```vhdl
--345678901234567890123456789012345678901234567890123456789012345678901234567890
-- $Id: transmitter.vhd,v 1.2 2004/12/08 22:52:28 tofp Exp $
--*****************************************************************************
--*  TRANSMITTER.VHD
--*
--*
--*  REVISION HISTORY:
--*    12-Oct-2001 CS  Original coding
--*     2-May-2002 CS  TXINGAP before TXDATA (fix trigger problem)
--*    31-Oct-2002 CS  Internal memory is added to the data path
--*
--*****************************************************************************

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY transmitter IS
  PORT (
    clock      : IN  std_logic;
    arstn      : IN  std_logic;
    trigger    : IN  std_logic;
    gap_active : OUT std_logic;
    block_read : IN  std_logic;
    event_read : IN  std_logic;
    reg_read   : IN  std_logic;
    reg_addr   : IN  std_logic_vector ( 5 DOWNTO 0);
    tid        : IN  std_logic_vector ( 3 DOWNTO 0);
    ps_reg     : IN  std_logic_vector ( 7 DOWNTO 0);
    bl_reg     : IN  std_logic_vector ( 7 DOWNTO 0);
    dt_reg     : IN  std_logic_vector ( 7 DOWNTO 0);
    fc_reg     : IN  std_logic_vector ( 7 DOWNTO 0);
    te_reg     : IN  std_logic_vector ( 7 DOWNTO 0);
    xx_reg     : IN  std_logic_vector ( 7 DOWNTO 0);
    pg_dout    : IN  std_logic_vector (32 DOWNTO 0);
    pg_doutval : IN  std_logic;
    pg_enable  : OUT std_logic;
    im_dout    : IN  std_logic_vector (32 DOWNTO 0);
    im_doutval : IN  std_logic;
    im_enable  : OUT std_logic;
    foD        : OUT std_logic_vector (31 DOWNTO 0);
    foTEN_N    : OUT std_logic;
    foCTRL_N   : OUT std_logic;
    fiDIR      : IN  std_logic;
    fiBEN_N    : IN  std_logic;
    fiLF_N     : IN  std_logic
    );
END transmitter;

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE work.my_conversions.ALL;
USE work.my_utilities.ALL;

ARCHITECTURE SYN OF transmitter IS

  CONSTANT FESTW : std_logic_vector := "01000100";

  TYPE output_state IS (
    OS_IDLE,
    OS_WAITOUT,
    OS_TXSTATUS,
    OS_TXDATA,
    OS_TXINGAP,
    OS_WAITIN
    );

```

```
67      TYPE feebus_state IS (
68        FB_INPUT,
69        FB_FLOAT,
70        FB_OUTPUT,
71        FB_RESET
72        );
73
74    BEGIN
75
76      main : PROCESS (clock, arstn)
77
78        VARIABLE datao          : std_logic_vector (32 DOWNTO 0);
79        VARIABLE datao_valid    : std_logic;
80        VARIABLE st_dout        : std_logic_vector (32 DOWNTO 0);
81        VARIABLE b_block_end    : boolean;
82        VARIABLE reg_read_req   : std_logic;
83        VARIABLE output_present : output_state;
84        VARIABLE output_next    : output_state;
85        VARIABLE feebus_present : feebus_state;
86        VARIABLE feebus_next    : feebus_state;
87
88      BEGIN
89        IF (arstn = '0') THEN
90
91          gap_active <= '0';
92          pg_enable  <= '0';
93          im_enable  <= '0';
94          foD        <= (OTHERS => '0');
95          foTEN_N    <= '1';
96          foCTRL_N   <= '1';
97
98          datao          := (OTHERS => '0');
99          datao_valid    := '0';
100         st_dout        := (OTHERS => '0');
101         b_block_end    := false;
102         reg_read_req   := '0';
103         output_present := OS_IDLE;
104         output_next    := OS_IDLE;
105         feebus_present := FB_INPUT;
106         feebus_next    := FB_INPUT;
107
108       ELSIF (clock'event AND clock = '1') THEN
109
110         IF (output_present = OS_TXSTATUS) THEN
111           datao       := st_dout;
112           datao_valid := '1';
113           b_block_end := false;
114         ELSIF (output_present = OS_TXDATA) THEN
115           IF block_read = '1' THEN
116             datao       := im_dout;
117             datao_valid := im_doutval;
118             b_block_end := (im_dout(32) = '1' AND im_doutval = '1');
119           ELSE
120             datao       := pg_dout;
121             datao_valid := pg_doutval;
122             b_block_end := (pg_dout(32) = '1' AND pg_doutval = '1');
123           END IF;
124         ELSE
125           datao       := (OTHERS => '0');
126           datao_valid := '0';
127         END IF;
128
129         foD      <= datao(31 DOWNTO 0);
130         foCTRL_N <= NOT (datao_valid AND datao(32));
131         foTEN_N  <= NOT (datao_valid);
132
```

```
133         CASE reg_addr(2 DOWNTO 0) IS
134           WHEN "000" =>
135             st_dout := "1000000000000" & ps_reg & tid & FESTW;
136           WHEN "001" =>
137             st_dout := "1000000000000" & bl_reg & tid & FESTW;
138           WHEN "010" =>
139             st_dout := "1000000000000" & dt_reg & tid & FESTW;
140           WHEN "011" =>
141             st_dout := "1000000000000" & fc_reg & tid & FESTW;
142           WHEN "100" =>
143             st_dout := "1000000000000" & te_reg & tid & FESTW;
144           WHEN "101" =>
145             st_dout := "1000000000000" & xx_reg & tid & FESTW;
146           WHEN OTHERS =>
147             st_dout := "1000000000000" & "00000000" & tid & FESTW;
148         END CASE;
149
150         IF (reg_read = '1') AND (reg_read_req = '0') THEN
151           reg_read_req := '1';
152         ELSIF (output_present = OS_TXSTATUS) THEN
153           reg_read_req := '0';
154         END IF;
155
156         CASE output_present IS
157           WHEN OS_IDLE =>
158             IF (feebus_present = FB_FLOAT) THEN
159               output_next := OS_WAITOUT;
160             ELSE
161               output_next := OS_IDLE;
162             END IF;
163           WHEN OS_WAITOUT =>
164             IF (feebus_present = FB_OUTPUT) THEN
165               IF (reg_read_req = '1') THEN
166                 output_next := OS_TXSTATUS;
167               ELSIF (block_read = '1') THEN
168                 output_next := OS_TXDATA;
169               ELSIF (event_read = '1') THEN
170                 output_next := OS_TXINGAP;
171               ELSE
172                 output_next := OS_WAITIN;
173               END IF;
174             ELSIF (feebus_present = FB_INPUT) THEN
175               output_next := OS_IDLE;
176             ELSE
177               output_next := OS_WAITOUT;
178             END IF;
179           WHEN OS_TXSTATUS =>
180             output_next := OS_WAITIN;
181           WHEN OS_TXDATA =>
182             IF (feebus_present = FB_FLOAT OR feebus_present = FB_RESET) THEN
183               output_next := OS_WAITIN;
184             ELSIF (block_read = '1') AND (b_block_end) THEN
185               output_next := OS_WAITIN;
186             ELSIF (event_read = '1') AND (b_block_end) THEN
187               output_next := OS_TXINGAP;
188             ELSE
189               output_next := OS_TXDATA;
190             END IF;
191           WHEN OS_TXINGAP =>
192             IF (feebus_present = FB_FLOAT OR feebus_present = FB_RESET) THEN
193               output_next := OS_WAITIN;
194             ELSIF (trigger = '1') THEN
195               output_next := OS_TXDATA;
196             ELSE
197               output_next := OS_TXINGAP;
198             END IF;
```

```
199          WHEN OS_WAITIN =>
200            IF (feebus_present = FB_INPUT) THEN
201              output_next := OS_IDLE;
202            ELSE
203              output_next := OS_WAITIN;
204            END IF;
205        END CASE;
206        output_present := output_next;
207        pg_enable      <= bool2sl(output_next = OS_TXDATA) AND event_read;
208        im_enable      <= bool2sl(output_next = OS_TXDATA) AND block_read;
209        gap_active     <= bool2sl(output_next = OS_TXINGAP);
210
211        CASE feebus_present IS
212          WHEN FB_INPUT =>
213            IF (fiBEN_N = '1') THEN
214              feebus_next := FB_FLOAT;
215            ELSE
216              feebus_next := FB_INPUT;
217            END IF;
218          WHEN FB_FLOAT =>
219            IF (fiBEN_N = '0') THEN
220              IF (fiDIR = '0') THEN
221                feebus_next := FB_INPUT;
222              ELSE
223                feebus_next := FB_OUTPUT;
224              END IF;
225            ELSE
226              feebus_next := FB_FLOAT;
227            END IF;
228          WHEN FB_OUTPUT =>
229            IF (fiBEN_N = '1') THEN
230              feebus_next := FB_FLOAT;
231            ELSIF (fiDIR = '0') THEN      -- 22.03.2002
232              feebus_next := FB_RESET;    -- 22.03.2002
233            ELSE
234              feebus_next := FB_OUTPUT;
235            END IF;
236          WHEN FB_RESET =>                -- 22.03.2002
237            feebus_next := FB_INPUT;      -- 22.03.2002
238        END CASE;
239        feebus_present := feebus_next;
240
241      END IF;
242
243    END PROCESS;
244
245  END SYN;
246
```