```vhdl
--345678901234567890123456789012345678901234567890123456789012345678901234567890
-- $Id: receiver.vhd,v 1.2 2004/12/08 22:52:28 tofp Exp $
--*****************************************************************************
--*  RECEIVER.VHD
--*
--*
--*  REVISION HISTORY:
--*    11-Oct-2001 CS  Original coding
--*    22-Mar-2002 CS  Event ID reset
--*                    FEIC reset
--*
--*****************************************************************************


LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY receiver IS
  PORT (
    clock       : IN  std_logic;
    arstn       : IN  std_logic;
    fc_reg      : IN  std_logic_vector ( 7 DOWNTO 0);
    block_read  : OUT std_logic;
    block_write : OUT std_logic;
    event_read  : OUT std_logic;
    reset_evid  : OUT std_logic;
    im_din      : OUT std_logic_vector (31 DOWNTO 0);
    im_dinval   : OUT std_logic;
    reg_data    : OUT std_logic_vector ( 7 DOWNTO 0);
    reg_addr    : OUT std_logic_vector ( 5 DOWNTO 0);
    reg_load    : OUT std_logic;
    reg_read    : OUT std_logic;
    reg_lock    : OUT std_logic;
    tid         : OUT std_logic_vector ( 3 DOWNTO 0);
    fiD         : IN  std_logic_vector (31 DOWNTO 0);
    fiTEN_N     : IN  std_logic;
    fiCTRL_N    : IN  std_logic;
    fiDIR       : IN  std_logic;
    fiBEN_N     : IN  std_logic;
    foBSY_N     : OUT std_logic
    );
END receiver;

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

ARCHITECTURE SYN OF receiver IS

  CONSTANT CMD_FECTRL : std_logic_vector := "11000100";
  CONSTANT CMD_FESTRD : std_logic_vector := "01000100";
  CONSTANT CMD_STBWR  : std_logic_vector := "11010100";
  CONSTANT CMD_STBRD  : std_logic_vector := "01010100";
  CONSTANT CMD_RDYRX  : std_logic_vector := "00010100";

  CONSTANT EACHWORD : std_logic_vector := "01";
  CONSTANT EACH128W : std_logic_vector := "10";
  CONSTANT EACH16KW : std_logic_vector := "11";

  TYPE input_state IS (
    IS_IDLE,
    IS_EVAL,
    IS_FECTRL,
    IS_FESTRD,
    IS_STBWR,
```

```
67        IS_STBRD,
68        IS_EVDATA);
69
70      TYPE flow_state IS (
71        FS_IDLE,
72        FS_COUNT,
73        FS_BUSY,
74        FS_WAIT);
75
76      TYPE feebus_state IS (
77        FB_INPUT,
78        FB_FLOAT,
79        FB_OUTPUT,
80        FB_RESET
81        );
82
83    BEGIN
84
85      main : PROCESS (clock, arstn)
86
87        VARIABLE flow_present   : flow_state;
88        VARIABLE flow_next      : flow_state;
89        VARIABLE input_present  : input_state;
90        VARIABLE input_next     : input_state;
91        VARIABLE feebus_present : feebus_state;
92        VARIABLE feebus_next    : feebus_state;
93        VARIABLE f_count        : std_logic_vector (14 DOWNTO 0);
94        VARIABLE f_count_init   : std_logic_vector (14 DOWNTO 0);
95        VARIABLE f_count_of     : std_logic;
96        VARIABLE b_rxany        : boolean;
97        VARIABLE b_rxdat        : boolean;
98        VARIABLE b_rxcmd        : boolean;
99        VARIABLE command_code   : std_logic_vector (7 DOWNTO 0);
100       VARIABLE command_tid    : std_logic_vector (3 DOWNTO 0);
101       VARIABLE command_param  : std_logic_vector (18 DOWNTO 0);
102       VARIABLE download       : std_logic;
103
104     BEGIN
105
106       IF (arstn = '0') THEN
107         foBSY_N        <= '1';
108         block_read     <= '0';
109         block_write    <= '0';
110         event_read     <= '0';
111         reset_evid     <= '0';
112         im_din         <= (OTHERS => '0');
113         im_dinval      <= '0';
114         reg_data       <= (OTHERS => '0');
115         reg_addr       <= (OTHERS => '0');
116         reg_load       <= '0';
117         reg_read       <= '0';
118         reg_lock       <= '0';
119         tid            <= (OTHERS => '0');
120         flow_present   := FS_IDLE;
121         flow_next      := FS_IDLE;
122         input_present  := IS_IDLE;
123         input_next     := IS_IDLE;
124         feebus_present := FB_INPUT;
125         feebus_next    := FB_INPUT;
126         f_count        := (OTHERS => '0');
127         f_count_init   := (OTHERS => '0');
128         b_rxany        := false;
129         b_rxdat        := false;
130         b_rxcmd        := false;
131         command_code   := (OTHERS => '0');
132         command_tid    := (OTHERS => '0');
```

```
133         command_param  := (OTHERS => '0');
134         download       := '0';
135       ELSIF (clock'event AND clock = '1') THEN
136
137         CASE fc_reg(1 DOWNTO 0) IS
138           WHEN EACHWORD =>
139             f_count_init := "111111111111111";
140           WHEN EACH128W =>
141             f_count_init := "000000001111110";
142           WHEN EACH16KW =>
143             f_count_init := "011111111111110";
144           WHEN OTHERS =>
145             f_count_init := "011111111111110";
146         END CASE;
147
148         b_rxany    := (fiDIR = '0') AND (fiTEN_N = '0');
149         f_count_of := f_count(14);
150         IF (b_rxany) OR (flow_present = FS_IDLE) THEN
151           f_count := f_count_init;
152         ELSE
153           f_count := dec(f_count);
154         END IF;
155
156         CASE flow_present IS
157           WHEN FS_IDLE =>
158             IF (download = '1') THEN
159               flow_next := FS_COUNT;
160             ELSE
161               flow_next := FS_IDLE;
162             END IF;
163           WHEN FS_COUNT =>
164             IF (f_count_of = '1') AND (b_rxany) THEN
165               flow_next := FS_BUSY;
166             ELSE
167               flow_next := FS_COUNT;
168             END IF;
169           WHEN FS_BUSY =>
170             flow_next := FS_WAIT;
171           WHEN FS_WAIT =>
172             IF (download = '0') THEN
173               flow_next := FS_IDLE;
174             ELSIF (NOT b_rxany) THEN
175               flow_next := FS_COUNT;
176             ELSE
177               flow_next := FS_WAIT;
178             END IF;
179         END CASE;
180         flow_present := flow_next;
181         IF (flow_next = FS_BUSY) THEN
182           foBSY_N <= '0';
183         ELSE
184           foBSY_N <= '1';
185         END IF;
186
187         b_rxdat := b_rxany AND (fiCTRL_N = '1');
188         IF (input_present = IS_STBWR) THEN
189           im_dinval <= bool2sl(b_rxdat);
190           im_din    <= fiD;
191         ELSE
192           im_dinval <= '0';
193           im_din    <= (OTHERS => '0');
194         END IF;
195
196         IF (input_present = IS_FECTRL) THEN
197           reg_load <= '1';
198         ELSE
```

```
199           reg_load <= '0';
200         END IF;
201         IF (input_present = IS_FESTRD) THEN
202           reg_read <= '1';
203         ELSE
204           reg_read <= '0';
205         END IF;
206         reg_data <= command_param( 7 DOWNTO 0);
207         reg_addr <= command_param(13 DOWNTO 8);
208         reg_lock <= NOT bool2sl(input_present = IS_IDLE);
209
210         IF (input_present = IS_STBRD) THEN
211           block_read <= '1';
212         ELSE
213           block_read <= '0';
214         END IF;
215
216         IF (input_present = IS_STBWR) THEN
217           block_write <= '1';
218         ELSE
219           block_write <= '0';
220         END IF;
221
222         IF (input_present = IS_EVDATA) THEN
223           event_read <= '1';
224         ELSE
225           event_read <= '0';
226         END IF;
227
228         IF (input_present = IS_EVAL AND command_code = CMD_RDYRX) THEN
229           reset_evid <= '1';
230         ELSE
231           reset_evid <= '0';
232         END IF;
233
234         CASE input_present IS
235           WHEN IS_IDLE =>
236             download := '0';
237             IF (b_rxcmd) THEN
238               input_next := IS_EVAL;
239             ELSE
240               input_next := IS_IDLE;
241             END IF;
242           WHEN IS_EVAL =>
243             download := '0';
244             IF (command_code = CMD_FECTRL) THEN
245               input_next := IS_FECTRL;
246             ELSIF (command_code = CMD_FESTRD) THEN
247               input_next := IS_FESTRD;
248             ELSIF (command_code = CMD_STBWR) THEN
249               input_next := IS_STBWR;
250             ELSIF (command_code = CMD_STBRD) THEN
251               input_next := IS_STBRD;
252             ELSIF (command_code = CMD_RDYRX) THEN
253               input_next := IS_EVDATA;
254             ELSE
255               input_next := IS_IDLE;
256             END IF;
257           WHEN IS_FECTRL =>
258             download   := '0';
259             input_next := IS_IDLE;
260           WHEN IS_FESTRD =>
261             download   := '0';
262             input_next := IS_IDLE;
263           WHEN IS_STBWR =>
264             download := '1';
```

```
265             IF (b_rxcmd) THEN
266               input_next := IS_IDLE;
267             ELSE
268               input_next := IS_STBWR;
269             END IF;
270           WHEN IS_STBRD =>
271             download := '0';
272             IF (b_rxcmd OR feebus_present = FB_RESET) THEN
273               input_next := IS_IDLE;
274             ELSE
275               input_next := IS_STBRD;
276             END IF;
277           WHEN IS_EVDATA =>
278             download := '0';
279             IF (b_rxcmd OR feebus_present = FB_RESET) THEN
280               input_next := IS_IDLE;
281             ELSE
282               input_next := IS_EVDATA;
283             END IF;
284         END CASE;
285         input_present := input_next;
286
287         tid <= command_tid;
288
289         b_rxcmd := b_rxany AND (fiCTRL_N = '0');
290         IF (b_rxcmd) THEN
291           command_code  := fiD( 7 DOWNTO 0);
292           command_tid   := fiD(11 DOWNTO 8);
293           command_param := fiD(30 DOWNTO 12);
294         END IF;
295
296         CASE feebus_present IS
297           WHEN FB_INPUT =>
298             IF (fiBEN_N = '1') THEN
299               feebus_next := FB_FLOAT;
300             ELSE
301               feebus_next := FB_INPUT;
302             END IF;
303           WHEN FB_FLOAT =>
304             IF (fiBEN_N = '0') THEN
305               IF (fiDIR = '0') THEN
306                 feebus_next := FB_INPUT;
307               ELSE
308                 feebus_next := FB_OUTPUT;
309               END IF;
310             ELSE
311               feebus_next := FB_FLOAT;
312             END IF;
313           WHEN FB_OUTPUT =>
314             IF (fiBEN_N = '1') THEN
315               feebus_next := FB_FLOAT;
316             ELSIF (fiDIR = '0') THEN      -- 22.03.2002
317               feebus_next := FB_RESET;    -- 22.03.2002
318             ELSE
319               feebus_next := FB_OUTPUT;
320             END IF;
321           WHEN FB_RESET =>                -- 22.03.2002
322             feebus_next := FB_INPUT;      -- 22.03.2002
323         END CASE;
324         feebus_present := feebus_next;
325
326       END IF;
327     END PROCESS;
328
329   END SYN;
330
```