```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL;


ENTITY I2C IS
PORT (
    i2c_busy:   OUT     STD_ULOGIC;
    i2c_scl:    OUT     STD_ULOGIC;
    i2c_send_flag:  IN  STD_ULOGIC;
    i2C_sda:    INOUT   STD_ULOGIC;
    i2c_addr:   IN  STD_ULOGIC_VECTOR (7 DOWNTO 0);
    i2c_done:   OUT STD_ULOGIC;
    i2c_data:   IN  STD_ULOGIC_VECTOR (15 DOWNTO 0);
    i2c_clock_50:   IN  STD_ULOGIC
);
END i2c;



ARCHITECTURE structure of I2C IS

SIGNAL  i2c_clk_en: STD_ULOGIC          :=  '0';
SIGNAL  clk_prs:    NATURAL RANGE 0 TO 300  := 0;
SIGNAL  clk_en:     STD_ULOGIC          := '0';
SIGNAL  ack_en:     STD_ULOGIC          := '0';
SIGNAL  clk_i2c:    STD_ULOGIC          := '0';
SIGNAL  get_ack:    STD_ULOGIC          := '0';
SIGNAL  data_index: NATURAL RANGE 0 TO 15   := 0;
TYPE        fsm IS (st0, st1, st2, st3, st4, st5, st6, st7, st8);
SIGNAL  i2c_fsm:    fsm                     := st0;

BEGIN


PROCESS (i2c_clock_50)
BEGIN

    IF i2c_clock_50'EVENT AND i2c_clock_50 = '1' THEN

        IF (clk_prs < 250) THEN
            clk_prs <= clk_prs + 1;
        ELSE
            clk_prs <= 0;
        END IF;



        -- 50% duty cycle clock for i2c
        IF (clk_prs < 125) THEN
            clk_i2c <= '1';
        ELSE
            clk_i2c <= '0';
        END IF;



        -- clock for ack on SCL = HIGH
        IF (clk_prs = 62) THEN
            ack_en <= '1';
        ELSE
            ack_en <= '0';
        END IF;



        -- clock for data on SCL = LOW
        IF (clk_prs = 187) THEN
            clk_en <= '1';
        ELSE
            clk_en <= '0';
        END IF;

```

```
74      END IF;
75
76
77
78      IF i2c_clock_50'EVENT AND i2c_clock_50 = '1' THEN
79
80          IF (i2c_clk_en = '1') THEN
81              i2c_scl <= clk_i2c;
82          ELSE
83              i2c_scl <= '1';
84          END IF;
85
86
87          -- ack on SCL = HIGH
88          IF (ack_en = '1') THEN
89
90              CASE i2c_fsm IS
91
92                  WHEN st3 => -- get ack
93
94                          IF (i2c_sda = '0') THEN
95                              i2c_fsm <= st4;         --ack
96                              data_index <= 15;
97                          ELSE
98                              i2c_clk_en <= '0';
99                              i2c_fsm <= st0;         --nack
100                         END IF;
101
102
103                 WHEN st5 => -- get ack
104
105
106                         IF (i2c_sda = '0') THEN
107                             i2c_fsm <= st6;         --ack
108                             data_index <= 7;
109                         ELSE
110                             i2c_fsm <= st0;         --nack
111                             i2c_clk_en <= '0';
112                         END IF;
113
114
115                 WHEN st7 => -- get ack
116
117
118                         IF (i2c_sda = '0') THEN
119                             i2c_fsm <= st0;         -- ack
120                         ELSE
121                             i2c_fsm <= st0;         --nack
122                             i2c_clk_en <= '0';
123                         END IF;
124
125
126                 WHEN OTHERS => NULL;
127             END CASE;
128         END IF;
129
130
131         IF (clk_en = '1') THEN
132
133             CASE i2c_fsm IS
134
135                 WHEN st0 => -- stand by
136
137                     i2c_sda <= '1';
138                     i2c_busy <= '0';
139                     i2c_done <= '0';
140
141                     IF (i2c_send_flag = '1') THEN
142                         i2c_fsm <= st1;
143                         i2c_busy <= '1';
144                     END IF;
145
146
```

```
147                 WHEN st1 => -- start condition
148
149                     i2c_sda <= '0';
150                     i2c_fsm <= st2;
151                     data_index <= 7;
152
153
154
155                 WHEN st2 => -- send addr
156
157                     i2c_clk_en <= '1'; --start clocking i2c_scl
158
159                     IF (data_index > 0) THEN
160                         data_index <= data_index - 1;
161                         i2c_sda <= i2c_addr(data_index);
162                     ELSE
163                         i2c_sda <= i2c_addr(data_index);
164                         get_ack <= '1';
165                     END IF;
166
167                     IF (get_ack = '1') THEN
168                         get_ack <= '0';
169                         i2c_fsm <= st3;
170                         i2c_sda <= 'Z';
171                     END IF;
172
173
174                 WHEN st4 => -- send 1st 8 bit
175
176
177                     IF (data_index > 8) THEN
178                         data_index <= data_index - 1;
179                         i2c_sda <= i2c_data(data_index);
180                     ELSE
181                         i2c_sda <= i2c_data(data_index);
182                         get_ack <= '1';
183                     END IF;
184
185                     IF (get_ack = '1') THEN
186                         get_ack <= '0';
187                         i2c_fsm <= st5;
188                         i2c_sda <= 'Z';
189                     END IF;
190
191
192                 WHEN st6 => -- send 2nd 8 bit
193
194
195                     IF (data_index >0) THEN
196                         data_index <= data_index - 1;
197                         i2c_sda <= i2c_data(data_index);
198                     ELSE
199                         i2c_sda <= i2c_data(data_index);
200                         get_ack <= '1';
201                     END IF;
202
203                     IF (get_ack = '1') THEN
204                         get_ack <= '0';
205                         i2c_fsm <= st7;
206                         i2c_sda <= 'Z';
207                     END IF;
208
209
210                 WHEN st8 => -- stop condition
211
212
213                     i2c_clk_en <= '0';
214                     i2c_sda <= '0';
215                     i2c_fsm <= st0;
216                     i2c_done <= '1';
217
218
219                 WHEN OTHERS => NULL;
```

```
220             END CASE;
221         END IF;
222
223     END IF;
224     END PROCESS;
225     END structure;
```