```
////////////////////////////////////////////////////////////////////
////                                                            ////
////  File name "tb_xge_mac.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 "timescale.v"
`include "defines.v"

//`define GXB
//`define XIL

module tb;


/*AUTOREG*/

reg [7:0]     tx_buffer[0:10000];
integer       tx_length;

reg           clk_156m25;
reg           clk_312m50;
reg           clk_xgmii_rx;
reg           clk_xgmii_tx;

reg           reset_156m25_n;
reg           reset_xgmii_rx_n;
reg           reset_xgmii_tx_n;
```

```
61
62 reg           pkt_rx_ren;
63
64 reg  [63:0]   pkt_tx_data;
65 reg           pkt_tx_val;
66 reg           pkt_tx_sop;
67 reg           pkt_tx_eop;
68 reg  [2:0]    pkt_tx_mod;
69
70 integer       tx_count;
71 integer       rx_count;
72
73 /*AUTOWIRE*/
74 // Beginning of automatic wires (for undeclared instantiated-module outputs)
75 wire                  pkt_rx_avail;           // From dut of xge_mac.v
76 wire [63:0]           pkt_rx_data;            // From dut of xge_mac.v
77 wire                  pkt_rx_eop;             // From dut of xge_mac.v
78 wire                  pkt_rx_err;             // From dut of xge_mac.v
79 wire [2:0]            pkt_rx_mod;             // From dut of xge_mac.v
80 wire                  pkt_rx_sop;             // From dut of xge_mac.v
81 wire                  pkt_rx_val;             // From dut of xge_mac.v
82 wire                  pkt_tx_full;            // From dut of xge_mac.v
83 wire                  wb_ack_o;               // From dut of xge_mac.v
84 wire [31:0]           wb_dat_o;               // From dut of xge_mac.v
85 wire                  wb_int_o;               // From dut of xge_mac.v
86 wire [7:0]            xgmii_txc;              // From dut of xge_mac.v
87 wire [63:0]           xgmii_txd;              // From dut of xge_mac.v
88 // End of automatics
89
90 wire  [7:0]   wb_adr_i;
91 wire  [31:0]  wb_dat_i;
92
93 wire [7:0]            xgmii_rxc;
94 wire [63:0]           xgmii_rxd;
95
96 wire [3:0]            tx_dataout;
97
98 wire                  xaui_tx_l0_n;
99 wire                  xaui_tx_l0_p;
100 wire                 xaui_tx_l1_n;
101 wire                 xaui_tx_l1_p;
102 wire                 xaui_tx_l2_n;
103 wire                 xaui_tx_l2_p;
104 wire                 xaui_tx_l3_n;
105 wire                 xaui_tx_l3_p;
106
107 xge_mac dut(/*AUTOINST*/
108             // Outputs
109             .pkt_rx_avail              (pkt_rx_avail),
110             .pkt_rx_data               (pkt_rx_data[63:0]),
111             .pkt_rx_eop                (pkt_rx_eop),
112             .pkt_rx_err                (pkt_rx_err),
113             .pkt_rx_mod                (pkt_rx_mod[2:0]),
114             .pkt_rx_sop                (pkt_rx_sop),
115             .pkt_rx_val                (pkt_rx_val),
116             .pkt_tx_full               (pkt_tx_full),
117             .wb_ack_o                  (wb_ack_o),
118             .wb_dat_o                  (wb_dat_o[31:0]),
119             .wb_int_o                  (wb_int_o),
120             .xgmii_txc                 (xgmii_txc[7:0]),
```

```
121             .xgmii_txd                  (xgmii_txd[63:0]),
122             // Inputs
123             .clk_156m25                 (clk_156m25),
124             .clk_xgmii_rx               (clk_xgmii_rx),
125             .clk_xgmii_tx               (clk_xgmii_tx),
126             .pkt_rx_ren                 (pkt_rx_ren),
127             .pkt_tx_data                (pkt_tx_data[63:0]),
128             .pkt_tx_eop                 (pkt_tx_eop),
129             .pkt_tx_mod                 (pkt_tx_mod[2:0]),
130             .pkt_tx_sop                 (pkt_tx_sop),
131             .pkt_tx_val                 (pkt_tx_val),
132             .reset_156m25_n             (reset_156m25_n),
133             .reset_xgmii_rx_n           (reset_xgmii_rx_n),
134             .reset_xgmii_tx_n           (reset_xgmii_tx_n),
135             .wb_adr_i                   (wb_adr_i[7:0]),
136             .wb_clk_i                   (wb_clk_i),
137             .wb_cyc_i                   (wb_cyc_i),
138             .wb_dat_i                   (wb_dat_i[31:0]),
139             .wb_rst_i                   (wb_rst_i),
140             .wb_stb_i                   (wb_stb_i),
141             .wb_we_i                    (wb_we_i),
142             .xgmii_rxc                  (xgmii_rxc[7:0]),
143             .xgmii_rxd                  (xgmii_rxd[63:0]));
144
145 `ifdef GXB
146 // Example of transceiver instance
147 gxb gxb(// Outputs
148         .rx_ctrldetect              ({xgmii_rxc[7],
149                                       xgmii_rxc[5],
150                                       xgmii_rxc[3],
151                                       xgmii_rxc[1],
152                                       xgmii_rxc[6],
153                                       xgmii_rxc[4],
154                                       xgmii_rxc[2],
155                                       xgmii_rxc[0]}),
156         .rx_dataout                 ({xgmii_rxd[63:56],
157                                       xgmii_rxd[47:40],
158                                       xgmii_rxd[31:24],
159                                       xgmii_rxd[15:8],
160                                       xgmii_rxd[55:48],
161                                       xgmii_rxd[39:32],
162                                       xgmii_rxd[23:16],
163                                       xgmii_rxd[7:0]}),
164         .tx_dataout                 (tx_dataout[3:0]),
165         // Inputs
166         .pll_inclk                  (clk_156m25),
167         .rx_analogreset             (~reset_156m25_n),
168         .rx_cruclk                  ({clk_156m25, clk_156m25, clk_156m25,
    clk_156m25}),
169         .rx_datain                  (tx_dataout[3:0]),
170         .rx_digitalreset            (~reset_156m25_n),
171         .tx_ctrlenable              ({xgmii_txc[7],
172                                       xgmii_txc[5],
173                                       xgmii_txc[3],
174                                       xgmii_txc[1],
175                                       xgmii_txc[6],
176                                       xgmii_txc[4],
177                                       xgmii_txc[2],
178                                       xgmii_txc[0]}),
179         .tx_datain                  ({xgmii_txd[63:56],
```

```
                                          xgmii_txd[47:40],
                                          xgmii_txd[31:24],
                                          xgmii_txd[15:8],
                                          xgmii_txd[55:48],
                                          xgmii_txd[39:32],
                                          xgmii_txd[23:16],
                                          xgmii_txd[7:0]}),
        .tx_digitalreset                 (~reset_156m25_n));
`endif

`ifdef XIL
// Example of transceiver instance
xaui_block xaui(// Outputs
                .txoutclk                (),
                .xgmii_rxd               (xgmii_rxd[63:0]),
                .xgmii_rxc               (xgmii_rxc[7:0]),
                .xaui_tx_l0_p            (xaui_tx_l0_p),
                .xaui_tx_l0_n            (xaui_tx_l0_n),
                .xaui_tx_l1_p            (xaui_tx_l1_p),
                .xaui_tx_l1_n            (xaui_tx_l1_n),
                .xaui_tx_l2_p            (xaui_tx_l2_p),
                .xaui_tx_l2_n            (xaui_tx_l2_n),
                .xaui_tx_l3_p            (xaui_tx_l3_p),
                .xaui_tx_l3_n            (xaui_tx_l3_n),
                .txlock                  (),
                .align_status            (),
                .sync_status             (),
                .mgt_tx_ready            (),
                .drp_o                   (),
                .drp_rdy                 (),
                .status_vector           (),
                // Inputs
                .dclk                    (clk_156m25),
                .clk156                  (clk_156m25),
                .clk312                  (clk_312m50),
                .refclk                  (clk_156m25),
                .reset                   (~reset_156m25_n),
                .reset156                (~reset_156m25_n),
                .xgmii_txd               (xgmii_txd[63:0]),
                .xgmii_txc               (xgmii_txc[7:0]),
                .xaui_rx_l0_p            (xaui_tx_l0_p),
                .xaui_rx_l0_n            (xaui_tx_l0_n),
                .xaui_rx_l1_p            (xaui_tx_l1_p),
                .xaui_rx_l1_n            (xaui_tx_l1_n),
                .xaui_rx_l2_p            (xaui_tx_l2_p),
                .xaui_rx_l2_n            (xaui_tx_l2_n),
                .xaui_rx_l3_p            (xaui_tx_l3_p),
                .xaui_rx_l3_n            (xaui_tx_l3_n),
                .signal_detect           (4'b1111),
                .drp_addr                (7'b0),
                .drp_en                  (2'b0),
                .drp_i                   (16'b0),
                .drp_we                  (2'b0),
                .configuration_vector    (7'b0));

glbl glbl();
`endif

//---
// Unused for this testbench
```

```
240
241 assign wb_adr_i = 8'b0;
242 assign wb_clk_i = 1'b0;
243 assign wb_cyc_i = 1'b0;
244 assign wb_dat_i = 32'b0;
245 assign wb_rst_i = 1'b1;
246 assign wb_stb_i = 1'b0;
247 assign wb_we_i = 1'b0;
248
249
250 initial begin
251     tx_count = 0;
252     rx_count = 0;
253 end
254
255 //---
256 // XGMII Loopback
257 // This test is done with loopback on XGMII or using one of the tranceiver examples
258
259 `ifndef GXB
260   `ifndef XIL
261     assign xgmii_rxc = xgmii_txc;
262     assign xgmii_rxd = xgmii_txd;
263   `endif
264 `endif
265
266 //---
267 // Clock generation
268
269 initial begin
270     clk_156m25 = 1'b0;
271     clk_xgmii_rx = 1'b0;
272     clk_xgmii_tx = 1'b0;
273     forever begin
274         WaitPS(3200);
275         clk_156m25 = ~clk_156m25;
276         clk_xgmii_rx = ~clk_xgmii_rx;
277         clk_xgmii_tx = ~clk_xgmii_tx;
278     end
279 end
280
281 initial begin
282     clk_312m50 = 1'b0;
283     forever begin
284         WaitPS(1600);
285         clk_312m50 = ~clk_312m50;
286     end
287 end
288
289 //---
290 // Reset Generation
291
292 initial begin
293     reset_156m25_n = 1'b0;
294     reset_xgmii_rx_n = 1'b0;
295     reset_xgmii_tx_n = 1'b0;
296     WaitNS(20);
297     reset_156m25_n = 1'b1;
298     reset_xgmii_rx_n = 1'b1;
299     reset_xgmii_tx_n = 1'b1;
```

```
end


//---
// Init signals

initial begin

    for (tx_length = 0; tx_length <= 1000; tx_length = tx_length + 1) begin
        tx_buffer[tx_length] = 0;
    end

    pkt_rx_ren = 1'b0;

    pkt_tx_data = 64'b0;
    pkt_tx_val = 1'b0;
    pkt_tx_sop = 1'b0;
    pkt_tx_eop = 1'b0;
    pkt_tx_mod = 3'b0;

end

task WaitNS;
  input [31:0] delay;
    begin
        #(1000*delay);
    end
endtask

task WaitPS;
  input [31:0] delay;
    begin
        #(delay);
    end
endtask


//---
// Task to send a single packet

task TxPacket;
  integer        i;
    begin

        $display("Transmit packet with length: %d", tx_length);

        @(posedge clk_156m25);
        WaitNS(1);
        pkt_tx_val = 1'b1;

        for (i = 0; i < tx_length; i = i + 8) begin

            pkt_tx_sop = 1'b0;
            pkt_tx_eop = 1'b0;
            pkt_tx_mod = 2'b0;

            if (i == 0) pkt_tx_sop = 1'b1;

            if (i + 8 >= tx_length) begin
                pkt_tx_eop = 1'b1;
```

```
                pkt_tx_mod = tx_length % 8;
            end

            pkt_tx_data[`LANE7] = tx_buffer[i];
            pkt_tx_data[`LANE6] = tx_buffer[i+1];
            pkt_tx_data[`LANE5] = tx_buffer[i+2];
            pkt_tx_data[`LANE4] = tx_buffer[i+3];
            pkt_tx_data[`LANE3] = tx_buffer[i+4];
            pkt_tx_data[`LANE2] = tx_buffer[i+5];
            pkt_tx_data[`LANE1] = tx_buffer[i+6];
            pkt_tx_data[`LANE0] = tx_buffer[i+7];

            @(posedge clk_156m25);
            WaitNS(1);

        end

        pkt_tx_val = 1'b0;
        pkt_tx_eop = 1'b0;
        pkt_tx_mod = 3'b0;

        tx_count = tx_count + 1;

    end

endtask


//---
// Task to read a single packet from command file and transmit

task CmdTxPacket;
  input [31:0] file;
  integer count;
  integer data;
  integer i;
    begin

        count = $fscanf(file, "%2d", tx_length);

        if (count == 1) begin

            for (i = 0; i < tx_length; i = i + 1) begin

                count = $fscanf(file, "%2X", data);
                if (count) begin
                    tx_buffer[i] = data;
                end

            end

            TxPacket();

        end
    end

endtask


//---
```

```
420 // Task to read commands from file and stop when complete
421
422 task ProcessCmdFile;
423   integer    file_cmd;
424   integer  count;
425   reg [8*8-1:0] str;
426     begin
427
428         file_cmd = $fopen("../testbench/verilog/packets_tx.txt", "r");
429         if (!file_cmd) $stop;
430
431         while (!$feof(file_cmd)) begin
432
433             count = $fscanf(file_cmd, "%s", str);
434             if (count != 1) continue;
435
436             $display("CMD %s", str);
437
438             case (str)
439
440               "SEND_PKT":
441                 begin
442                     CmdTxPacket(file_cmd);
443                 end
444
445             endcase
446
447         end
448
449         $fclose(file_cmd);
450
451         WaitNS(50000);
452         //$stop;
453         $finish;
454
455     end
456 endtask
457
458 initial begin
459     WaitNS(5000);
460 `ifdef XIL
461     WaitNS(200000);
462 `endif
463     ProcessCmdFile();
464 end
465
466
467 //---
468 // Task to read a single packet from receive interface and display
469
470 task RxPacket;
471   reg done;
472     begin
473
474         done = 0;
475
476         pkt_rx_ren <= 1'b1;
477         @(posedge clk_156m25);
478
479         while (!done) begin
```

```
480
481             if (pkt_rx_val) begin
482
483                 if (pkt_rx_sop) begin
484                     $display("\n\n-----------------------");
485                     $display("Received Packet");
486                     $display("-----------------------");
487                 end
488
489                 $display("%x", pkt_rx_data);
490
491                 if (pkt_rx_eop) begin
492                     done <= 1;
493                     pkt_rx_ren <= 1'b0;
494                 end
495
496                 if (pkt_rx_eop) begin
497                     $display("-----------------------\n\n");
498                 end
499
500             end
501
502             @(posedge clk_156m25);
503
504         end
505
506         rx_count = rx_count + 1;
507
508     end
509 endtask
510
511 initial begin
512
513     forever begin
514
515         if (pkt_rx_avail) begin
516
517             RxPacket();
518
519             if (rx_count == tx_count) begin
520                 $display("All packets received. Sumulation done!!!\n");
521             end
522
523         end
524
525         @(posedge clk_156m25);
526
527     end
528
529 end
530
531 initial begin
532   $vcdpluson;     // Enable waveform dumping
533 end
534
535 endmodule
536
```