```
1   module DisplayCounter(SW, HEX0, CLOCK_50);
2      input [9:0] SW;
3      input CLOCK_50;
4      output [6:0] HEX0;
5
6      wire [3:0] Q;
7      wire enable;
8      wire [27:0] upperBound, counter;
9
10     reg reset;
11     reg [1:0] Sel;
12
13     always @(*)
14     begin
15        reset= SW[9];
16        Sel = SW[1:0];
17     end
18
19     GetFreq f(Sel, upperBound);
20     RateDivider r(CLOCK_50, upperBound, enable,counter);
21     counter4b c(enable, CLOCK_50, reset, Q);
22     HexDecoder h0(Q, HEX0);
23  endmodule
24
25  module GetFreq(SW, upperBound);
26     input [1:0] SW;
27     output reg [27:0] upperBound;
28     always @(*)
29        case(SW[1:0])
30           2'b00: upperBound = 27'b0;
31           2'b01: upperBound = 27'b001011111010111100000111111 ;
32           2'b10: upperBound = 27'b010111111010111100000111111 ;
33           2'b11: upperBound = 27'b101111101011110000011111111 ;
34        endcase
35  endmodule
36
37  module RateDivider(clk, upperBound, enable, counter);
38     input clk;
39     input [27:0] upperBound;
40     output reg enable;
41     output reg [27:0] counter;
42     always @(posedge clk)
43     begin
44        if (counter === 27'bx)
45        begin
46           counter <= 27'b0;
47        end
48        else if (counter == upperBound)
49        begin
50           enable= 1'b1;
51           counter <= 27'b0;
52        end
53        else
54        begin
55           enable = 1'b0;
56           counter <= counter + 1 ;
57        end
58     end
59
60  endmodule
61
62  module counter4b(enable, clk, reset, q);
63     input enable, clk, reset;
64     output reg [3:0] q;
65     always @(posedge clk)
66     begin
67        if (reset == 1'b1)
68        begin
69           q <= 0;
70        end
71        else if (enable == 1'b1)
72        begin
73           q <= q + 1;
74        end
75     end
76  endmodule
```

```
77
78
79    module HexDecoder (In, HEX);
80       input [3:0] In;
81       output [6:0] HEX;
82
83       assign c0 = In[0];
84       assign c1 = In[1];
85       assign c2 = In[2];
86       assign c3 = In[3];
87
88       assign HEX[0] = (~c3 & ~c2 & ~c1 & c0) + (~c3 & c2 & ~c1 & ~c0) + (c3 & ~c2 & c1 & c0) +
      (c3 & c2 & ~c1 & c0);
89       assign HEX[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);
90       assign HEX[2] = (~c3 & ~c2 & c1 & ~c0) + (c3 & c2 & ~c1 & ~c0) + (c3 & c2 & c1 & ~c0) + (
      c3 & c2 & c1 & c0);
91       assign HEX[3] = (~c3 & ~c2 & ~c1 & c0) + (~c3 & c2 & ~c1 & ~c0) + (~c3 & c2 & c1 & c0) +
      (c3 & ~c2 & ~c1 & c0) + (c3 & ~c2 & c1 & ~c0) + (c3 & c2 & c1 & c0);
92       assign HEX[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);
93       assign HEX[5] = (~c3 & ~c2 & ~c1 & c0) + (~c3 & ~c2 & c1 & ~c0) + (~c3 & ~c2 & c1 & c0) +
       (~c3 & c2 & c1 & c0) + (c3 & c2 & ~c1 & c0);
94       assign HEX[6] = (~c3 & ~c2 & ~c1 & ~c0) + (~c3 & ~c2 & ~c1 & c0) + (~c3 & c2 & c1 & c0) +
       (c3 & c2 & ~c1 & ~c0);
95    endmodule
96
```