NAME : RAAHUL L S

ROLL NO : CB.EN.U4ECE22143

SUBJECT : 19ECE347 – ESLDV

ASSIGNMENT : 02

UP-DOWN COUNTER

CODE:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 30.07.2024 11:41:00
// Design Name:
// Module Name: counter_up_down_7segment
// Project Name:
// Target Devices:
// Tool Versions:
// Description: This module implements a 4-bit up/down counter with a 7-segment
//              display driver. The counter increments or decrements based on
//              the 'up_down' signal. The count value is displayed on the 7-segment
//              display, which is controlled by the 'digit' signal to show different
//              values on the display based on the counter value.
// Dependencies:
//
```

```
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module counter_up_down_7segment(
    input clk,           // Clock signal
    input reset,         // Reset signal
    input up_down,       // Up/Down control signal (1 for up, 0 for down)
    output reg [3:0] an, // Anode signals of the 7-segment display
    output reg [6:0] seg, // Segment signals for 7-segment display
    output led1,         // Output LED1
    output led           // Output LED
);

    reg [3:0] counter = 0; // 4-bit counter
    reg [25:0] count_value; // 26-bit counter register for clock division
    reg [1:0] digit;       // Digit control signal for 7-segment display
    reg clk_out;           // Divided clock output for slower clock
    reg clk_out1;          // Another divided clock output
    integer co_const = 500_000 - 1; // Clock divider constant for 1-second
delay
    reg [25:0] count_value1; // 26-bit counter register for the second clock
division
```

```
    // Clock divider to generate a slower clock signal (1 Hz from 50 MHz input clock)
    always @(posedge clk or posedge reset) begin
        if (reset) begin
            count_value <= 0;
            clk_out <= 0;
        end else if (count_value == 50_000_000 - 1) begin
            count_value <= 0;
            clk_out <= ~clk_out;
        end else begin
            count_value <= count_value + 1;
        end
    end

    // Another clock divider to generate a slower clock signal (50 Hz)
    always @(posedge clk or posedge reset) begin
        if (reset) begin
            count_value1 <= 0;
            clk_out1 <= 0;
        end else if (count_value1 == 1_000_000 - 1) begin
            count_value1 <= 0;
            clk_out1 <= ~clk_out1;
        end else begin
            count_value1 <= count_value1 + 1;
```

```
    end
  end

  // Determine the digit to display based on the count_value1
  always @(posedge clk or posedge reset) begin
    if (reset) begin
      digit <= 2'b00;
    end else if (count_value1 < 250_000) begin
      digit <= 2'b01;
    end else if (count_value1 > 250_000 && count_value1 < 500_000)
begin
      digit <= 2'b10;
    end else if (count_value1 > 500_000 && count_value1 < 750_000)
begin
      digit <= 2'b11;
    end else begin
      digit <= 2'b00;
    end
  end

  // Assign LED outputs based on divided clocks
  assign led = clk_out;
  assign led1 = clk_out1;

  // Increment or decrement the counter based on the up_down signal
```

```
    always @(posedge clk_out or posedge reset) begin
        if (reset) begin
            counter <= 0;
        end else if (up_down) begin
            counter <= counter + 1; // Count up
        end else begin
            counter <= counter - 1; // Count down
        end
    end


    // 7-segment display driver based on the digit control signal and counter value
    always @(digit or counter) begin
        case (digit)
        2'b00 : begin
            an = 4'b0111; // Activate the first 7-segment display
            case (counter[3:0])
            4'b0000: seg = 7'b1000000; // Display 0
            4'b0001: seg = 7'b1111001; // Display 1
            4'b0010: seg = 7'b0100100; // Display 2
            4'b0011: seg = 7'b0110000; // Display 3
            4'b0100: seg = 7'b0011001; // Display 4
            4'b0101: seg = 7'b0010010; // Display 5
            4'b0110: seg = 7'b0000010; // Display 6
            4'b0111: seg = 7'b1111000; // Display 7
```

```
    4'b1000: seg = 7'b0000000; // Display 8
    4'b1001: seg = 7'b0010000; // Display 9
    4'b1010: seg = 7'b0001000; // Display A
    4'b1011: seg = 7'b0000011; // Display B
    4'b1100: seg = 7'b1000110; // Display C
    4'b1101: seg = 7'b0100001; // Display D
    4'b1110: seg = 7'b0000110; // Display E
    4'b1111: seg = 7'b0001110; // Display F
    default: seg = 7'b1001111; // Default to all segments off
    endcase
end

2'b01 : begin
    an = 4'b1011; // Activate the second 7-segment display
    case (counter[2:0])
    4'b0000: seg = 7'b1000000; // Display 0
    4'b0001: seg = 7'b1111001; // Display 1
    4'b0010: seg = 7'b0100100; // Display 2
    4'b0011: seg = 7'b0110000; // Display 3
    4'b0100: seg = 7'b0011001; // Display 4
    4'b0101: seg = 7'b0010010; // Display 5
    4'b0110: seg = 7'b0000010; // Display 6
    4'b0111: seg = 7'b1111000; // Display 7
    4'b1000: seg = 7'b0000000; // Display 8
    4'b1001: seg = 7'b0010000; // Display 9
```

```
    4'b1010: seg = 7'b0001000; // Display A
    4'b1011: seg = 7'b0000011; // Display B
    4'b1100: seg = 7'b1000110; // Display C
    4'b1101: seg = 7'b0100001; // Display D
    4'b1110: seg = 7'b0000110; // Display E
    4'b1111: seg = 7'b0001110; // Display F
    default: seg = 7'b1110011; // Default to all segments off
    endcase
end

2'b10 : begin
    an = 4'b1101; // Activate the third 7-segment display
    case (counter[1:0])
    4'b0000: seg = 7'b1000000; // Display 0
    4'b0001: seg = 7'b1111001; // Display 1
    4'b0010: seg = 7'b0100100; // Display 2
    4'b0011: seg = 7'b0110000; // Display 3
    4'b0100: seg = 7'b0011001; // Display 4
    4'b0101: seg = 7'b0010010; // Display 5
    4'b0110: seg = 7'b0000010; // Display 6
    4'b0111: seg = 7'b1111000; // Display 7
    4'b1000: seg = 7'b0000000; // Display 8
    4'b1001: seg = 7'b0010000; // Display 9
    4'b1010: seg = 7'b0001000; // Display A
    4'b1011: seg = 7'b0000011; // Display B
```

```
    4'b1100: seg = 7'b1000110; // Display C
    4'b1101: seg = 7'b0100001; // Display D
    4'b1110: seg = 7'b0000110; // Display E
    4'b1111: seg = 7'b0001110; // Display F
    default: seg = 7'b1110011; // Default to all segments off
    endcase
end

2'b11 : begin
    an = 4'b1110; // Activate the fourth 7-segment display
    case (counter[0])
    4'b0000: seg = 7'b1000000; // Display 0
    4'b0001: seg = 7'b1111001; // Display 1
    4'b0010: seg = 7'b0100100; // Display 2
    4'b0011: seg = 7'b0110000; // Display 3
    4'b0100: seg = 7'b0011001; // Display 4
    4'b0101: seg = 7'b0010010; // Display 5
    4'b0110: seg = 7'b0000010; // Display 6
    4'b0111: seg = 7'b1111000; // Display 7
    4'b1000: seg = 7'b0000000; // Display 8
    4'b1001: seg = 7'b0010000; // Display 9
    4'b1010: seg = 7'b0001000; // Display A
    4'b1011: seg = 7'b0000011; // Display B
    4'b1100: seg = 7'b1000110; // Display C
    4'b1101: seg = 7'b0100001; // Display D
```

```
            4'b1110: seg = 7'b0000110; // Display E
            4'b1111: seg = 7'b0001110; // Display F
            default: seg = 7'b1111001; // Default to all segments off
            endcase
        end

        endcase
    end

endmodule
```

OUTPUT: