

```
1 //=====
2 //
3 // encoder functions
4 //
5 // Written by:
6 //           Kenneth Maxon 16/Mar/04
7 //
8 //=====
9
10 module encoder_func #(parameter counter_width = 8) (
11     input wire sys_clock,
12     output reg [counter_width - 1:0] encoder_data_out,
13     input wire encoder_a,
14     input wire encoder_b
15 );
16
17 // -----parameters
18
19 parameter [3:0]
20     ENC_A = 2'b00,
21     ENC_B = 2'b01,
22     ENC_C = 2'b10,
23     ENC_D = 2'b11;
24
25 parameter [3:0]
26     STATE_A = 4'b0001,
27     STATE_B = 4'b0010,
28     STATE_C = 4'b0100,
29     STATE_D = 4'b1000;
30
31 parameter [3:0]
32     CASE_A = 4'bxxx1,
33     CASE_B = 4'bxx1x,
34     CASE_C = 4'bx1xx,
35     CASE_D = 4'b1xxx;
36
37 // -----local var
38
39 reg [counter_width - 1:0] counter;
40 reg [3:0] enc_state;
41 reg encoder_reg_a;
42 reg encoder_reg_b;
43 wire [1:0] encoders = {encoder_reg_a,encoder_reg_b}; //grouped together
44
45 // register signals crossing clock domains.
46
47 always @(posedge sys_clock)
48 begin
49     encoder_reg_a <= #1 encoder_a;
50     encoder_reg_b <= #1 encoder_b;
51     encoder_data_out[counter_width - 1:0] <= #1 counter[counter_width - 1:0];
52 end
53
54 always @(posedge sys_clock)
55 begin
56     casex (enc_state) //synopsys full_case parallel_case
57         CASE_A:
58             begin
59                 if(encoders[1:0] == ENC_B)
60                     begin
61                         counter[counter_width - 1:0] <= #1 counter[counter_width - 1:0] + 1'b1;
62                         enc_state <= #1 STATE_B;
63                     end
64                 if(encoders[1:0] == ENC_D)
65                     begin
66                         counter[counter_width - 1:0] <= #1 counter[counter_width - 1:0] - 1'b1;
67                         enc_state <= #1 STATE_D;
68                     end
69             end
70
71         CASE_B:
72             begin
73                 if(encoders[1:0] == ENC_C)
74                     begin
75                         counter[counter_width - 1:0] <= #1 counter[counter_width - 1:0] + 1'b1;
76                         enc_state <= #1 STATE_C;
77                     end
78                 if(encoders[1:0] == ENC_A)
79                     begin
80                         counter[counter_width - 1:0] <= #1 counter[counter_width - 1:0] - 1'b1;
```

```
81         enc_state <= #1 STATE_A;
82     end
83 end
84
85 CASE_C:
86 begin
87     if(encoders[1:0] == ENC_D)
88     begin
89         counter[counter_width - 1:0] <= #1 counter[counter_width - 1:0] + 1'b1;
90         enc_state <= #1 STATE_D;
91     end
92     if(encoders[1:0] == ENC_B)
93     begin
94         counter[counter_width - 1:0] <= #1 counter[counter_width - 1:0] - 1'b1;
95         enc_state <= #1 STATE_B;
96     end
97 end
98
99 CASE_D:
100 begin
101     if(encoders[1:0] == ENC_A)
102     begin
103         counter[counter_width - 1:0] <= #1 counter[counter_width - 1:0] + 1'b1;
104         enc_state <= #1 STATE_A;
105     end
106     if(encoders[1:0] == ENC_C)
107     begin
108         counter[counter_width - 1:0] <= #1 counter[counter_width - 1:0] - 1'b1;
109         enc_state <= #1 STATE_C;
110     end
111 end
112
113 endcase
114 end
115
116 endmodule
117
```