`timescale 1ns / 1ps

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

// Company: Texas Tech University

// Engineer: Steve Gillet

//

// Create Date: 09/11/2021 06:16:29 PM

// Design Name: miniProject2

// Module Name: projectLab1b

// Project Name: DethKopter2000

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

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

module projectLab1b(

input clock,

input switch12,

input senseA,

input senseB,

input senseFront,

input senseLeft,

input senseRight,

input senseObject,

output a, b, c, d, e, f, g, dp, //the individual LED output for the seven segment along with the digital point

output [3:0] anSS,

output input1,

output input2,

output input3,

output input4,

output PWMenB,

output PWMenA

);

localparam N = 16;

reg [N:0]countSS;

reg [20:0] counter; //sets period

reg [6:0] counterTurn;

// reg [26:0] counterOverCurrent;

reg [20:0] widthEnB;

reg [20:0] widthEnA;

reg [20:0] tempWidthEnB;

reg [20:0] tempWidthEnA;

reg [20:0] tempWidthReverse;

reg tempPWMenB;

reg tempPWMenA;

reg [1:0]sseg;

reg [3:0]anSS\_temp;

reg [6:0] sseg\_temp; // 7 bit register to hold the binary value of each input given

reg input1reg;

reg input2reg;

reg input3reg;

reg input4reg;

reg [1:0] stateTurn;

parameter forward = 0,

backward = 1,

backwardFull = 2;

sevenSeg mainSS(

.clock(clock),

.senseA(senseA),

.senseB(senseB),

.a (a),

.b (b),

.c (c),

.d (d),

.e (e),

.f (f),

.g (g),

.dp (dp),

.input1(input1),

.input2(input2),

.input3(input3),

.input4(input4),

.anSS (anSS)

);

always@(posedge clock) begin

if (counter == 1666666)

counter <= 0;

else

counter <= counter + 1;

if (counterTurn == 10)

counterTurn <= 0;

else

counterTurn <= counterTurn + 1;

// if (senseA == 1)

// counterOverCurrent <= counterOverCurrent+1;

// else

// counterOverCurrent <= 0;

case (switch12)

1'b1:

begin

if(senseFront == 1 && senseLeft == 0 && senseRight == 0)

begin

tempWidthEnA <= 1666666;

tempWidthEnB <= 1666666;

input1reg <= 1;

input2reg <= 0;

input3reg <= 0;

input4reg <= 1;

widthEnA <= tempWidthEnA;

widthEnB <= tempWidthEnB;

tempWidthReverse = 0;

stateTurn = forward;

end

if(senseLeft && senseFront && !senseRight)

begin

if(senseObject == 1)

begin

tempWidthEnA <= 1666666;

tempWidthEnB <= 1666666;

input1reg <= 0;

input2reg <= 1;

input3reg <= 0;

input4reg <= 1;

widthEnA <= tempWidthEnA;

widthEnB <= tempWidthEnB;

tempWidthReverse = 0;

stateTurn = forward;

end

if(senseObject == 0)

begin

tempWidthEnA <= 1666666;

tempWidthEnB <= 1666666;

input1reg <= 1;

input2reg <= 0;

input3reg <= 0;

input4reg <= 1;

widthEnA <= tempWidthEnA;

widthEnB <= tempWidthEnB;

tempWidthReverse = 0;

end

end

if(senseRight && senseFront && !senseLeft)

begin

if(senseObject)

begin

tempWidthEnA <= 1666666;

tempWidthEnB <= 1666666;

input1reg <= 1;

input2reg <= 0;

input3reg <= 1;

input4reg <= 0;

widthEnA <= tempWidthEnA;

widthEnB <= tempWidthEnB;

tempWidthReverse = 0;

end

if(!senseObject)

begin

tempWidthEnA <= 1666666;

tempWidthEnB <= 1666666;

input1reg <= 1;

input2reg <= 0;

input3reg <= 0;

input4reg <= 1;

widthEnA <= tempWidthEnA;

widthEnB <= tempWidthEnB;

tempWidthReverse = 0;

stateTurn = forward;

end

end

if(senseLeft == 1 && senseFront == 0 && senseRight == 0)

begin

tempWidthEnA <= 1666666;

tempWidthEnB <= 1666666;

input1reg <= 0;

input2reg <= 1;

input3reg <= 0;

input4reg <= 1;

widthEnA <= tempWidthEnA;

widthEnB <= tempWidthEnB;

tempWidthReverse = 0;

stateTurn = forward;

end

if(senseRight == 1 && senseFront == 0 && senseLeft == 0)

begin

tempWidthEnA <= 1666666;

tempWidthEnB <= 1666666;

input1reg <= 1;

input2reg <= 0;

input3reg <= 1;

input4reg <= 0;

widthEnA <= tempWidthEnA;

widthEnB <= tempWidthEnB;

tempWidthReverse = 0;

stateTurn = forward;

end

end

default:

begin

tempWidthEnA <= 0;

tempWidthEnB <= 0;

input1reg <= 0;

input2reg <= 0;

input3reg <= 0;

input4reg <= 0;

widthEnA <= tempWidthEnA;

widthEnB <= tempWidthEnB;

end

endcase

if (counter <= widthEnB)

tempPWMenB <= 1;

else

tempPWMenB <= 0;

if (counter <= widthEnA)

tempPWMenA <= 1;

else

tempPWMenA <= 0;

end

assign {g, f, e, d, c, b, a} = sseg\_temp;

assign dp = 1'b1;

assign anSS = anSS\_temp;

assign input1 = input1reg;

assign input2 = input2reg;

assign input4 = input4reg;

assign input3 = input3reg;

assign PWMenB = tempPWMenB;

assign PWMenA = tempPWMenA;

endmodule

timescale 1ns / 1ps

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

// Company:

// Engineer:

//

// Create Date: 10/09/2021 02:56:54 PM

// Design Name:

// Module Name: sevenSeg

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

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

module sevenSeg(

input clock,

input senseA,

input senseB,

output input1,

output input2,

output input3,

output input4,

output a, b, c, d, e, f, g, dp, //the individual LED output for the seven segment along with the digital point

output [3:0] anSS

);

localparam N = 16;

reg [N:0] countSS;

reg [1:0] sseg;

reg [3:0] anSS\_temp;

reg [6:0] sseg\_temp;

reg input1reg;

reg input2reg;

reg input3reg;

reg input4reg;

always @ (posedge clock)

begin

countSS <= countSS + 1;

end

always @ (\*)

begin

case(countSS[N:N-1]) //using only the 2 MSB's of the counter

2'b00 : //When the 2 MSB's are 00 enable the fourth display

begin

if(input1reg == 1)

sseg = 2'b1;

else if(input2reg == 1)

sseg = 2'b10;

else

sseg = 2'b0;

anSS\_temp = 4'b1110;

end

2'b01: //When the 2 MSB's are 01 enable the third display

begin

if(input4reg == 1)

sseg = 2'b1;

else if(input3reg == 1)

sseg = 2'b10;

else

sseg = 2'b0;

anSS\_temp = 4'b1101;

end

2'b10: //When the 2 MSB's are 10 enable the second display

begin

if(senseA == 1)

sseg = 2'b11;

else

sseg = 0;

anSS\_temp = 4'b1011;

end

2'b11: //When the 2 MSB's are 11 enable the first display

begin

if(senseB == 1)

sseg = 2'b11;

else

sseg = 0;

anSS\_temp = 4'b0111;

end

endcase

end

always @ (\*)

begin

case(sseg)

2'b1: sseg\_temp = 7'b0001110; //f

2'b10: sseg\_temp = 7'b0000011; //b

2'b11: sseg\_temp = 7'b1000000; //0

default: sseg\_temp = 7'b0111111; //dash

endcase

end

assign input1 = input1reg;

assign input2 = input2reg;

assign input4 = input4reg;

assign input3 = input3reg;

assign {g, f, e, d, c, b, a} = sseg\_temp;

assign dp = 1'b1;

assign anSS = anSS\_temp;

endmodule