```
1   `timescale 1ns / 1ns // `timescale time_unit/time_precision
2
3   // Morse code encoder with LED flash output
4   module MorseEncoder(input[9:0] SW, input[3:0] KEY, input CLOCK_50, output[9:0] LEDR);
5       wire[2:0] letterSelect;
6       wire PLoad,reset;
7
8       wire[12:0] MorseCode;
9       wire enableClock;
10      wire[24:0] rDivider;
11      wire flash;
12
13      // Input assignment
14      assign letterSelect = SW[2:0];
15      assign PLoad = KEY[1];
16      assign reset = KEY[0];
17
18      // Letter to Morse Code LUT
19      MorseTable MT0(.letterSelect(letterSelect),.MorseCode(MorseCode));
20
21      // Clock ever 0.5 seconds
22      rateDivider rD0(.clock(CLOCK_50),.reset(reset),.counter(rDivider));
23      assign enableClock = (rDivider==0);
24
25      // Shift register to display the morse code
26      shiftRegister reg0(.clock(CLOCK_50),.reset(reset),.PL(PLoad),
27                         .enable(enableClock),.DATA_IN(MorseCode),.leftBit(flash));
28
29      // Output
30      assign LEDR[0] = flash;
31  endmodule
32
33  // Left 13-bit shift register, inserts a 0 onto the right
34  // Active high async reset, active high async parallel load
35  module shiftRegister(input [12:0] DATA_IN, input reset,clock,PL,enable, output reg
    leftBit);
36      reg[12:0] Q;
37
38      always @(posedge clock)
39      begin
40          if(reset) // Async reset
41          begin
42              Q <= 0;
43              leftBit <= 0;
44          end
45          else if(PL) // Async parallel load
46              Q <= DATA_IN;
47          else if(enable) // 0.5s clock pulse
48          begin // Do the shifting
49              leftBit <= Q[12];
50              Q[12] <= Q[11];
51              Q[11] <= Q[10];
52              Q[10] <= Q[9];
53              Q[9] <= Q[8];
54              Q[8] <= Q[7];
55              Q[7] <= Q[6];
56              Q[6] <= Q[5];
57              Q[5] <= Q[4];
58              Q[4] <= Q[3];
59              Q[3] <= Q[2];
60              Q[2] <= Q[1];
61              Q[1] <= Q[0];
62              Q[0] <= 0;
63          end
64      end
65  endmodule
66
67
68
```

```
69  // LUT for the Morse code of the selected letter (S-Z)
70  module MorseTable(input[2:0] letterSelect, output reg[12:0] MorseCode);
71      always @(*)
72      begin
73          case(letterSelect)
74              0: // S
75                  MorseCode = 13'b1010100000000;
76              1: // T
77                  MorseCode = 13'b1110000000000;
78              2: // U
79                  MorseCode = 13'b1010111000000;
80              3: // V
81                  MorseCode = 13'b1010101110000;
82              4: // W
83                  MorseCode = 13'b1011101110000;
84              5: // X
85                  MorseCode = 13'b1110101011100;
86              6: // Y
87                  MorseCode = 13'b1110101110111;
88              7: // Z
89                  MorseCode = 13'b1110111010100;
90              default: // Latch prevention
91                  MorseCode = 0;
92          endcase
93      end
94  endmodule
95
96  // Active high, synchronous reset positive edge triggered counter with parallel load
97  // Counts down from 25M-1 to 0, 25M is 25 bits
98  module rateDivider(input clock,reset, output reg[24:0] counter);
99      // When simulating use divisor = 1000000
100     // Uploading to FPGA use divisor = 1
101     // Used because simulating a 50 Mhz clock means a very long run time
102     parameter divisor = 1;
103
104     always @(posedge clock)
105     begin
106         if(reset) // Active high reset to loadIn
107             counter <= 25000000/divisor-1;
108         else if(counter==0) // Reset to initial value (1 cycle)
109             counter <= 25000000/divisor-1;
110         else // Count down
111             counter <= counter-1;
112     end
113 endmodule
114
```