```
1   module Counter8b (SW, KEY, HEX0, HEX1);
2      input [1:0] SW;
3      input [1:0] KEY;
4      output [6:0] HEX0, HEX1;
5
6      wire [6:0] ANDGate;
7      wire [7:0] Qout;
8
9      TFlipFlop ff0(SW[1], KEY[0], SW[0], Qout[0]);
10     assign ANDGate[0] = SW[1] & Qout[0];
11
12     TFlipFlop ff1(ANDGate[0], KEY[0], SW[0], Qout[1]);
13     assign ANDGate[1] = ANDGate[0] & Qout[1];
14
15     TFlipFlop ff2(ANDGate[1], KEY[0], SW[0], Qout[2]);
16     assign ANDGate[2] = ANDGate[1] & Qout[2];
17
18     TFlipFlop ff3(ANDGate[2], KEY[0], SW[0], Qout[3]);
19     assign ANDGate[3] = ANDGate[2] & Qout[3];
20
21     TFlipFlop ff4(ANDGate[3], KEY[0], SW[0], Qout[4]);
22     assign ANDGate[4] = ANDGate[3] & Qout[4];
23
24     TFlipFlop ff5(ANDGate[4], KEY[0], SW[0], Qout[5]);
25     assign ANDGate[5] = ANDGate[4] & Qout[5];
26
27     TFlipFlop ff6(ANDGate[5], KEY[0], SW[0], Qout[6]);
28     assign ANDGate[6] = ANDGate[5] & Qout[6];
29
30     TFlipFlop ff7(ANDGate[6], KEY[0], SW[0], Qout[7]);
31
32     HexDecoder h0(Qout[3:0], HEX0);
33     HexDecoder h1(Qout[7:4], HEX1);
34  endmodule
35
36
37  module TFlipFlop (T, clk, reset, Q);
38     input T, clk, reset;
39     output reg Q;
40
41     always @(posedge clk)
42     begin
43        if (reset)
44           Q <= 1'b0;
45        else
46           Q <= Q ^ T;
47     end
48  endmodule
49
50  module HexDecoder (In, HEX);
51     input [3:0] In;
52     output [6:0] HEX;
53
54     assign c0 = In[0];
55     assign c1 = In[1];
56     assign c2 = In[2];
57     assign c3 = In[3];
58
59     assign HEX[0] = (~c3 & ~c2 & ~c1 & c0) + (~c3 & c2 & ~c1 & ~c0) + (c3 & ~c2 & c1 & c0) +
   (c3 & c2 & ~c1 & c0);
60
61     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);
62
63     assign HEX[2] = (~c3 & ~c2 & c1 & ~c0) + (c3 & c2 & ~c1 & ~c0) + (c3 & c2 & c1 & ~c0) + (
   c3 & c2 & c1 & c0);
64
65     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);
66
67     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);
68
69     assign HEX[5] = (~c3 & ~c2 & ~c1 & c0) + (~c3 & ~c2 & c1 & ~c0) + (~c3 & ~c2 & c1 & c0) +
    (~c3 & c2 & c1 & c0) + (c3 & c2 & ~c1 & c0);
70
```

```
71      assign HEX[6] = (~c3 & ~c2 & ~c1 & ~c0) + (~c3 & ~c2 & ~c1 & c0) + (~c3 & c2 & c1 & c0) +
       (c3 & c2 & ~c1 & ~c0);
72    endmodule
73
74
```