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

    wire b_out;

    debouncer d(
        .clk(clk),
        .reset(reset),
        .b_push(inc),
        .b_out(b_out)
    );
    counter c(
        .clk(clk),
        .reset(reset),
        .inc(b_out),
        .count(count)
    );
endmodule
```

```
module top2(
    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
```

```
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
```

```
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
```

```
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
```

```
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
```

```
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
```

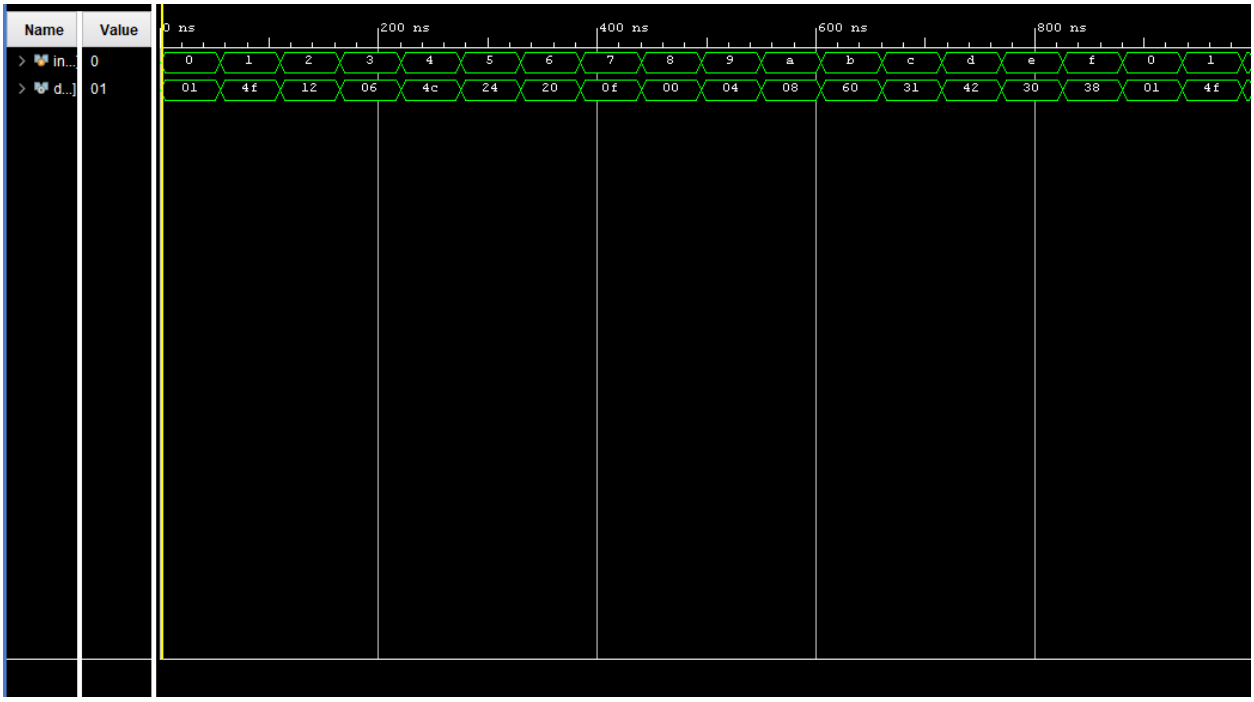
```
module seven_segment_decoder(
        input [3:0] in,
        output reg [6:0] display
    );
     always @(in)begin
            case(in)
                4'b0000: display <= 7'b0000001;
                4'b0001: display <= 7'b1001111;
                4'b0010: display <= 7'b0010010;
                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
endmodule
```

```
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
```

Testbenches:

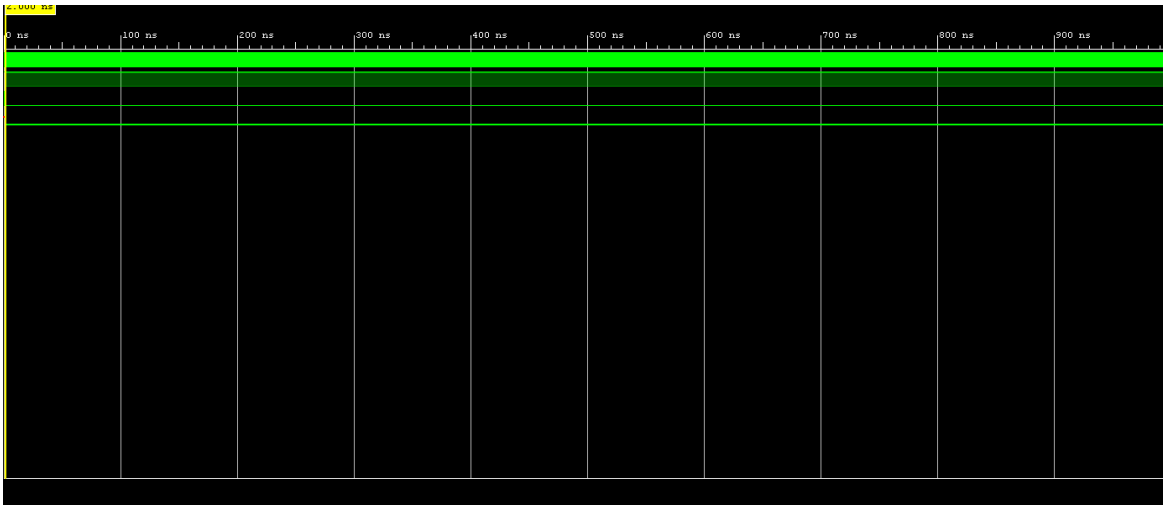
```
module seven_seg_test();
    reg[3:0] in;
    wire[6:0] display;
    seven_segment_decoder s(
        .in(in),
        .display(display)
    );

    initial begin
        in <= 4'b0000;
    end
    always begin
        #55 in <= in + 4'b1;
    end
endmodule
```

Output:



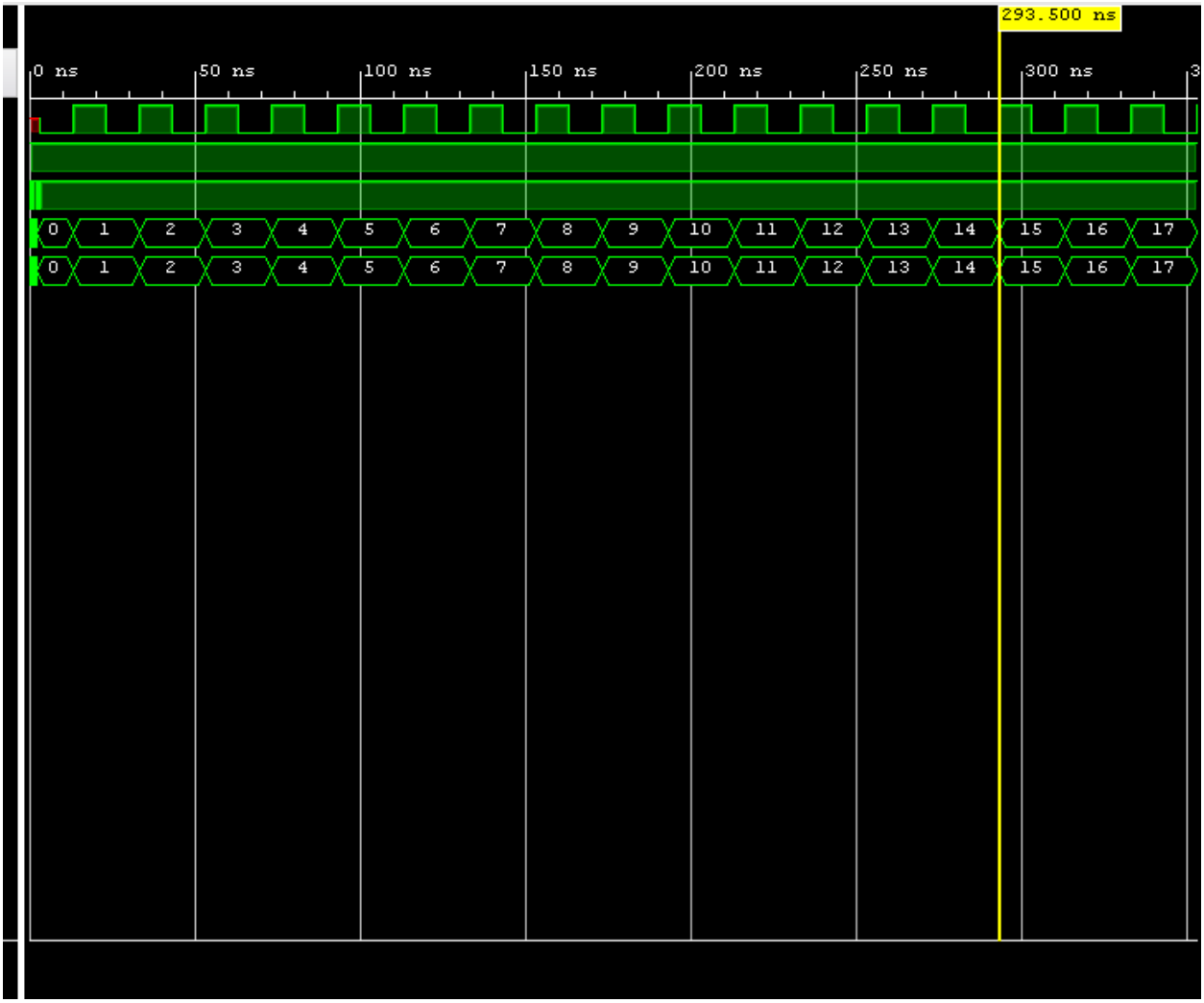
```
module deb_test();
    reg clk, b_in, reset;
    wire b_out;

    debouncer d(
        .clk(clk),
        .b_push(b_in),
        .reset(reset),
        .b_out(b_out)
    );

    initial begin
        b_in <= 1'b1;
        reset <= 1'b1;
        #1 reset = 1'b0;
        clk <= 1'b0;
        forever begin
        #1 clk = ~clk;
        end
        #1;
    end
endmodule
```

Output:



```
module count_test();
    reg clk, inc, reset;
    wire [7:0] count;
    wire [15:0] count2;

    counter c(
        .clk(clk),
        .inc(inc),
        .reset(reset),
        .count(count)
    );
    counter16 c2(
        .clk(clk),
        .inc(inc),
        .reset(reset),
        .count(count2)
    );

    initial begin
        inc <= 1'b1;
        #1 reset <= 1'b1;
        #1 reset <= 1'b0;
        #1 reset <= 1'b1;
        clk <= 1'b0;
        forever begin
        #10 clk = ~clk;
        end
    end
endmodule
```

Output:


293.500 ns
0 ns
50 ns
100 ns
150 ns
200 ns
250 ns
300 ns
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17