```
 1  module timer (HEX0, HEX1, HEX2, HEX3, CLOCK_50, timer_enable);
 2     input CLOCK_50;
 3     output [6:0] HEX0;
 4     output [6:0] HEX1;
 5     output [6:0] HEX2;
 6     output [6:0] HEX3;
 7     reg [1:0] Sel;
 8     wire [3:0] seconds, seconds2, minutes, minutes2;
 9     wire enable;
10     wire [25:0] counter;
11     input timer_enable;
12
13     wire  [25:0] upperBound;
14     assign upperBound = 26'b01011111010111100000111111;
15
16     rateDividerForTime r(CLOCK_50, upperBound, enable,counter);
17     fourbitcounter c(enable, timer_enable, CLOCK_50, seconds, seconds2, minutes, minutes2);
18
19
20     hexDisplay H0(seconds[3:0], HEX0);
21     hexDisplay H1(seconds2[3:0], HEX1);
22     hexDisplay H2(minutes[3:0], HEX2);
23     hexDisplay H3(minutes2[3:0], HEX3);
24
25
26  endmodule
27
28  module rateDividerForTime(input clock, input [25:0] upperBound, output reg enable, output
    reg [25:0] counter);
29     always @(posedge clock)
30     begin
31        if (counter === 26'bx)
32        begin
33           counter <= 26'b0;
34        end
35        else if (counter == upperBound)
36        begin
37           enable= 1'b1;
38           counter <= 26'b0;
39        end
40        else
41        begin
42           enable = 1'b0;
43           counter <= counter + 1 ;
44        end
45     end
46
47
48  endmodule
49
50  module fourbitcounter(input enable, input timer_enable, clock, output reg [3:0] seconds,
    output reg [3:0] seconds2, output reg [3:0] minutes, output reg [3:0] minutes2);
51
52     initial minutes2 = 0;
53
54     always @(posedge clock)
55     begin
56        if ((enable == 1'b1) && (timer_enable == 1'b1))
57        begin
58           seconds <= seconds + 1;
59           if ((seconds % 9) == 0 && (seconds != 0)) begin
60              seconds <= 0 ;
61              seconds2 <= seconds2 + 1;
62           end
63           if ((seconds2 == 5) && (seconds == 9)) begin
64              seconds2 <= 0;
65              seconds <= 0;
66              minutes <= minutes + 1;
67           end
68           if (((minutes % 9) == 0) && (minutes != 0) && (seconds2 == 5) && (seconds == 9))
    begin
69              seconds <= 0;
70              seconds2 <= 0;
```

```
71                minutes <= 0;
72                minutes2 <= minutes2 + 1;
73             end
74          end
75          else if (timer_enable == 1'b0) begin
76             seconds <= 0;
77             seconds2 <= 0;
78             minutes <= 0;
79             minutes2 <= 0;
80          end
81
82       end
83
84    endmodule
85
86
87    module hexDisplay(input [3:0] SW, output [6:0] HEX0);
88       reg c0,c1,c2,c3;
89       always @(*)
90          begin
91             c3=SW[3];
92             c2=SW[2];
93             c1=SW[1];
94             c0=SW[0];
95          end
96
97       assign HEX0[0]=~((c3|c2|c1|~c0)&(c3|~c2|c1|c0)&(~c3|c2|~c1|~c0)&(~c3|~c2|c1|~c0));
98       assign HEX0[1]=~((c3|~c2|c1|~c0)& (c3|~c2|~c1|c0)&(~c3|c2|~c1|~c0)&(~c3|~c2|c1|c0)&(~c3|~
      c2|~c1|c0)& (~c3|~c2|~c1|~c0));
99       assign HEX0[2]=~((c3|c2|~c1|c0)&(~c3|~c2|c1|c0)&(~c3|~c2|~c1|c0)&(~c3|~c2|~c1|~c0));
100      assign HEX0[3]=~((c3|c2|c1|~c0)&(c3|~c2|c1|c0)&(c3|~c2|~c1|~c0)&(~c3|c2|~c1|c0)&(~c3|~c2
      |~c1|~c0));
101      assign HEX0[4]=~((c3|c2|c1|~c0)&(c3|c2|~c1|~c0)&(c3|~c2|c1|c0)&(c3|~c2|c1|~c0)&(c3|~c2|~
      c1|~c0)&(~c3|c2|c1|~c0));
102      assign HEX0[5]=~((c3|c2|c1|~c0)&(c3|c2|~c1|c0)&(c3|c2|~c1|~c0)&(c3|~c2|~c1|~c0)&(~c3|~c2|
      c1|~c0));
103      assign HEX0[6]=~((c3|c2|c1|c0)&(c3|c2|c1|~c0)&(c3|~c2|~c1|~c0)& (~c3|~c2|c1|c0));
104   endmodule
105
```