module alarmClock(input CLK, set\_clock, set\_alarm, set\_sec, set\_min, set\_hr, increment, reset, output [6:0] HH, hh, MM, mm, SS, ss);

reg [23:0] clock\_in;

wire [23:0] clock\_out;

reg [23:0] alarm\_time;

reg [23:0] display\_time;

wire clk\_1hz;

reg set\_time=1'b1;

reg alarm\_active=1'b0;

initial begin

clock\_in = 24'h123456;

alarm\_time = 24'h000000;

end

fiftyM\_to\_one\_clk one\_hz\_clk(.clk(CLK), .clk\_out(clk\_1hz));

always @(posedge increment) begin

if (set\_clock && !set\_alarm) begin

if (set\_sec && !set\_min && !set\_hr) begin

if (clock\_out[7:0] == 8'h59)

clock\_in[7:0] = 8'h00;

else if (clock\_out[3:0] == 4'h9) begin

clock\_in[7:4] = clock\_out[7:4] + 4'h1;

clock\_in[3:0] = 4'h0;

end else

clock\_in = clock\_out + 24'h000001;

end else if (!set\_sec && set\_min && !set\_hr) begin

if (clock\_out[15:8] == 8'h59)

clock\_in[15:8] = 8'h00;

else if (clock\_out[11:8] == 4'h9) begin

clock\_in[15:12] = clock\_out[15:12] + 4'h1;

clock\_in[11:8] = 4'h0;

end else

clock\_in = clock\_out + 24'h000100;

end else if (!set\_sec && !set\_min && set\_hr) begin

if (clock\_out[23:16] == 8'h23)

clock\_in[23:16] = 8'h00;

else if (clock\_out[19:16] == 4'h9) begin

clock\_in[23:20] = clock\_out[23:20] + 4'h1;

clock\_in[19:16] = 4'h0;

end else

clock\_in = clock\_out + 24'h010000;

end

end else if (!set\_clock && set\_alarm) begin

if (set\_sec && !set\_min && !set\_hr) begin

if (display\_time[7:0] == 8'h59)

alarm\_time[7:0] = 8'h00;

else if (display\_time[3:0] == 4'h9) begin

alarm\_time[7:4] = alarm\_time[7:4] + 4'h1;

alarm\_time[3:0] = 4'h0;

end else

alarm\_time = display\_time + 24'h000001;

end else if (!set\_sec && set\_min && !set\_hr) begin

if (display\_time[15:8] == 8'h59)

alarm\_time[15:8] = 8'h00;

else if (display\_time[11:8] == 4'h9) begin

alarm\_time[15:12] = display\_time[15:12] + 4'h1;

alarm\_time[11:8] = 4'h0;

end else

alarm\_time = display\_time + 24'h000100;

end else if (!set\_sec && !set\_min && set\_hr) begin

if (display\_time[23:16] == 8'h23)

alarm\_time[23:16] = 8'h00;

else if (display\_time[19:16] == 4'h9) begin

alarm\_time[23:20] = display\_time[23:20] + 4'h1;

alarm\_time[19:16] = 4'h0;

end else

alarm\_time = display\_time + 24'h010000;

end

end else begin

clock\_in = clock\_out;

end

end

always @(posedge CLK) begin

if (set\_clock) begin

set\_time = 1'b1;

display\_time = clock\_out;

end else begin

set\_time = 1'b0; // active clock

if (set\_alarm)

display\_time = alarm\_time;

else

display\_time = clock\_out;

end

end

Clock\_24\_Hour\_behavioral clock\_24(.clk(clk\_1hz), .reset\_time(!reset), .set\_time(set\_time), .inc\_sec(inc\_sec), .time\_in(clock\_in), .time\_out(clock\_out));

hex\_seven Hr(.i(display\_time[23:20]), .hex(HH)),

hr(.i(display\_time[19:16]), .hex(hh)),

Mi(.i(display\_time[15:12]), .hex(MM)),

mi(.i(display\_time[11:8]), .hex(mm)),

Se(.i(display\_time[7:4]), .hex(SS)),

se(.i(display\_time[3:0]), .hex(ss));

endmodule

module hex\_seven(i,hex);

input [3:0] i;

output reg [6:0] hex;

always @ (i)

case (i)

0 : hex <= 7'b1000000;

1 : hex <= 7'b1111001;

2 : hex <= 7'b0100100;

3 : hex <= 7'b0110000;

4 : hex <= 7'b0011001;

5 : hex <= 7'b0010010;

6 : hex <= 7'b0000010;

7 : hex <= 7'b1111000;

8 : hex <= 7'b0000000;

9 : hex <= 7'b0011000;

10 : hex <= 7'b0001000;

11 : hex <= 7'b0000011;

12 : hex <= 7'b1000110;

13 : hex <= 7'b0100001;

14 : hex <= 7'b0000110;

15 : hex <= 7'b0001110;

default : hex <= 7'bx;

endcase

endmodule

module fiftyM\_to\_one\_clk (

input clk,

output reg clk\_out

);

reg [24:0] counter;

always @(posedge clk) begin

if (counter == 24999999) begin // 50 MHz / 2 = 25M for half period

counter <= 0;

clk\_out <= ~clk\_out; // toggle

end else begin

counter <= counter + 1;

end

end

endmodule