```
////////////////////////////////////////////////////////////////////
////                                                            ////
////  File name "fault_sm.v"                                    ////
////                                                            ////
////  This file is part of the "10GE MAC" project               ////
////  http://www.opencores.org/cores/xge_mac/                   ////
////                                                            ////
////  Author(s):                                                ////
////      - A. Tanguay (antanguay@opencores.org)                ////
////                                                            ////
////////////////////////////////////////////////////////////////////
////                                                            ////
//// Copyright (C) 2008 AUTHORS. All rights reserved.           ////
////                                                            ////
//// This source file may be used and distributed without       ////
//// restriction provided that this copyright statement is not  ////
//// removed from the file and that any derivative work contains ////
//// the original copyright notice and the associated disclaimer. ////
////                                                            ////
//// This source file is free software; you can redistribute it ////
//// and/or modify it under the terms of the GNU Lesser General ////
//// Public License as published by the Free Software Foundation; ////
//// either version 2.1 of the License, or (at your option) any ////
//// later version.                                             ////
////                                                            ////
//// This source is distributed in the hope that it will be     ////
//// useful, but WITHOUT ANY WARRANTY; without even the implied ////
//// warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR    ////
//// PURPOSE.  See the GNU Lesser General Public License for more ////
//// details.                                                   ////
////                                                            ////
//// You should have received a copy of the GNU Lesser General  ////
//// Public License along with this source; if not, download it ////
//// from http://www.opencores.org/lgpl.shtml                   ////
////                                                            ////
////////////////////////////////////////////////////////////////////


`include "defines.v"

module fault_sm(/*AUTOARG*/
  // Outputs
  status_local_fault_crx, status_remote_fault_crx,
  // Inputs
  clk_xgmii_rx, reset_xgmii_rx_n, local_fault_msg_det,
  remote_fault_msg_det
  );

input         clk_xgmii_rx;
input         reset_xgmii_rx_n;

input  [1:0]  local_fault_msg_det;
input  [1:0]  remote_fault_msg_det;

output        status_local_fault_crx;
output        status_remote_fault_crx;

/*AUTOREG*/
// Beginning of automatic regs (for this module's undeclared outputs)
reg                     status_local_fault_crx;
```

```
reg                     status_remote_fault_crx;
// End of automatics

reg    [1:0]  curr_state;

reg    [7:0]  col_cnt;
reg    [1:0]  fault_sequence;
reg    [1:0]  last_seq_type;
reg    [1:0]  link_fault;
reg    [2:0]  seq_cnt;
reg    [1:0]  seq_type;

reg    [1:0]  seq_add;

/*AUTOWIRE*/


parameter [1:0]
              SM_INIT       = 2'd0,
              SM_COUNT      = 2'd1,
              SM_FAULT      = 2'd2,
              SM_NEW_FAULT  = 2'd3;


always @(/*AS*/local_fault_msg_det or remote_fault_msg_det) begin

    //---
    // Fault indication. Indicate remote or local fault

    fault_sequence = local_fault_msg_det | remote_fault_msg_det;


    //---
    // Sequence type, local, remote, or ok

    if (|local_fault_msg_det) begin
        seq_type = `LINK_FAULT_LOCAL;
    end
    else if (|remote_fault_msg_det) begin
        seq_type = `LINK_FAULT_REMOTE;
    end
    else begin
        seq_type = `LINK_FAULT_OK;
    end


    //---
    // Adder for number of faults, if detected in lower 4 lanes and
    // upper 4 lanes, add 2. That's because we process 64-bit at a time
    // instead of typically 32-bit xgmii.

    if (|remote_fault_msg_det) begin
        seq_add = remote_fault_msg_det[1] + remote_fault_msg_det[0];
    end
    else begin
        seq_add = local_fault_msg_det[1] + local_fault_msg_det[0];
    end

end

```

```
always @(posedge clk_xgmii_rx or negedge reset_xgmii_rx_n) begin

    if (reset_xgmii_rx_n == 1'b0) begin


        status_local_fault_crx <= 1'b0;
        status_remote_fault_crx <= 1'b0;

    end
    else begin

        //---
        // Status signal to generate local/remote fault interrupts

        status_local_fault_crx <= curr_state == SM_FAULT &&
                                  link_fault == `LINK_FAULT_LOCAL;

        status_remote_fault_crx <= curr_state == SM_FAULT &&
                                   link_fault == `LINK_FAULT_REMOTE;

    end

end

always @(posedge clk_xgmii_rx or negedge reset_xgmii_rx_n) begin

    if (reset_xgmii_rx_n == 1'b0) begin

        curr_state <= SM_INIT;

        col_cnt <= 8'b0;
        last_seq_type <= `LINK_FAULT_OK;
        link_fault <= `LINK_FAULT_OK;
        seq_cnt <= 3'b0;

    end
    else begin

        case (curr_state)

          SM_INIT:
            begin

                last_seq_type <= seq_type;

                if (|fault_sequence) begin

                    // If a fault is detected, capture the type of
                    // fault and start column counter. We need 4 fault
                    // messages in 128 columns to accept the fault.

                    if (fault_sequence[0]) begin
                        col_cnt <= 8'd2;
                    end
                    else begin
                        col_cnt <= 8'd1;
                    end
                    seq_cnt <= {1'b0, seq_add};
                    curr_state <= SM_COUNT;

```

```
181                 end
182                 else begin
183
184                     // If no faults, stay in INIT and clear counters
185
186                     col_cnt <= 8'b0;
187                     seq_cnt <= 3'b0;
188
189                 end
190             end
191
192           SM_COUNT:
193             begin
194
195                 col_cnt <= col_cnt + 8'd2;
196                 seq_cnt <= seq_cnt + {1'b0, seq_add};
197
198                 if (!fault_sequence[0] && col_cnt >= 8'd127) begin
199
200                     // No new fault in lower lanes and almost
201                     // reached the 128 columns count, abort fault.
202
203                     curr_state <= SM_INIT;
204
205                 end
206                 else if (col_cnt > 8'd127) begin
207
208                     // Reached the 128 columns count, abort fault.
209
210                     curr_state <= SM_INIT;
211
212                 end
213                 else if (|fault_sequence) begin
214
215                     // If fault type has changed, move to NEW_FAULT.
216                     // If not, after detecting 4 fault messages move to
217                     // FAULT state.
218
219                     if (seq_type != last_seq_type) begin
220                         curr_state <= SM_NEW_FAULT;
221                     end
222                     else begin
223                         if ((seq_cnt + {1'b0, seq_add}) > 3'd3) begin
224                             col_cnt <= 8'b0;
225                             link_fault <= seq_type;
226                             curr_state <= SM_FAULT;
227                         end
228                     end
229
230                 end
231             end
232
233           SM_FAULT:
234             begin
235
236                 col_cnt <= col_cnt + 8'd2;
237
238                 if (!fault_sequence[0] && col_cnt >= 8'd127) begin
239
240                     // No new fault in lower lanes and almost
```

```
                    // reached the 128 columns count, abort fault.

                    curr_state <= SM_INIT;

                end
                else if (col_cnt > 8'd127) begin

                    // Reached the 128 columns count, abort fault.

                    curr_state <= SM_INIT;

                end
                else if (|fault_sequence) begin

                    // Clear the column count each time we see a fault,
                    // if fault changes, go no next state.

                    col_cnt <= 8'd0;

                    if (seq_type != last_seq_type) begin
                        curr_state <= SM_NEW_FAULT;
                    end
                end

            end

          SM_NEW_FAULT:
            begin

                // Capture new fault type. Start counters.

                col_cnt <= 8'b0;
                last_seq_type <= seq_type;

                seq_cnt <= {1'b0, seq_add};
                curr_state <= SM_COUNT;

            end

        endcase

    end

end

endmodule
```