

```
1  `timescale 1ns / 1ps
2  ///////////////////////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date:    12:47:59 06/25/2021
7  // Design Name:
8  // Module Name:    I2C_slave
9  // Project Name:
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 ///////////////////////////////////////////////////////////////////
21 module I2C_Slave(
22     inout sda,
23     inout scl
24 );
25
26     localparam ADDRESS = 7'b1000100;
27
28     localparam READ_ADDR = 0;
29     localparam SEND_ACK = 1;
30     localparam READ_DATA = 2;
31     localparam WRITE_DATA = 3;
32     localparam SEND_ACK2 = 4;
33
34     reg [7:0] addr;
35     reg [7:0] counter;
36     reg [7:0] state = 0;
37     reg [7:0] data_in = 0;
38     reg [7:0] data_out = 8'b11001100;
39     reg sda_out = 0;
40     reg sda_in = 0;
41     reg start = 0;
42     reg write_enable = 0;
43
44     assign sda = (write_enable == 1) ? sda_out : 'bz;
45
46     always @(negedge sda) begin
47         if ((start == 0) && (scl == 1)) begin
48             start <= 1;
49             counter <= 7;
50         end
51     end
52
53     always @(posedge sda) begin
54         if ((start == 1) && (scl == 1)) begin
55             state <= READ_ADDR;
56             start <= 0;
57             write_enable <= 0;
```

```
58     end
59 end
60
61 always @(posedge scl) begin
62     if (start == 1) begin
63         case(state)
64             READ_ADDR: begin
65                 addr[counter] <= sda;
66                 if(counter == 0) state <= SEND_ACK;
67                 else counter <= counter - 1;
68             end
69
70             SEND_ACK: begin
71                 if(addr[7:1] == ADDRESS) begin
72                     counter <= 7;
73                     if(addr[0] == 0) begin
74                         state <= READ_DATA;
75                     end
76                     else state <= WRITE_DATA;
77                 end
78             end
79
80             READ_DATA: begin
81                 data_in[counter] <= sda;
82                 if(counter == 0) begin
83                     state <= SEND_ACK2;
84                 end else counter <= counter - 1;
85             end
86
87             SEND_ACK2: begin
88                 state <= READ_ADDR;
89             end
90
91             WRITE_DATA: begin
92                 if(counter == 0) state <= READ_ADDR;
93                 else counter <= counter - 1;
94             end
95         endcase
96     end
97 end
98
99
100 always @(negedge scl) begin
101     case(state)
102
103         READ_ADDR: begin
104             write_enable <= 0;
105         end
106
107         SEND_ACK: begin
108             sda_out <= 0;
109             write_enable <= 1;
110         end
111
112         READ_DATA: begin
113             write_enable <= 0;
114         end
```

```
115
116         WRITE_DATA: begin
117             sda_out <= data_out[counter];
118             write_enable <= 1;
119         end
120
121         SEND_ACK2: begin
122             sda_out <= 0;
123             write_enable <= 1;
124         end
125     endcase
126 end
127 endmodule
128
```