Lab04

Altay İlker Yiğitel
CS223 Sec:01
22203024

```
module FourDigitDisplay(Clk, Number, out7, en_out);
   input  Clk;
   input  [4:0] Number;
   input  [4:0] Numbe;
   input  [4:0] Numb;
   input  [4:0] Num;
   output [6:0] out7; //seg a, b, ... g
   output reg [3:0] en_out;

   reg  [3:0] in4;
   reg  [3:0] firstdigit;
   reg  [3:0] seconddigit;
   reg  [3:0] thirddigit;
   reg  [3:0] fourthdigit;
   SevenSegment m1(in4, out7);
   reg  [19:0] cnt;
   always @(posedge Clk) begin
      cnt <= cnt + 1;
   end

   always @(Number) begin
```

```
if (Number < 10)
   begin
      firstdigit <= Number;

   end
 else if (Number < 100)
   begin
      firstdigit <= Number/10;

   end
if (Numbe < 10)
      begin
         seconddigit <= Numbe;

      end
    else if (Numbe < 100)
      begin
         seconddigit <= Numbe/10;

      end
if (Numb < 10)
         begin
            thirddigit <= Numb;
```

```
        end
      else if (Numb < 100)
        begin
          thirddigit <= Numb/10;

        end
if (Num < 10)
          begin
            fourthdigit <= Num;

          end
         else if (Num < 100)
          begin
            fourthdigit <= Num/10;

          end

else
begin
    firstdigit <= 4'b1111;
    seconddigit <= 4'b1111;
    thirddigit <= 4'b1111;
    fourthdigit <= 4'b1111;
end
```

```
    end

    always @(*) begin
        case(cnt[19:17])
            3'b000: begin en_out <= 4'b1110; in4 <= firstdigit; end
            3'b001: begin en_out <= 4'b1110; in4 <= firstdigit; end
            3'b010: begin en_out <= 4'b1101; in4 <= seconddigit; end
            3'b011: begin en_out <= 4'b1101; in4 <= seconddigit; end

            3'b100: begin en_out <= 4'b1011;in4 <= thirddigit; end
            3'b101: begin en_out <= 4'b1011;in4 <= thirddigit; end
            3'b110: begin en_out <= 4'b0111;in4 <= fourthdigit; end
            3'b111: begin en_out <= 4'b0111;in4 <= fourthdigit; end

            default: begin en_out <= 4'b1111; in4 <= 4'b1111; end
        endcase
    end



endmodule



module SevenSegment(numin, segout);
```

```
   input       [3:0] numin;
   output      reg [6:0] segout;
   always @(numin)
   begin
      segout[6] <=   (numin[3]& numin[1]) |(numin[3]& numin[2]) |
                  (numin[2]& ~numin[1]& ~numin[0]) | (~numin[3]&
~numin[2]& ~numin[1]& numin[0]);

      segout[5] <=   (numin[3]&numin[1]) | (numin[3]&numin[2]) |
(numin[2]&~numin[1]&numin[0]) |
                  (numin[2]&numin[1]&~numin[0]);

      segout[4] <=   (numin[3]&numin[1]) | (numin[3]&numin[2]) |
(~numin[2]&numin[1]&~numin[0]);

      segout[3] <=   (numin[3]&numin[1]) | (numin[3]&numin[2]) |
(numin[2]&~numin[1]&~numin[0]) |
                  (~numin[2]&~numin[1]&numin[0]) |
(numin[2]&numin[1]&numin[0]);

      segout[2] <=   (numin[3]&numin[1]) | (numin[3]&numin[2]) |
(~numin[3]&numin[0]) |
                  (~numin[1]&numin[0]) | (numin[2]&~numin[1]);

      segout[1] <=   (numin[3]&numin[2]) | (~numin[2]&numin[1]) |
(numin[1]&numin[0]) |
```

```
                (~numin[3]&~numin[2]&numin[0]);


      segout[0] <=    (numin[3]&numin[1]) | (numin[3]&numin[2]) |
(~numin[3]&~numin[2]&~numin[1])|
                (numin[2]&numin[1]&numin[0]);


   end


endmodule



module multifunction(
   input  [3:0] d,
   input  [1:0] s,
   input  clk,
   input  reset,
   input  shift_in,
   output  [3:0] q
);


   reg [3:0] q_temp;
   always @(posedge clk) begin
      if (reset) begin
         q_temp <= 4'b0000;
      end
```

```
    else begin
      case(s)
        2'b00: begin
          q_temp <= q_temp;
        end
        2'b01: begin
          q_temp <= d;
        end
        2'b10: begin
          q_temp <= {q_temp[2:0], shift_in};
        end
        2'b11: begin
          q_temp <= {shift_in, q_temp[3:1]};
        end
        default: begin
          q_temp <= 4'b0000;
        end
      endcase
    end
  end

  assign q = q_temp;

endmodule
```