```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 11/08/2019 12:19:39 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 clk,
    input reset,
    input mode_select,
    input inc,
    output [15:0] count,
    output reg [3:0] digit_select,
    output [6:0] display
    );
```

```
wire b_out, f_clk, s_clk;
wire [1:0] dig_sel;
reg [3:0] disp_num;
reg c_inc;

clock_divider_f fc(
    .clk(clk),
    .reset(reset),
    .new_clk(f_clk)
);
clock_divider_s sc(
    .clk(clk),
    .reset(reset),
    .new_clk(s_clk)
);
seven_segment_decoder sd(
    .in(disp_num),
    .display(display)
);
display_control dc(
    .clk(f_clk),
    .reset(reset),
    .out(dig_sel)
);
debouncer d(
    .clk(clk),
    .reset(reset),
    .b_push(c_inc),
    .b_out(b_out)
);
counter16 c(
    .clk(clk),
    .reset(reset),
    .inc(b_out),
    .count(count)
);

always@(posedge clk) begin
    if(mode_select) begin
        c_inc = s_clk;
    end else begin
        c_inc = inc;
    end
end
always@(posedge f_clk, negedge reset) begin
    if(reset) begin
        digit_select <= 4'b0000;
    end else if(f_clk) begin
        case(dig_sel)
            2'b00: begin
                digit_select <= 4'b1110;
                disp_num <= count[3:0];
                end
            2'b01: begin
                digit_select <= 4'b1101;
                disp_num <= count[7:4];
                end
            2'b10: begin
                digit_select <= 4'b1011;
                disp_num <= count[11:8];
                end
            2'b11: begin
                digit_select <= 4'b0111;
                disp_num <= count[15:12];
                end
        endcase
        end
    end
endmodule
```

```
`timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////////////////
//////
// Company:
// Engineer:
//
// Create Date: 11/20/2019 03:32:48 PM
// Design Name:
// Module Name: clock_divider_f
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////
//////


module clock_divider_f(
    input clk,
    input reset,
    output reg new_clk
    );
    reg [31:0] count;
    parameter slow_clk = 50000000;
    parameter fast_clk = 50000;

    always @(posedge clk, negedge reset) begin
```

```
        if(reset) begin
            count <= 32'b0;
            new_clk <= 1'b0;
        end
        else if(count == fast_clk -1)begin
            count <= 32'b0;
            new_clk <= ~new_clk;
        end
        else begin
            count <= count + 32'b1;
            new_clk <= new_clk;
        end
    end
endmodule
```

```
`timescale 1ns / 1ps
///////////////////////////////////////////////////////////////////////////////
/////
// Company:
// Engineer:
//
// Create Date: 11/13/2019 02:30:55 PM
// Design Name:
// Module Name: clock_divider
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
///////////////////////////////////////////////////////////////////////////////
/////


module clock_divider_s(
    input clk,
    input reset,
    output reg new_clk
    );
    reg [31:0] count;
    parameter slow_clk = 50000000;
    parameter fast_clk = 50000;

    always @(posedge clk, negedge reset) begin
```

```
        if(reset) begin
            count <= 32'b0;
            new_clk <= 1'b0;
        end
        else if(count == slow_clk -1)begin
            count <= 32'b0;
            new_clk <= ~new_clk;
        end
        else begin
            count <= count + 32'b1;
            new_clk <= new_clk;
        end
    end
endmodule
```

```
`timescale 1ns / 1ps

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

// Company:

// Engineer:

//

// Create Date: 11/18/2019 01:13:47 PM

// Design Name:

// Module Name: seven_segment_decoder

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

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



module seven_segment_decoder(

        input [3:0] in,

        output reg [6:0] display

    );

     always @(in)begin

            case(in)

                4'b0000: display <= 7'b0000001;

                4'b0001: display <= 7'b1001111;

                4'b0010: display <= 7'b0010010;
        4'b0011: display <= 7'b0000110;
        4'b0100: display <= 7'b1001100;
        4'b0101: display <= 7'b0100100;
        4'b0110: display <= 7'b0100000;
        4'b0111: display <= 7'b0001111;
        4'b1000: display <= 7'b0000000;
        4'b1001: display <= 7'b0000100;
        4'b1010: display <= 7'b0001000;
        4'b1011: display <= 7'b1100000;
        4'b1100: display <= 7'b0110001;
        4'b1101: display <= 7'b1000010;
        4'b1110: display <= 7'b0110000;
        4'b1111: display <= 7'b0111000;
    endcase
end
```

```
`timescale 1ns / 1ps

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

// Company:

// Engineer:

//

// Create Date: 11/18/2019 01:14:56 PM

// Design Name:

// Module Name: display_control

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

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



module display_control(

    input clk,

    input reset,

    output reg [1:0] out

    );



    always @(posedge clk, negedge reset) begin

        if(reset) begin
```

```
            out <= 2'b00;
        end else if(clk) begin
            out <= out + 2'b01;
        end
    end
endmodule
```

```
`timescale 1ns / 1ps

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

// Company:

// Engineer:

//

// Create Date: 11/08/2019 12:20:16 PM

// Design Name:

// Module Name: debouncer

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

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



module debouncer(

    input clk,

    input reset,

    input b_push,

    output reg b_out

    );

    parameter MAX = 1600000;

    reg [31:0] count;
    always @(posedge clk) begin
        if(reset)
            count <= 0;
        else if(b_push)
            count <= count +1;
        else
            count <= 0;
    end

    always @(posedge clk) begin
        if(reset)
            b_out <= 1'b0;
        else if(count == MAX)
            b_out <= 1'b1;
        else
            b_out <= 1'b0;
    end
endmodule
```

```
`timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////////////////
//
// Company:
// Engineer: Ryan Sullivan
//
// Create Date: 11/13/2019 02:26:55 PM
// Design Name:
// Module Name: counter16
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////



module counter16(
    input clk,
    input reset,
    input inc,
    output reg [15:0] count
    );

    always @(posedge clk, negedge reset) begin
        if(reset) begin
```

```
            count <= 16'b0000000000000000;
        end else if(clk) begin
            if(inc) begin
                count <= count + 16'b0000000000000001;
            end
        end
    end
endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////
/////
// Company:
// Engineer: Ryan Sullivan
//
// Create Date: 11/08/2019 12:20:16 PM
// Design Name:
// Module Name: counter
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////
/////


module counter(
    input clk,
    input reset,
    input inc,
    output reg [7:0] count
    );

    always @(posedge clk, negedge reset) begin
        if(reset) begin
```

```
            count <= 8'b00000000;
        end else if(clk) begin
            if(inc) begin
                count <= count + 8'b00000001;
            end
        end
    end
endmodule
```