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'b000000000000000;

end else if(clk) begin

if(inc) begin

count <= count + 16'b000000000000001;

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

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

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

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