btn input (11-06-2018)

// 1.02 - added reset, fixed cool-mux

// added clk hooks for simulation (11-10-2018)

// 1.03 - offset x-bit (by one) to improve display (11-12-2018)

// Additional Comments:

//

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

module seq\_detect\_driver(

input [7:0] switches,

input btn,

input clk,

input reset,

output [7:0] segs,

output [3:0] an,

output reg [7:0] leds

);

wire clk\_slow;

wire [2:0] mux\_sel;

wire switch\_bit;

wire clk\_mux\_disp;

wire [1:0] multiplex\_sel;

reg [7:0] cool\_seg\_data;

reg [7:0] crud\_seg\_data;

wire fsm\_Z;

// dummy FSM module

fsm\_seq my\_fsm (

.clk (clk\_slow),

// .clk (clk),

.btn (btn),

.x (switch\_bit),

.z (fsm\_Z)

);

//- driver for LEDs

always @ (mux\_sel)

begin

case (mux\_sel)

1: leds = 'h01; //

2: leds = 'h02; //

3: leds = 'h04; //

4: leds = 'h08; //

5: leds = 'h10; //

6: leds = 'h20; //

7: leds = 'h40; //

0: leds = 'h80; //

default leds = 0;

endcase

end

//- clock divider ~2Hz

clk\_divder\_nbit #(.n(25)) MY\_DIV (

.clockin (clk),

.clockout (clk\_slow)

);

// MUX to decider FSM data input

mux\_8t1\_nb #(.n(1)) my\_8t1\_mux (

.SEL (mux\_sel),

.D0 (switches[0]),

.D1 (switches[1]),

.D2 (switches[2]),

.D3 (switches[3]),

.D4 (switches[4]),

.D5 (switches[5]),

.D6 (switches[6]),

.D7 (switches[7]),

.D\_OUT (switch\_bit) );

// counter to drive switch input MUX sel

cntr\_udclr\_nb #(3) my\_led\_clk (

.clk (clk\_slow),

// .clk (clk),

.clr (reset),

.up (1),

.ld (0),

.D (0),

.count (mux\_sel),

.rco () );

// counter to drive switch input MUX sel

cntr\_udclr\_nb #(2) my\_disp\_multiplex\_cntr (

.clk (clk\_mux\_disp),

.clr (0),

.up (1),

.ld (0),

.D (0),

.count (multiplex\_sel),

.rco () );

//- clock divider for muliplexed displayz

clk\_divder\_nbit #(.n(13)) mux\_display\_clk (

.clockin (clk),

.clockout (clk\_mux\_disp)

);

//- standard decoder to drive anodes

stand\_dcdr\_2t4\_1cold my\_stand\_dcdr (

.SEL (multiplex\_sel),

.D\_OUT (an) );

//- 7 seg decoder for good message

always @ (multiplex\_sel)

begin

case (multiplex\_sel)

0: cool\_seg\_data = 'h63;

1: cool\_seg\_data = 'hC5;

2: cool\_seg\_data = 'hC5;

3: cool\_seg\_data = 'hE3;

default cool\_seg\_data = 0;

endcase

end

//- 7 seg decoder for bad message

always @ (multiplex\_sel)

begin

case (multiplex\_sel)

0: crud\_seg\_data = 'h63;

1: crud\_seg\_data = 'hF5;

2: crud\_seg\_data = 'hC7;

3: crud\_seg\_data = 'h85;

default crud\_seg\_data = 0;

endcase

end

//- Selects either good/bad message based on FSM output

mux\_2t1\_nb #(.n(8)) my\_2t1\_mux (

.SEL (fsm\_Z),

.D0 (crud\_seg\_data),

.D1 (cool\_seg\_data),

.D\_OUT (segs) );

endmodule