Code RTL:

```
module timer_top
(
        input wire sys_clk,
        input wire sys_rst_n,
        input wire tim_psel,
        input wire tim_pwrite,
        input wire tim_penable,
        input wire [11:0] tim_paddr,
        input wire [31:0] tim_pwdata,
        input wire [3:0] tim_pstrb,
        input wire dbg_mode,
        output wire [31:0] tim_prdata,
        output wire tim_pready,
        output wire tim_pslverr,
        output wire tim_int
);

        wire wr_en_w, rd_en_w, pready_w_w, pslverr_w_w;
        wire [31:0] wdata_w;
        wire [11:0] addr_w;
        wire [3:0] wstrb_w;
        wire [31:0] rdata_w;
        wire [31:0] data0_out_w;

        assign tim_pslverr = pslverr_w_w;

        apb_slave apb
(
        .psel(tim_psel),
        .pwrite(tim_pwrite),
        .penable(tim_penable),
        .pwdata(tim_pwdata),
        .paddr(tim_paddr),
        .pstrb(tim_pstrb),
        .pready_w(pready_w_w),
        .rdata(rdata_w),
        .data0_out(data0_out_w),
        .addr(addr_w),
        .wr_en(wr_en_w),
        .rd_en(rd_en_w),
        .wdata(wdata_w),
        .wstrb(wstrb_w),
        .pready(tim_pready),
        .prdata(tim_prdata),
        .pslverr(pslverr_w_w)
);


        reg_module register
(
        .clk(sys_clk),
        .rst_n(sys_rst_n),
        .wr_en(wr_en_w),
        .rd_en(rd_en_w),
        .addr(addr_w),
```

```
24
25          assign tim_pslverr = pslverr_w_w;
26
27          apb_slave apb
28  (
29          .psel(tim_psel),
30          .pwrite(tim_pwrite),
31          .penable(tim_penable),
32          .pwdata(tim_pwdata),
33          .paddr(tim_paddr),
34          .pstrb(tim_pstrb),
35          .pready_w(pready_w_w),
36          .rdata(rdata_w),
37          .data0_out(data0_out_w),
38          .addr(addr_w),
39          .wr_en(wr_en_w),
40          .rd_en(rd_en_w),
41          .wdata(wdata_w),
42          .wstrb(wstrb_w),
43          .pready(tim_pready),
44          .prdata(tim_prdata),
45          .pslverr(pslverr_w_w)
46  );
47
48
49          reg_module register
50  (
51          .clk(sys_clk),
52          .rst_n(sys_rst_n),
53          .wr_en(wr_en_w),
54          .rd_en(rd_en_w),
55          .addr(addr_w),
56          .wdata(wdata_w),
57          .wstrb(wstrb_w),
58          .debug_mode(dbg_mode),
59          .pready_w(pready_w_w),
60          .rdata(rdata_w),
61          .data0_out(data0_out_w),
62          .pslverr(pslverr_w_w),
63          .timer_int(tim_int)
64  );
65
66  endmodule
```

```
module apb_slave
(
        input wire psel,
        input wire pwrite,
        input wire penable,
        input wire [31:0]pwdata,
        input wire [11:0]paddr,
        input wire [3:0]pstrb,
        input wire pready_w,
        input wire [31:0]rdata,
        input wire [31:0]data0_out,
        output reg [11:0]addr,
        output reg wr_en,
        output reg rd_en,
        output reg [31:0]wdata,
        output reg [3:0]wstrb,
        output reg pready,
        output reg [31:0]prdata,
        output reg pslverr
);
        reg pslverr_w;

always @(*) begin

        wr_en   = psel && penable && pwrite && !pready_w;
        rd_en   = psel && penable && !pwrite && !pready_w;

        addr    = paddr;
        wdata   = pwdata;
        wstrb   = pstrb;
        pready  = pready_w;
        pslverr = pslverr_w;
        prdata  = rdata;
end

always @(*) begin
        pslverr_w =1'b0;
        if(psel && penable && pwrite && (paddr == 12'h000) && pstrb[1] ) begin
                if(data0_out[0]) begin
                        if( (pwdata[1] != data0_out[1]) || (pwdata[11:8] >= 4'd9) || (pwdata[11:8] != data0_out[11:8]) ) begin
                        pslverr_w =1'b1;
                        end
                end else if (pwdata [11:8] >= 4'd9) begin
                        pslverr_w =1'b1;
                end
        end
end

endmodule


~
```

```
module reg_module
(
        input wire clk,
        input wire rst_n,
        input wire wr_en,
        input wire rd_en,
        input wire [11:0] addr,
        input wire [31:0] wdata,
        input wire [3:0] wstrb,
        input wire debug_mode,
        input wire pslverr,
        output reg pready_w,
        output reg [31:0] rdata,
        output wire [31:0] data0_out,
        output reg timer_int
);

        reg [31:0] data0, data1, data2, data3, data4, data5, data6, data7;
        wire [31:0] data0_next, data1_next, data2_next, data3_next, data4_next, data5_next, data7_next;
        reg [7:0] int_cnt;
        reg pslverr_w;
        reg [31:0] data0_d;
        reg count_en;
        reg thcr_halt_reg;

        wire [7:0] int_cnt_pre;
        wire [7:0] int_cnt_0;
        wire [63:0] counter_64bit;
        wire count_en_pre, cnt_rst, thcr_halt_reg_pre;
        wire tri_condition;



        assign data0_out = data0;

//============= NEXT STATE LOGIC FOR REGISTER ====================

//==================== TCR ======================
        assign data0_next[7:0] = (wr_en && (addr == 12'h000) && wstrb[0]) ? {6'b0, wdata[1:0]} : data0[7:0];
        assign data0_next[15:8] = (wr_en && (addr == 12'h000) && wstrb[1]) ? {4'b0, wdata[11:8]} : data0[15:8];
        assign data0_next[23:16] = 8'h0;
        assign data0_next[31:24] = 8'h0;

//==================== TDR0 ======================
        assign data1_next[7:0] = (wr_en && (addr == 12'h004) && wstrb[0]) ? wdata[7:0] : counter_64bit[7:0];
        assign data1_next[15:8] = (wr_en && (addr == 12'h004) && wstrb[1]) ? wdata[15:8] :  counter_64bit[15:8];
        assign data1_next[23:16] =(wr_en && (addr == 12'h004) && wstrb[2]) ? wdata[23:16] :  counter_64bit[23:16];
        assign data1_next[31:24] =(wr_en && (addr == 12'h004) && wstrb[3]) ? wdata[31:24] :  counter_64bit[31:24];

//==================== TDR1 ======================
        assign data2_next[7:0] = (wr_en && (addr == 12'h008) && wstrb[0]) ? wdata[7:0] : counter_64bit[39:32];
        assign data2_next[15:8] = (wr_en && (addr == 12'h008) && wstrb[1]) ? wdata[15:8] :  counter_64bit[47:40];
        assign data2_next[23:16] =(wr_en && (addr == 12'h008) && wstrb[2]) ? wdata[23:16] :  counter_64bit[55:48];
        assign data2_next[31:24] =(wr_en && (addr == 12'h008) && wstrb[3]) ? wdata[31:24] :  counter_64bit[63:56];

```

```
55
56 //=================== TCMP0 ======================
57         assign data3_next[7:0] = (wr_en && (addr == 12'h00C) && wstrb[0]) ? wdata[7:0] : data3[7:0];
58         assign data3_next[15:8] = (wr_en && (addr == 12'h00C) && wstrb[1]) ? wdata[15:8] : data3[15:8];
59         assign data3_next[23:16] = (wr_en && (addr == 12'h00C) && wstrb[2]) ? wdata[23:16] : data3[23:16];
60         assign data3_next[31:24] = (wr_en && (addr == 12'h00C) && wstrb[3]) ? wdata[31:24] : data3[31:24];
61
62
63 //=================== TCMP1 ======================
64         assign data4_next[7:0] = (wr_en && (addr == 12'h010) && wstrb[0]) ? wdata[7:0] : data4[7:0];
65         assign data4_next[15:8] = (wr_en && (addr == 12'h010) && wstrb[1]) ? wdata[15:8] : data4[15:8];
66         assign data4_next[23:16] = (wr_en && (addr == 12'h010) && wstrb[2]) ? wdata[23:16] : data4[23:16];
67         assign data4_next[31:24] = (wr_en && (addr == 12'h010) && wstrb[3]) ? wdata[31:24] : data4[31:24];
68
69
70 //=================== TIER ======================
71         assign data5_next[7:0] = (wr_en && (addr == 12'h014) && wstrb[0]) ? {7'h0, wdata[0]} : data5[7:0];
72         assign data5_next[15:8] = 8'h0;
73         assign data5_next[23:16] = 8'h0;
74         assign data5_next[31:24] = 8'h0;
75
76
77 //=================== THCSR ======================
78         assign data7_next[7:0] = (wr_en && (addr == 12'h01C) && wstrb[0]) ? {6'h0, thcr_halt_reg_pre ,wdata[0]} : {data7[7:2],thcr_halt_reg_pre,data7[0]};
79         assign data7_next[15:8] = 8'h0;
80         assign data7_next[23:16] = 8'h0;
81         assign data7_next[31:24] = 8'h0;
82
83
84
85
86 // ================================================
87 // PREADY CONTROL - DELAY 1 CYCLE
88 // ================================================
89
90         always @(posedge clk or negedge rst_n) begin
91                 if(!rst_n) begin
92                         pready_w <= 1'b0;
93
94                 end else begin
95                         pready_w <= (wr_en || rd_en) ? 1'b1 : 1'b0;
96                 end
97         end
98
99
100 // ================================================
101 // REGISTER UPDATE
102 // ================================================
103
104         always @(posedge clk or negedge rst_n) begin
105                 if(!rst_n) begin
106                         data0 <= {20'h0, 4'b0001, 8'b0};
107                         data1 <= 32'h0000_0000;
```

```
100 // ================================================
101 // REGISTER UPDATE
102 // ================================================
103
104         always @(posedge clk or negedge rst_n) begin
105                 if(!rst_n) begin
106                         data0 <= {20'h0, 4'b0001, 8'b0};
107                         data1 <= 32'h0000_0000;
108                         data2 <= 32'h0000_0000;
109                         data3 <= 32'hFFFF_FFFF;
110                         data4 <= 32'hFFFF_FFFF;
111                         data5 <= 32'h0000_0000;
112                         data6 <= 32'h0000_0000;
113                         data7 <= 32'h0000_0000;
114                         data0_d <= 32'h0000_0000;
115
116                 end else begin
117                         data0_d <= data0;
118
119                         if(!data0[0] && data0_d[0]) begin
120                                 data1 <= 32'h0000_0000;
121                                 data2 <= 32'h0000_0000;
122                         end else begin
123                                 data0 <= pslverr_w ? data0 : data0_next;
124                                 data1 <= data1_next;
125                                 data2 <= data2_next;
126                                 data3 <= data3_next;
127                                 data4 <= data4_next;
128                                 data5 <= data5_next;
129                                 data7 <= data7_next;
130                         end
131                 end
132         end
133
134
135 // ================================================
136 // READ LOGIC
137 // ================================================
138
139         always @(posedge clk or negedge rst_n) begin
140                 if(!rst_n) begin
141                         rdata <= 32'h0000_0000;
142                 end else begin
143                         rdata <= 32'h0000_0000;
144
145                 if (rd_en) begin
146                         case(addr)
147                                 12'h000 : rdata <= data0;
148                                 12'h004 : rdata <= data1;
149                                 12'h008 : rdata <= data2;
150                                 12'h00C : rdata <= data3;
```

```
144
145                 if (rd_en) begin
146                         case(addr)
147                                 12'h000 : rdata <= data0;
148                                 12'h004 : rdata <= data1;
149                                 12'h008 : rdata <= data2;
150                                 12'h00C : rdata <= data3;
151                                 12'h010 : rdata <= data4;
152                                 12'h014 : rdata <= data5;
153                                 12'h018 : rdata <= data6;
154                                 12'h01C : rdata <= data7;
155                                 default : rdata <= 32'h0;
156                         endcase
157                         end
158                 end
159         end
160
161
162
163 // ================================================
164 // TIMER CLOCK DIVIDER LOGIC
165 // ================================================
166
167         assign cnt_rst = !data0[0] || !data0[1] || (int_cnt == ((1 << data0[11:8]) - 1));
168
169         assign count_en_pre = thcr_halt_reg ? 1'b0 : (
170                 (!data0[1] && data0[0]) || (data0[11:8] != 0 && data0[1] && data0[0] && (int_cnt == ((1 << data0[11:8]) - 1)))
171                 || (data0[11:8] == 0 && data0[1] && data0[0]));
172
173         assign int_cnt_0 = cnt_rst ? 8'h0 : int_cnt +1;
174         assign int_cnt_pre = thcr_halt_reg ? int_cnt : int_cnt_0;
175
176         always@(posedge clk or negedge rst_n) begin
177                 if(!rst_n) begin
178                         int_cnt <= 8'h0;
179                         count_en <= 1'b0;
180                 end else begin
181                         int_cnt <= int_cnt_pre;
182                         count_en <= count_en_pre;
183                 end
184         end
185
186         assign counter_64bit = count_en ? {data2, data1} +1 : {data2, data1};
187
188 // ================================================
189 // COMPARE LOGIC && INTERRUPT OUTPUT
190 // ================================================
191
192         always@(*) begin
193                 pslverr_w = pslverr;
194                 timer_int = data5[0] && data6[0];
195         end
196
```

```
177                 if(!rst_n) begin
178                         int_cnt <= 8'h0;
179                         count_en <= 1'b0;
180                 end else begin
181                         int_cnt <= int_cnt_pre;
182                         count_en <= count_en_pre;
183                 end
184         end
185
186         assign counter_64bit = count_en ? {data2, data1} +1 : {data2, data1};
187
188 // ================================================
189 // COMPARE LOGIC && INTERRUPT OUTPUT
190 // ================================================
191
192         always@(*) begin
193                 pslverr_w = pslverr;
194                 timer_int = data5[0] && data6[0];
195         end
196
197         assign tri_condition = (data3 == data1) && (data4 == data2);
198
199
200 // ================================================
201 // INTERRUPT STATUS REGISTER (data6[0])
202 // ================================================
203         always@(posedge clk or rst_n) begin
204         if(!rst_n) begin
205                 data6[0] <= 1'b0;
206         end else begin
207                 if(tri_condition) begin
208                         data6[0] <= 1'b1;
209                 end else if(wr_en && (addr == 12'h018) && wstrb[0] && wdata[0] && data6[0]) begin
210                         data6[0] <= 1'b0;
211                         end
212                 end
213         end
214
215
216
217 // ================================================
218 // HALT CONTROL IN DEBUG MODE
219 // ================================================
220         assign thcr_halt_reg_pre = debug_mode && data7_next[0];
221
222         always@(posedge clk or negedge rst_n) begin
223                 if(!rst_n) begin
224                         thcr_halt_reg <= 1'b0;
225                 end else begin
226                         thcr_halt_reg <= thcr_halt_reg_pre;
227                 end
228         end
229
230 endmodule
231
```

Test bench and Test Case

```
module test_bench;

        reg sys_clk;
        reg sys_rst_n;

        reg tim_psel;
        reg tim_pwrite;
        reg tim_penable;
        reg [11:0] tim_paddr;
        reg [31:0] tim_pwdata;
        reg [3:0] tim_pstrb;
        reg dbg_mode;
        integer err;

        wire [31:0] tim_prdata;
        wire tim_pready;
        wire tim_pslverr;
        wire tim_int;

        timer_top dut (
                .sys_clk(sys_clk),
                .sys_rst_n(sys_rst_n),
                .tim_psel(tim_psel),
                .tim_pwrite(tim_pwrite),
                .tim_penable(tim_penable),
                .tim_paddr(tim_paddr),
                .tim_pwdata(tim_pwdata),
                .tim_pstrb(tim_pstrb),
                .dbg_mode(dbg_mode),
                .tim_prdata(tim_prdata),
                .tim_pready(tim_pready),
                .tim_pslverr(tim_pslverr),
                .tim_int(tim_int)
        );

        `include "run_test.v"

         initial begin
           sys_clk = 0;
           forever #25 sys_clk = ~sys_clk;
         end


        initial begin

                $display("\n");
```

```
45
46              $display("\n");
47              $display("======================================");
48              $display("      Timer Top Testbench      ");
49              $display("======================================");
50
51              initialize();
52              err = 0;
53              #100;
54              run_test();
55              #100;
56              if(err == 0 )
57                      $display("\nTest_result PASSED");
58               else
59                      $display("\nTest_result FAILED");
60              #100;
61              $finish;
62      end
63
64      task initialize;
65              begin
66
67                      sys_rst_n = 0;
68                      #30;
69                      sys_rst_n = 1;
70                      #10;
71
72                      tim_psel = 0;
73                      tim_pwrite = 0;
74                      tim_penable = 0;
75                      tim_paddr = 12'h0;
76                      tim_pwdata = 32'h0;
77                      tim_pstrb = 4'h0;
78                      dbg_mode = 0;
79
80                      #20;
81              end
82      endtask
83
84
85      task apb_write_register;
86
87              input [11:0] address;
88              input [31:0] data;
89              input [3:0] strobe;
90              begin
```

```
 88                input [31:0] data;
 89                input [3:0] strobe;
 90                begin
 91                        @(posedge sys_clk);
 92                        #1;
 93                        tim_psel = 1;
 94                        tim_pwrite = 1;
 95                        tim_penable = 0;
 96                        tim_paddr = address;
 97                        tim_pwdata = data;
 98                        tim_pstrb = strobe;
 99
100                        @(posedge sys_clk);
101                        #1;
102                        tim_penable = 1;
103
104                        wait(tim_pready == 1);
105                        @(posedge sys_clk);
106                        #1;
107                        tim_psel = 0;
108                        tim_pwrite = 0;
109                        tim_penable = 0;
110                        tim_paddr = 12'h0;
111                        tim_pwdata = 32'h0;
112                        tim_pstrb = 4'h0;
113                        #10;
114                end
115        endtask
116
```

```
116
117
118
119         task apb_write_register_div_val;
120
121                 input [11:0] address;
122                 input [31:0] data;
123                 input [3:0] strobe;
124                 input [3:0] i;
125                 begin
126                         @(posedge sys_clk);
127                         #1;
128                         tim_psel = 1;
129                         tim_pwrite = 1;
130                         tim_penable = 0;
131                         tim_paddr = address;
132                         tim_pwdata = {20'h0,i,6'h0,data[1:0]
133                         tim_pstrb = strobe;
134
135                         @(posedge sys_clk);
136                         #1;
137                         tim_penable = 1;
138
139                         wait(tim_pready == 1);
140                         @(posedge sys_clk);
141                         #1;
142                         tim_psel = 0;
143                         tim_pwrite = 0;
144                         tim_penable = 0;
145                         tim_paddr = 12'h0;
146                         tim_pwdata = 32'h0;
147                         tim_pstrb = 4'h0;
148                         #10;
149                 end
150         endtask
151
```

```
152
153       task apb_write_register_err;
154
155               input [11:0] address;
156               input [31:0] data;
157               input [3:0] strobe;
158               input penable;
159               input psel;
160               begin
161                       @(posedge sys_clk);
162                       #1;
163                       tim_psel = psel;
164                       tim_pwrite = 1;
165                       tim_penable = 0;
166                       tim_paddr = address;
167                       tim_pwdata = data;
168                       tim_pstrb = strobe;
169
170                       @(posedge sys_clk);
171                       #1;
172                       tim_penable = penable;
173                       $display("write data : 32'h%h with wrong protocol",data);
174
175
176               //      wait(tim_pready == 1);
177                       repeat(5)@(posedge sys_clk);
178                       #1;
179                       tim_psel = 0;
180                       tim_pwrite = 0;
181                       tim_penable = 0;
182                       tim_paddr = 12'h0;
183                       tim_pwdata = 32'h0;
184                       tim_pstrb = 4'h0;
185                       #10;
186               end
187       endtask
188
```

```
189
190         task apb_read_register;
191                 input [11:0] address;
192                 output [31:0] read_data;
193                 begin
194                         @(posedge sys_clk);
195                         #1;
196                         tim_psel = 1;
197                         tim_pwrite = 0;
198                         tim_penable = 0;
199                         tim_paddr = address;
200
201                         @(posedge sys_clk);
202                         #1;
203                         tim_penable = 1;
204
205                         wait (tim_pready == 1);
206                         #1;
207                         read_data = tim_prdata;
208
209                         @(posedge sys_clk);
210                         #1;
211                         tim_psel = 0;
212                         tim_pwrite = 0;
213                         tim_penable = 0;
214                         tim_paddr = 12'h0;
215                         #10;
216                 end
217         endtask
218
```

```
219         task apb_read_register_err;
220                 input [11:0] address;
221                 output [31:0] read_data;
222                 input penable;
223                 input psel;
224                 begin
225                         @(posedge sys_clk);
226                         #1;
227                         tim_psel = psel;
228                         tim_pwrite = 0;
229                         tim_penable = 0;
230                         tim_paddr = address;
231 
232                         @(posedge sys_clk);
233                         #1;
234                         tim_penable = penable;
235                         $display("read data with wrong protocol");
236 
237 
238                 //      wait (tim_pready == 1);
239                         #1;
240                         read_data = tim_prdata;
241 
242                         repeat(5)@(posedge sys_clk);
243                         #1;
244                         tim_psel = 0;
245                         tim_pwrite = 0;
246                         tim_penable = 0;
247                         tim_paddr = 12'h0;
248                         #10;
249                 end
250         endtask
251 
252 
253         task check_result;
254                 input [31:0] expected;
255                 input [31:0] actual;
256                 begin
257                         if(expected == actual) begin
258                                 $display("pass: expected = 32'h%h, actual = 32'h%h",expected, actual);
259                         end else begin
260                                 $display("error: expected = 32'h%h, actual = 32'h%h",expected, actual);
261                                 err = err +1;
262                         end
263                 end
264         endtask
265 
266         task check_signal;
267                 input expected;
268                 input actual;
269                 begin
270                 if(expected == actual) begin
271                                 $display("pass: expected = %h, actual = %h",expected, actual);
272                         end else begin
273                                 $display("error: expected = %h, actual = %h",expected, actual);
274                                 err = err +1;
275                         end
276                 end
277         endtask
278 
```

```
278
279         task check_pslverr;
280
281                 input [11:0] address;
282                 input [31:0] data;
283                 input [3:0] strobe;
284                 input [3:0] i;
285                 begin
286                         @(posedge sys_clk);
287                         #1;
288                         tim_psel = 1;
289                         tim_pwrite = 1;
290                         tim_penable = 0;
291                         tim_paddr = address;
292                         tim_pwdata = {20'h0,i,6'h0,data[1],data[0]};
293                         tim_pstrb = strobe;
294
295                         @(posedge sys_clk);
296                         #1;
297                         tim_penable = 1;
298                         #1;
299                         if(tim_pslverr == 1 )begin
300                 $display("------------------------------------------------------------");
301                 $display(" PASSED: tim_pslverr is asserted. tim_pslverr = %d",tim_pslverr);
302                 $display("------------------------------------------------------------");
303         end else begin
304                 $display("------------------------------------------------------------");
305                 $display(" FAIL: tim_pslverr does not asserted. tim_pslverr = %d",tim_pslverr );
306                 $display("------------------------------------------------------------");
307                 err = err + 1;
308         end
309
310                         wait(tim_pready == 1);
311                         @(posedge sys_clk);
312                         #1;
313                         tim_psel = 0;
314                         tim_pwrite = 0;
315                         tim_penable = 0;
316                         tim_paddr = 12'h0;
317                         tim_pwdata = 32'h0;
318                         tim_pstrb = 4'h0;
319                         #10;
320                 end
321         endtask
322
```

```
323
324         task check_pready;
325
326                 input [11:0] address;
327                 input [31:0] data;
328                 input [3:0] strobe;
329                 input write;
330                 input signal;
331                 begin
332                         @(posedge sys_clk);
333                         #1;
334                         tim_psel = 1;
335                         tim_pwrite = write;
336                         tim_penable = 0;
337                         tim_paddr = address;
338                         tim_pwdata = data;
339                         tim_pstrb = strobe;
340
341                         @(posedge sys_clk);
342                         #1;
343                         tim_penable = 1;
344
345                         @(posedge sys_clk);
346                         #1;
347                         check_signal(signal ,tim_pready);
348
349
350                         @(posedge sys_clk);
351                         #1;
352                         tim_psel = 0;
353                         tim_pwrite = 0;
354                         tim_penable = 0;
355                         tim_paddr = 12'h0;
356                         tim_pwdata = 32'h0;
357                         tim_pstrb = 4'h0;
358                         #10;
359                 end
360         endtask
361
```

```
363
364         task check_timer_count;
365
366                 input [11:0] address;
367                 input [31:0] data;
368                 input [3:0] strobe;
369                 input [3:0] n;
370
371                 reg [63:0] data_before, data_after;
372
373                 begin
374                         @(posedge sys_clk);
375                         #1;
376                         tim_psel = 1;
377                         tim_pwrite = 1;
378                         tim_penable = 0;
379                         tim_paddr = address;
380                         tim_pwdata = data;
381                         tim_pstrb = strobe;
382
383                         @(posedge sys_clk);
384                         #1;
385                         tim_penable = 1;
386
387                         wait(tim_pready == 1);
388
389                         #1;
390                         repeat(2**n)@(posedge sys_clk);
391                         #1;
392                         data_before = dut.register.counter_64bit;
393
394                         @(posedge sys_clk);
395                         #1;
396                         repeat(2**n)@(posedge sys_clk);
397                         data_after = dut.register.counter_64bit;
398                         #1;
399
400                         if(data_after == data_before +1) begin
401                         $display("Pass, before = 32'h%h, after = 32'h%h", data_before, data_after);
402                         end else begin
403                         $display("Error, before = 32'h%h, after = 32'h%h", data_before, data_after);
404                         err = err +1;
405                         end
406
407                         #1;
408                         data_before = data_after;
```

```
406
407                         #1;
408                         data_before = data_after;
409                         repeat(2**n)@(posedge sys_clk);
410                         data_after = dut.register.counter_64bit;
411                         #1;
412                         if(data_after == data_before +1) begin
413                         $display("Pass, before = 32'h%h, after = 32'h%h", data_before, data_after);
414                         end else begin
415                         $display("Error, before = 32'h%h, after = 32'h%h", data_before, data_after);
416                         end
417
418
419                         @(posedge sys_clk);
420                         #1;
421                         tim_psel = 0;
422                         tim_pwrite = 0;
423                         tim_penable = 0;
424                         tim_paddr = 12'h0;
425                         tim_pwdata = 32'h0;
426                         tim_pstrb = 4'h0;
427                         data_after = 0;
428                         data_before = 0;
429                         #10;
430                 end
431         endtask
```

```
433
434         task check_int;
435
436                 input [11:0] address;
437                 input [31:0] data;
438                 input [3:0] strobe;
439                 input signal;
440                 begin
441                         @(posedge sys_clk);
442                         #1;
443                         tim_psel = 1;
444                         tim_pwrite = 1;
445                         tim_penable = 0;
446                         tim_paddr = address;
447                         tim_pwdata = data;
448                         tim_pstrb = strobe;
449
450                         @(posedge sys_clk);
451                         #1;
452                         tim_penable = 1;
453
454                         wait(tim_pready == 1);
455                         repeat(16)@(posedge sys_clk);
456                         #1;
457
458                         check_signal(signal ,tim_pready);
459                         check_signal(signal, dut.register.data6[0]);
460
461
462                         @(posedge sys_clk);
463                         #1;
464                         tim_psel = 0;
465                         tim_pwrite = 0;
466                         tim_penable = 0;
467                         tim_paddr = 12'h0;
468                         tim_pwdata = 32'h0;
469                         tim_pstrb = 4'h0;
470                         #10;
471                 end
472         endtask
```

```
474         task check_halt_reg;
475
476                 input [11:0] address;
477                 input [31:0] data;
478                 input [3:0] strobe;
479                 input signal;
480                 input [31:0] read_data;
481                 begin
482                         @(posedge sys_clk);
483                         #1;
484                         tim_psel = 1;
485                         tim_pwrite = 1;
486                         tim_penable = 0;
487                         tim_paddr = address;
488                         tim_pwdata = data;
489                         tim_pstrb = strobe;
490
491                         @(posedge sys_clk);
492                         #1;
493                         tim_penable = 1;
494
495                         @(posedge sys_clk);
496                         #1;
497                         if(read_data +1 == dut.register.data1) begin
498                         $display("Pass, before = %h, after = %h", read_data, dut.register.data1);
499                         end else begin
500                         $display("Error, before =%h, after =%h", read_data, dut.register.data1 );
501                         end
502                         #1;
503
504                         check_signal(signal ,dut.register.data7[0]);
505                         check_signal(signal, dut.register.data7[1]);
506
507
508                         @(posedge sys_clk);
509                         #1;
510                         tim_psel = 0;
511                         tim_pwrite = 0;
512                         tim_penable = 0;
513                         tim_paddr = 12'h0;
514                         tim_pwdata = 32'h0;
515                         tim_pstrb = 4'h0;
516                         #10;
517                 end
518         endtask
519 endmodule
```

Test case:

```
task run_test();
    reg [31:0]  read_data;
    begin
            $display("========================================");
            $display("=== Check APB Multi-access  ====");
            $display("========================================");


        //multiple APB access
        $display("multiple APB access");
        $display("write TDR0 & write TDR1");
        @(posedge test_bench.sys_clk);
        #1;

        test_bench.tim_psel = 1;
        test_bench.tim_pwrite = 1;
        test_bench.tim_penable = 0;
        test_bench.tim_paddr = 12'h004;
        test_bench.tim_pwdata = 32'h3333_3333;
        test_bench.tim_pstrb = 4'hffff;

        @(posedge test_bench.sys_clk);
        #1;
        test_bench.tim_penable = 1 ;

        wait( tim_pready == 1);
        @(posedge test_bench.sys_clk);
        #1;

        test_bench.tim_psel = 1;
        test_bench.tim_pwrite = 1;
        test_bench.tim_penable = 0;
        test_bench.tim_paddr = 12'h008;
        test_bench.tim_pwdata = 32'h5555_5555;
        test_bench.tim_pstrb = 4'hffff;


        @(posedge test_bench.sys_clk);
        #1;
        test_bench.tim_penable = 1 ;

        wait( tim_pready == 1);
        @(posedge test_bench.sys_clk);
        #1;

        test_bench.tim_psel = 0;
```

```
46          test_bench.tim_psel = 0;
47          test_bench.tim_pwrite = 0;
48          test_bench.tim_penable = 0;
49          test_bench.tim_paddr = 12'h000;
50          test_bench.tim_pwdata = 32'h0;
51          test_bench.tim_pstrb = 4'h0;
52
53
54          //normal read
55          $display("normal read");
56          test_bench.apb_read_register(12'h004, read_data);
57          test_bench.check_result(32'h3333_3333, read_data);
58
59          test_bench.apb_read_register(12'h008, read_data);
60          test_bench.check_result(32'h5555_5555, read_data);
61
62          //normal write
63          $display("normal write");
64          test_bench.apb_write_register(12'h004, 32'h1111_1111, 4'b1111);
65          test_bench.apb_write_register(12'h008, 32'h2222_2222, 4'b1111);
66
67        //multiple read
68          $display("multiple read");
69          @(posedge test_bench.sys_clk);
70          #1;
71          test_bench.tim_psel = 1;
72          test_bench.tim_paddr = 12'h004;
73        @(posedge test_bench.sys_clk);
74          #1;
75          test_bench.tim_penable = 1 ;
76
77          wait( tim_pready == 1);
78          #1;
79          read_data = tim_prdata;
80          test_bench.check_result(32'h1111_1111,read_data);
81
82          @(posedge test_bench.sys_clk);
83          #1;
84          test_bench.tim_penable = 0 ;
85          test_bench.tim_paddr = 12'h008;
86          @(posedge test_bench.sys_clk);
87          #1;
88          test_bench.tim_penable = 1 ;
89          wait( tim_pready == 1);
90          #1;
91        read_data = tim_prdata;
```

```
81
82          @(posedge test_bench.sys_clk);
83          #1;
84          test_bench.tim_penable = 0 ;
85          test_bench.tim_paddr = 12'h008;
86          @(posedge test_bench.sys_clk);
87          #1;
88          test_bench.tim_penable = 1 ;
89          wait( tim_pready == 1);
90          #1;
91          read_data = tim_prdata;
92          test_bench.check_result(32'h2222_2222,read_data);
93
94          @(posedge test_bench.sys_clk);
95          #1;
96          test_bench.tim_psel = 0;
97          test_bench.tim_penable = 0 ;
98          test_bench.tim_paddr = 12'h000;
99
100
101     end
102
103
104 endtask
~
```

```
task run_test();
    begin
                $display("======== check_apb_pready ==========");

        #100;
        test_bench.sys_rst_n = 1'b0;
        #100;
        @(posedge test_bench.sys_clk);
        #1;
        test_bench.sys_rst_n = 1'b1;


                #10;
                test_bench.check_pready(12'h000, 32'h0000_0003, 4'b1111, 1'b0, 1'b1);

                #10;
                test_bench.check_pready(12'h004, 32'h0000_0003, 4'b1111, 1'b1, 1'b1);

                #10;
                test_bench.check_pready(12'h008, 32'h0000_0005, 4'b1111, 1'b0, 1'b1);

                #10;
                test_bench.check_pready(12'h00C, 32'h0000_0002, 4'b1111, 1'b1, 1'b1);

                #10;
                test_bench.check_pready(12'h010, 32'h5A5A_5A5A, 4'b1111, 1'b0, 1'b1);


                #20;
                $display("Test pready completed\n");

        #100;
        test_bench.sys_rst_n = 1'b0;
        #100;
        @(posedge test_bench.sys_clk);
        #1;
        test_bench.sys_rst_n = 1'b1;
    end


    endtask


```

```
task run_test();
        reg [31:0] read_data;
        begin
            err = 0;
            $display("=====================================");
            $display("=== Check APB Protocol ====");
            $display("=====================================");

        //normal APB
        $display("\nNormal APB ");
        test_bench.apb_write_register(12'h004, 32'hffff_ffff, 4'b1111);
        test_bench.apb_read_register(12'h004, read_data);
        test_bench.check_result(32'hffff_ffff,read_data);

        //write wrong protocol
        $display("\nPenable does not assert & Psel assert");
        test_bench.apb_write_register_err(12'h004, 32'h5555_5555, 4'b1111, 0,1);
        //read wrong protocol
        test_bench.apb_read_register_err(12'h004, read_data,0,1);
        test_bench.check_result(32'h0000_0000,read_data);

        //read right protocol
        test_bench.apb_read_register(12'h004, read_data);
        test_bench.check_result(32'hffff_ffff,read_data);


        //write wrong protocol
        $display("\nPsel does not assert & Penable assert");
        test_bench.apb_write_register_err(12'h004, 32'h3333_3333, 4'b1111, 1,0);
        //read wrong protocol
        test_bench.apb_read_register_err(12'h004, read_data,1,0);
        test_bench.check_result(32'h0000_0000,read_data);

        //read right protocol
        test_bench.apb_read_register(12'h004, read_data);
        test_bench.check_result(32'hffff_ffff,read_data);


        end


endtask
~
```

```
task run_test();
        reg [31:0] read_data;
        integer i;
    begin
            $display("====================================");
            $display("=== Check Pslverr =======");
            $display("====================================");

        #100;
        test_bench.sys_rst_n = 1'b0;
        #100;
        @(posedge test_bench.sys_clk);
        #1;
        test_bench.sys_rst_n = 1'b1;

        $display("\n==== Write div_val >= 9, timer_en = 0, div_en =0 ==============");


        for(i = 9; i <16; i = i+1) begin
        $display("\n========== Check at div_val =%d ===========",i);
        test_bench.check_pslverr(12'h000, 32'h0000_0000, 4'b1111,i);

        end

        test_bench.apb_write_register(12'h000, 32'h0000_0102, 4'b1111);
        test_bench.apb_write_register(12'h000, 32'h0000_0103, 4'b1111);


        $display("\n==== Write div_en = 1 when timer_en = 1,div_val = 1 ==============");

        test_bench.check_pslverr(12'h000, 32'h0000_0903, 4'b1111,9);
        test_bench.apb_write_register(12'h000, 32'h0000_0102, 4'b1111);


// div_val <= 8;
        test_bench.apb_write_register(12'h000, 32'h0000_0003, 4'b1111);
        $display("\n==== Write div_val <= 8 ,div_en = 1 when timer_en = 1==============");


        for(i = 1; i <9; i = i+1) begin
        $display("\n========== Check at div_val =%d ===========",i);

        test_bench.check_pslverr(12'h000, 32'h0000_0003, 4'b1111,i);

        end

```

```
34
35 // div_val <= 8;
36         test_bench.apb_write_register(12'h000, 32'h0000_0003, 4'b1111);
37         $display("\n==== Write div_val <= 8 ,div_en = 1 when timer_en = 1==============");
38
39
40         for(i = 1; i <9; i = i+1) begin
41         $display("\n========== Check at div_val =%d ===========",i);
42
43         test_bench.check_pslverr(12'h000, 32'h0000_0003, 4'b1111,i);
44
45         end
46
47 // div_val >= 9;
48         test_bench.apb_write_register(12'h000, 32'h0000_0002, 4'b1111);
49
50         $display("\n==== Write div_val >= 9, timer_en = 1, div_en =1 ==============");
51
52         for(i = 9; i <16; i = i+1) begin
53         $display("\n========== Check at div_val =%d ===========",i);
54
55         test_bench.check_pslverr(12'h000, 32'h0000_0003, 4'b1111,i);
56
57         end
58
59     end
60
61
62 endtask
~
```

```
task run_test();
        reg [31:0] read_data;
    begin
            $display("====================================");
            $display("=== Check : APB Un-aligned ===");
            $display("====================================");

        //normal APB
        test_bench.apb_write_register(12'h004, 32'hAAAA_AAAA, 4'b1111);
        test_bench.apb_read_register(12'h004, read_data);
        test_bench.check_result(32'hAAAA_AAAA, read_data);


        //un-aligned
        $display("\n========== un-aligned ==========");
        $display("write & read at address: 12'h005");
        test_bench.apb_write_register(12'h005, 32'hAAAA_AAAA, 4'b1111);
        test_bench.apb_read_register(12'h005, read_data);
        test_bench.check_result(32'h0000_0000, read_data);
        test_bench.apb_read_register(12'h004, read_data);
        test_bench.check_result(32'hAAAA_AAAA, read_data);


        //un-aligned
        $display("write & read at address: 12'h006");
        test_bench.apb_write_register(12'h006, 32'h3333_5555, 4'b1111);
        test_bench.apb_read_register(12'h006, read_data);
        test_bench.check_result(32'h0000_0000, read_data);
        test_bench.apb_read_register(12'h004, read_data);
        test_bench.check_result(32'hAAAA_AAAA, read_data);

        //un-aligned
        $display("write & read at address: 12'h007");
        test_bench.apb_write_register(12'h007, 32'h2222_2222, 4'b1111);
        test_bench.apb_read_register(12'h007, read_data);
        test_bench.check_result(32'h0000_0000, read_data);
        test_bench.apb_read_register(12'h004, read_data);
        test_bench.check_result(32'hAAAA_AAAA, read_data);

    end


endtask
~
```

```
1
2  task run_test();
3          reg [31:0] read_data;
4          integer i;
5      begin
6
7              $display("================================================================");
8              $display("====== Check counter couting  ================");
9              $display("================================================================");
10
11
12         $display("===== Set counter_64bit =====");
13         test_bench.apb_write_register(12'h004, 32'hffff_fff0, 4'b1111);
14         test_bench.apb_write_register(12'h008, 32'hffff_ffff, 4'b1111);
15
16         $display("===== Check timer_en =1, timer_en = 0 =====");
17         test_bench.apb_write_register(12'h000, 32'h0000_0001, 4'b1111);
18           #1
19         $display(" cnt value: 64'h%h",test_bench.dut.register.counter_64bit);
20         $display("===== Wait finish 15 cycles =====");
21         repeat(14)@(posedge test_bench.sys_clk);
22          #1
23         if(test_bench.dut.register.counter_64bit == 64'hffff_ffff_ffff_ffff)
24          $display("PASS: Counter reach 64'hffff_ffff_ffff_ffff");
25          else
26          $display("Fail: Counter not reach 64'hffff_ffff_ffff_ffff");
27
28
29         $display("===== Clear TCR =====");
30         test_bench.apb_write_register(12'h000, 32'h0000_0000, 4'b1111);
31         #1;
32
33
34
35         $display("===== Set counter_64bit =====");
36         test_bench.apb_write_register(12'h004, 32'hffff_fff0, 4'b1111);
37         test_bench.apb_write_register(12'h008, 32'hffff_ffff, 4'b1111);
38
39         $display("===== Check timer_en =1, timer_en = 1, div_val = 0 =====");
40         test_bench.apb_write_register(12'h000, 32'h0000_0003, 4'b1111);
41          #1
42         $display(" cnt value: 64'h%h",test_bench.dut.register.counter_64bit);
43         $display("===== Wait finish 15 cycles =====");
44         repeat(14)@(posedge test_bench.sys_clk);
45          #1
46         if(test_bench.dut.register.counter_64bit == 64'hffff_ffff_ffff_ffff)
```

```
41          #1
42         $display(" cnt value: 64'h%h",test_bench.dut.register.counter_64bit);
43         $display("===== Wait finish 15 cycles =====");
44         repeat(14)@(posedge test_bench.sys_clk);
45          #1
46         if(test_bench.dut.register.counter_64bit == 64'hffff_ffff_ffff_ffff)
47          $display("PASS: Counter reach 64'hffff_ffff_ffff_ffff");
48          else
49          $display("Fail: Counter not reach 64'hffff_ffff_ffff_ffff");
50
51         $display("===== Clear TCR =====");
52         test_bench.apb_write_register(12'h000, 32'h0000_0002, 4'b1111);
53         #1;
54
55
56
57         for (i =1 ;i <9; i =i+1) begin
58         $display("===== Set counter_64bit =====");
59         test_bench.apb_write_register(12'h004, 32'hffff_fff0, 4'b1111);
60         test_bench.apb_write_register(12'h008, 32'hffff_ffff, 4'b1111);
61
62
63         $display("===== Check timer_en =1, timer_en = 1, div_val = %d =====", i);
64         test_bench.apb_write_register_div_val(12'h000, 32'h0000_0003, 4'b1111, i);
65         #1;
66         $display(" cnt value: 64'h%h",test_bench.dut.register.counter_64bit);
67         $display("===== Wait finish 15 cycles =====");
68         repeat(((1<<i))*15-1)@(posedge test_bench.sys_clk);
69          #1
70         if(test_bench.dut.register.counter_64bit == 64'hffff_ffff_ffff_ffff)
71          $display("PASS: Counter reach 64'hffff_ffff_ffff_ffff");
72          else
73          $display("Fail: Counter not reach 64'hffff_ffff_ffff_ffff");
74
75         $display("===== Clear TCR =====");
76         test_bench.apb_write_register_div_val(12'h000, 32'h0000_0002, 4'b1111, i);
77         #10;
78         end
79
80         for (i =1 ;i <9; i =i+1) begin
81         $display("===== Set counter_64bit =====");
82         test_bench.apb_write_register(12'h004, 32'hffff_ffff, 4'b1111);
83         test_bench.apb_write_register(12'h008, 32'hffff_ffff, 4'b1111);
84
85
```

```
85
86          $display("===== Check timer_en =1, timer_en = 1, div_val = %d =====", i);
87          test_bench.apb_write_register_div_val(12'h000, 32'h0000_0003, 4'b1111, i);
88          #1;
89          $display(" cnt value: 64'h%h",test_bench.dut.register.counter_64bit);
90          $display("===== Wait finish max cycles =====");
91          repeat(2**i -1)@(posedge test_bench.sys_clk);
92           #1
93          if(test_bench.dut.register.counter_64bit == 64'h0000_0000_0000_0000)
94           $display("PASS: Counter reach 64'h0000_0000_0000_0000");
95           else begin
96           $display("Fail: Counter not reach 64'h0000_0000_0000_0000");
97           test_bench.err = test_bench.err +1;
98          end
99
100         $display("===== Clear TCR =====");
101         test_bench.apb_write_register_div_val(12'h000, 32'h0000_0002, 4'b1111, i);
102         #10;
103
104         end
105
106     end
107 endtask
```

```
task run_test();

        begin
        $display("====== Check_cnt_ctr_chk =====");

        #10;
        $display("\n ------- div_val = 0 ,div_en = 0 -------");
        test_bench.check_timer_count(12'h000, 32'h0000_0001, 4'b1111, 4'h0);
        #10;
        test_bench.apb_write_register(12'h000, 32'h000_0000, 4'b1111);


        #10;
        $display("\n ------- div_val = 0 ,div_en = 1 -------");
        test_bench.check_timer_count(12'h000, 32'h0000_0003, 4'b1111, 4'h0);
        #10;
        test_bench.apb_write_register(12'h000, 32'h000_0002, 4'b1111);


        #10;
        $display("\n ------- div_val = 1 ,div_en = 1 -------");
        test_bench.check_timer_count(12'h000, 32'h0000_0103, 4'b1111, 4'h1);
        #10;
        test_bench.apb_write_register(12'h000, 32'h000_0102, 4'b1111);


        #10;
        $display("\n ------- div_val = 2 ,div_en = 1 -------");
        test_bench.check_timer_count(12'h000, 32'h0000_0203, 4'b1111, 4'h2);
        #10;
        test_bench.apb_write_register(12'h000, 32'h000_0202, 4'b1111);


        #10;
        $display("\n ------- div_val = 3 ,div_en = 1 -------");
        test_bench.check_timer_count(12'h000, 32'h0000_0303, 4'b1111, 4'h3);
        #10;
        test_bench.apb_write_register(12'h000, 32'h000_0302, 4'b1111);


        #10;
        $display("\n ------- div_val = 4 ,div_en = 1 -------");
        test_bench.check_timer_count(12'h000, 32'h0000_0403, 4'b1111, 4'h4);
        #10;
        test_bench.apb_write_register(12'h000, 32'h000_0402, 4'b1111);
```

```
47
48
49          #10;
50          $display("\n ------- div_val = 5 ,div_en = 1 -------");
51          test_bench.check_timer_count(12'h000, 32'h0000_0503, 4'b1111, 4'h5);
52          #10;
53          test_bench.apb_write_register(12'h000, 32'h000_0502, 4'b1111);
54
55
56          #10;
57          $display("\n ------- div_val = 6 ,div_en = 1 -------");
58          test_bench.check_timer_count(12'h000, 32'h0000_0603, 4'b1111, 4'h6);
59          #10;
60          test_bench.apb_write_register(12'h000, 32'h000_0602, 4'b1111);
61
62
63          #10;
64          $display("\n ------- div_val = 7 ,div_en = 1 -------");
65          test_bench.check_timer_count(12'h000, 32'h0000_0703, 4'b1111, 4'h7);
66          #10;
67          test_bench.apb_write_register(12'h000, 32'h000_0702, 4'b1111);
68
69
70          #10;
71          $display("\n ------- div_val = 8 ,div_en = 1 -------");
72          test_bench.check_timer_count(12'h000, 32'h0000_0803, 4'b1111, 4'h8);
73          #10;
74          test_bench.apb_write_register(12'h000, 32'h000_0802, 4'b1111);
75
76
77
78          #100;
79          test_bench.sys_rst_n = 1'b0;
80          #100;
81          @(posedge test_bench.sys_clk);
82          #1;
83          sys_rst_n = 1'b1;
84
85          $display(" Check_cnt_ctr_chk completed\n");
86                  end
87          endtask
88
```

```
task run_test();
    reg [31:0] read_data;
    reg [63:0] data_before, data_after;

    begin

        #100;
        test_bench.sys_rst_n = 1'b0;
        #100;
        @(posedge test_bench.sys_clk);
        #1;
        test_bench.sys_rst_n = 1'b1;



           $display("====================================");
            $display("====== Check: Cnt Halt Check ====");
            $display("====================================");

        $display("\n====== No Debug mode ==========");
        $display("\n====== Check hatl_req = 1, hatl_status = 0 ==========");

        test_bench.dbg_mode = 0;
        test_bench.apb_write_register(12'h01C, 32'h0000_0001, 4'b1111);
        test_bench.apb_write_register(12'h000, 32'h0000_0001, 4'b1111);


        if(test_bench.dut.register.data7[0] == 1 )begin
                $display("------------------------------------------------------------");
                $display(" PASSED: halt_req is asserted. halt_req= %d",test_bench.dut.register.data7[0]);
                $display("------------------------------------------------------------");
        end else begin
                $display("------------------------------------------------------------");
                $display(" FAIL: halt_req does not asserted. halt_req= %d",test_bench.dut.register.data7[0]);
                $display("------------------------------------------------------------");
                test_bench.err = test_bench.err + 1;
        end

//      Check hatl_status = 0
        test_bench.apb_read_register(12'h01C, read_data);
        test_bench.check_result(32'h0000_0001, read_data);

        if(test_bench.dut.register.thcr_halt_reg_pre == 0 )begin
                $display("------------------------------------------------------------");
                $display(" PASSED: halt_status does not asserted. halt_status= %d",test_bench.dut.register.thcr_halt_reg_pre );
                $display("------------------------------------------------------------");
```

```
46                 $display("------------------------------------------------------");
47         end else begin
48                 $display("------------------------------------------------------");
49                 $display(" FAIL: halt_status is asserted. halt_status= %d",test_bench.dut.register.thcr_halt_reg_pre );
50                 $display("------------------------------------------------------");
51                 test_bench.err = test_bench.err + 1;
52         end
53
54
55         $display("\n======= Check counter while non hatl ======");
56
57         repeat(5) @(posedge test_bench.sys_clk);
58         #1;
59         data_before = test_bench.dut.register.counter_64bit;
60         @(posedge test_bench.sys_clk);
61         #1;
62         data_after = test_bench.dut.register.counter_64bit;
63         if(data_after == data_before +1) begin
64                 $display("Pass, before = 32'h%h, after = 32'h%h", data_before, data_after);
65         end else begin
66                 $display("Error, before = 32'h%h, after = 32'h%h", data_before, data_after);
67                 test_bench.err = test_bench.err +1;
68         end
69
70         test_bench.apb_write_register(12'h000, 32'h0000_0000, 4'b1111);
71         test_bench.apb_write_register(12'h01C, 32'h0000_0000, 4'b1111);
72
73
74
75 // Test hatl mode
76         $display("\n====== Assert Debug mode ==========");
77         $display("\n====== Check hatl_req = 1, hatl_status = 1 ==========");
78
79         test_bench.apb_write_register(12'h000, 32'h0000_0303, 4'b1111);
80         repeat(24) @(posedge test_bench.sys_clk);
81
82         test_bench.dbg_mode = 1;
83         test_bench.apb_write_register(12'h01C, 32'h0000_0001, 4'b1111);
84
85         if(test_bench.dut.register.data7[0] == 1 )begin
86                 $display("------------------------------------------------------");
87                 $display(" PASSED: halt_req is asserted. halt_req= %d",test_bench.dut.register.data7[0]);
88                 $display("------------------------------------------------------");
89         end else begin
90                 $display("------------------------------------------------------");
91                 $display(" FAIL: halt_req does not asserted. halt_req= %d",test_bench.dut.register.data7[0]);
```

```
90                $display("------------------------------------------------------");
91                $display(" FAIL: halt_req does not asserted. halt_req= %d",test_bench.dut.register.data7[0]);
92                $display("------------------------------------------------------");
93                test_bench.err = test_bench.err + 1;
94        end
95
96 //     Check hatl_status = 1
97        test_bench.apb_read_register(12'h01C, read_data);
98        test_bench.check_result(32'h0000_0003, read_data);
99
100       if(test_bench.dut.register.thcr_halt_reg_pre  == 1 )begin
101               $display("------------------------------------------------------");
102               $display(" PASSED: halt_status is asserted. halt_status= %d",test_bench.dut.register.thcr_halt_reg_pre );
103               $display("------------------------------------------------------");
104       end else begin
105               $display("------------------------------------------------------");
106               $display(" FAIL: halt_status does not asserted. halt_status= %d",test_bench.dut.register.thcr_halt_reg_pre );
107               $display("------------------------------------------------------");
108               test_bench.err = test_bench.err + 1;
109       end
110
111
112       $display("\n======= Check counter while hatl ======");
113
114       repeat(25) @(posedge test_bench.sys_clk);
115       #1;
116       data_before = test_bench.dut.register.counter_64bit;
117       repeat(20) @(posedge test_bench.sys_clk);
118       #1;
119       data_after = test_bench.dut.register.counter_64bit;
120       if(data_after == data_before ) begin
121               $display("Pass, before = 32'h%h, after = 32'h%h", data_before, data_after);
122       end else begin
123               $display("Error, before = 32'h%h, after = 32'h%h", data_before, data_after);
124               test_bench.err = test_bench.err +1;
125       end
126
127       $display("\n======= Check counter after clear hatl ======");
128       test_bench.apb_write_register(12'h01C, 32'h0000_0000, 4'b1111);
129       test_bench.dbg_mode = 0;
130
131
132       repeat((2**3)-1) @(posedge test_bench.sys_clk);
133
134       data_after = test_bench.dut.register.counter_64bit;
135       if(data_after == data_before +1 ) begin
```

```
116         data_before = test_bench.dut.register.counter_64bit;
117         repeat(20) @(posedge test_bench.sys_clk);
118         #1;
119         data_after = test_bench.dut.register.counter_64bit;
120         if(data_after == data_before ) begin
121                 $display("Pass, before = 32'h%h, after = 32'h%h", data_before, data_after);
122         end else begin
123                 $display("Error, before = 32'h%h, after = 32'h%h", data_before, data_after);
124                 test_bench.err = test_bench.err +1;
125         end
126
127         $display("\n======= Check counter after clear hatl ======");
128         test_bench.apb_write_register(12'h01C, 32'h0000_0000, 4'b1111);
129         test_bench.dbg_mode = 0;
130
131
132         repeat((2**3)-1) @(posedge test_bench.sys_clk);
133
134         data_after = test_bench.dut.register.counter_64bit;
135         if(data_after == data_before +1 ) begin
136                 $display("Pass, before = 32'h%h, after = 32'h%h, counter correct", data_before, data_after);
137         end else begin
138                 $display("Error, before = 32'h%h, after = 32'h%h counter wrong", data_before, data_after);
139                 test_bench.err = test_bench.err +1;
140         end
141         @(posedge test_bench.sys_clk);
142
143         test_bench.apb_write_register(12'h000, 32'h0000_0302, 4'b1111);
144         test_bench.apb_write_register(12'h000, 32'h0000_0303, 4'b1111);
145
146         repeat(25) @(posedge test_bench.sys_clk);
147         $display("\n======= Check change value TRC while hatl ======");
148         test_bench.dbg_mode = 1;
149         test_bench.apb_write_register(12'h01C, 32'h0000_0001, 4'b1111);
150         test_bench.apb_write_register(12'h000, 32'h0000_0302, 4'b1111);
151         test_bench.apb_write_register(12'h000, 32'h0000_0002, 4'b1111);
152         test_bench.apb_write_register(12'h000, 32'h0000_0103, 4'b1111);
153
154         #100;
155         test_bench.sys_rst_n = 1'b0;
156         #100;
157         @(posedge test_bench.sys_clk);
158         #1;
159         test_bench.sys_rst_n = 1'b1;
160     end
161 endtask
```

```
task run_test();
        reg [31:0] read_data;
        reg [63:0] data_before, data_after;


    begin

            $display("====================================");
            $display("====== Interrupt check ===");
            $display("====================================");

        #100;
        test_bench.sys_rst_n = 1'b0;
        #100;
        @(posedge test_bench.sys_clk);
        #1;
        test_bench.sys_rst_n = 1'b1;

        $display("\n===== Set value TDR0, TDR1, TIER ,TRC =====");
        test_bench.apb_write_register(12'h004, 32'hffff_ffff, 4'b1111);
        test_bench.apb_write_register(12'h008, 32'hffff_ffff, 4'b1111);
        test_bench.apb_write_register(12'h014, 32'h0000_0001, 4'b1111);
        test_bench.apb_write_register(12'h000, 32'h0000_0001, 4'b1111);

        repeat(5) @(posedge test_bench.sys_clk);
        #1;

        if( test_bench.tim_int === 1 ) begin
                        $display("-------------------------------------------------------");
                        $display(" PASSED: interrupt is asserted. tim_int = %d",test_bench.tim_int );
                        $display("-------------------------------------------------------");

        end else begin
                        $display("-------------------------------------------------------");
                        $display(" FAILED: interrupt does not asserted. tim_int = %d",test_bench.tim_int);
                        $display("-------------------------------------------------------");
            test_bench.err = test_bench.err + 1;
        end

        repeat(5) @(posedge test_bench.sys_clk);
        #1;

        $display("\n======= Clear Interrupt Enable ======");
        test_bench.apb_write_register(12'h014, 32'h0000_0000, 4'b1111);


```

```
46
47          $display("\n======= Check interrupt status is 1 ======");
48          test_bench.apb_read_register(12'h018, read_data);
49          test_bench.check_result(32'h0000_0001,read_data);
50
51          if (test_bench.dut.register.data6 == 1) begin
52                  $display("-------------------------------------------------------");
53                  $display(" PASSED: interrupt status is asserted. interrupt trigger condition = %d",test_bench.dut.register.data6 );
54                  $display("-------------------------------------------------------");
55          end else begin
56                  $display("-------------------------------------------------------");
57                  $display(" FAIL: interrupt status does not asserted. interrupt trigger condition = %d",test_bench.dut.register.data6 );
58                  $display("-------------------------------------------------------");
59                  test_bench.err = test_bench.err + 1;
60          end
61
62          $display("\n======= Check counter while interrupt ======");
63
64          repeat(5) @(posedge test_bench.sys_clk);
65          #1;
66          data_before = test_bench.dut.register.counter_64bit;
67          @(posedge test_bench.sys_clk);
68          #1;
69          data_after = test_bench.dut.register.counter_64bit;
70          if(data_after == data_before +1) begin
71                  $display("Pass, before = 32'h%h, after = 32'h%h", data_before, data_after);
72          end else begin
73                  $display("Error, before = 32'h%h, after = 32'h%h", data_before, data_after);
74                  test_bench.err = test_bench.err +1;
75          end
76
77          $display("\n======= Clear interrupt status  ======");
78
79          test_bench.apb_write_register(12'h018, 32'h0000_0001, 4'b1111);
80          test_bench.apb_read_register(12'h018, read_data);
81          test_bench.check_result(32'h0000_0000,read_data);
82
83          if (test_bench.dut.register.data6 == 0) begin
84                  $display("-------------------------------------------------------");
85                  $display(" PASSED: interrupt status is cleared. interrupt trigger condition = %d",test_bench.dut.register.data6 );
86                  $display("-------------------------------------------------------");
87          end else begin
88                  $display("-------------------------------------------------------");
89                  $display(" Fail: interrupt status does not cleared. interrupt trigger condition = %d",test_bench.dut.register.data6 );
90                  $display("-------------------------------------------------------");
91          ▉       test_bench.err = test_bench.err + 1;
```

```
83          if (test_bench.dut.register.data6 == 0) begin
84                  $display("-------------------------------------------------------");
85                  $display(" PASSED: interrupt status is cleared. interrupt trigger condition = %d",test_bench.dut.register.data6 );
86                  $display("-------------------------------------------------------");
87          end else begin
88                  $display("-------------------------------------------------------");
89                  $display(" Fail: interrupt status does not cleared. interrupt trigger condition = %d",test_bench.dut.register.data6 );
90                  $display("-------------------------------------------------------");
91                  test_bench.err = test_bench.err + 1;
92          end
93          end
94  endtask
```

```
task run_test();
    reg [31:0]  read_data;
    begin

        $display("===================================");
        $display("====== Test One Hot Check ====");
        $display("===================================");


        test_bench.apb_write_register(12'h000, 32'h1111_1813, 4'hf);
        test_bench.apb_write_register(12'h004, 32'h3333_3333, 4'hf);
        test_bench.apb_write_register(12'h008, 32'hAAAA_AAAA, 4'hf);
        test_bench.apb_write_register(12'h00C, 32'hffff_ffff, 4'hf);
        test_bench.apb_write_register(12'h010, 32'h5555_5555, 4'hf);
        test_bench.apb_write_register(12'h014, 32'h0101_1010, 4'hf);
        test_bench.apb_write_register(12'h018, 32'h0505_0505, 4'hf);
        test_bench.apb_write_register(12'h01C, 32'hffff_ffff, 4'hf);

        test_bench.apb_read_register(12'h000, read_data);
        test_bench.check_result(32'h0000_0803,read_data);

        test_bench.apb_read_register(12'h004, read_data);
        test_bench.check_result(32'h3333_3333,read_data);

        test_bench.apb_read_register(12'h008, read_data);
        test_bench.check_result(32'hAAAA_AAAA, read_data);

        test_bench.apb_read_register(12'h00C, read_data);
        test_bench.check_result(32'hffff_ffff,read_data);

        test_bench.apb_read_register(12'h010, read_data);
        test_bench.check_result(32'h5555_5555,read_data);

        test_bench.apb_read_register(12'h014, read_data);
        test_bench.check_result(32'h0000_0000,read_data);

        test_bench.apb_read_register(12'h018, read_data);
        test_bench.check_result(32'h0000_0000, read_data);

        test_bench.apb_read_register(12'h01C, read_data);
        test_bench.check_result(32'h0000_0001,read_data);

        #100;
        test_bench.sys_rst_n = 1'b0;
        #100;
        @(posedge test_bench.sys_clk);
```

```
42
43          #100;
44          test_bench.sys_rst_n = 1'b0;
45          #100;
46          @(posedge test_bench.sys_clk);
47          #1;
48          sys_rst_n = 1'b1;
49
50          $display(" Test One Hot Check completed\n");
51
52      end
53  endtask
```

```
task run_test();
   reg [31:0] read_data;
    begin

           $display("====================================");
           $display("=== check byte access ===");
           $display("====================================");

       $display("\n------- TCR Check -------");
       $display("========== Access Byte 0 ==========");
       test_bench.apb_write_register(12'h000, 32'h0000_0802, 4'b0001);
       test_bench.apb_read_register(12'h000, read_data);
       test_bench.check_result(32'h0000_0102, read_data);


       $display("========== Access Byte 1 ==========");
       test_bench.apb_write_register(12'h000, 32'h0000_0503, 4'b0010);
       test_bench.apb_read_register(12'h000, read_data);
       test_bench.check_result(32'h0000_0502, read_data);


       $display("========== Access Byte 2 ==========");
       test_bench.apb_write_register(12'h000, 32'h0000_0503, 4'b0100);
       test_bench.apb_read_register(12'h000, read_data);
       test_bench.check_result(32'h0000_0502, read_data);


       $display("========== Access Byte 3 ==========");
       test_bench.apb_write_register(12'h000, 32'h0000_0803, 4'b1000);
       test_bench.apb_read_register(12'h000, read_data);
       test_bench.check_result(32'h0000_0502, read_data);


       $display("\n------- TDR0 Check -------");
       $display("========== Access Byte 0 ==========");
       test_bench.apb_write_register(12'h004, 32'h0000_0802, 4'b0001);
       test_bench.apb_read_register(12'h004, read_data);
       test_bench.check_result(32'h0000_0002, read_data);


       $display("========== Access Byte 1 ==========");
       test_bench.apb_write_register(12'h004, 32'h0000_0503, 4'b0010);
       test_bench.apb_read_register(12'h004, read_data);
       test_bench.check_result(32'h0000_0502, read_data);


```

```
46
47          $display("========== Access Byte 2 ==========");
48          test_bench.apb_write_register(12'h004, 32'hffff_ffff, 4'b0100);
49          test_bench.apb_read_register(12'h004, read_data);
50          test_bench.check_result(32'h00ff_0502, read_data);
51
52
53          $display("========== Access Byte 3 ==========");
54          test_bench.apb_write_register(12'h004, 32'haaaa_0803, 4'b1000);
55          test_bench.apb_read_register(12'h004, read_data);
56          test_bench.check_result(32'haaff_0502, read_data);
57
58
59
60          $display("\n------- TDR1 Check -------");
61          $display("========== Access Byte 0 ==========");
62          test_bench.apb_write_register(12'h008, 32'h0000_0802, 4'b0001);
63          test_bench.apb_read_register(12'h008, read_data);
64          test_bench.check_result(32'h0000_0002, read_data);
65
66
67          $display("========== Access Byte 1 ==========");
68          test_bench.apb_write_register(12'h008, 32'h0000_0503, 4'b0010);
69          test_bench.apb_read_register(12'h008, read_data);
70          test_bench.check_result(32'h0000_0502, read_data);
71
72
73          $display("========== Access Byte 2 ==========");
74          test_bench.apb_write_register(12'h008, 32'hffff_ffff, 4'b0100);
75          test_bench.apb_read_register(12'h008, read_data);
76          test_bench.check_result(32'h00ff_0502, read_data);
77
78
79          $display("========== Access Byte 3 ==========");
80          test_bench.apb_write_register(12'h008, 32'haaaa_0803, 4'b1000);
81          test_bench.apb_read_register(12'h008, read_data);
82          test_bench.check_result(32'haaff_0502, read_data);
83
84
85
86          $display("\n------- TCMP0 Check -------");
87          $display("========== Access Byte 0 ==========");
88          test_bench.apb_write_register(12'h00C, 32'h1111_1111, 4'b0001);
89          test_bench.apb_read_register(12'h00C, read_data);
90          test_bench.check_result(32'hffff_ff11, read_data);
```

```
85
86          $display("\n------- TCMP0 Check -------");
87          $display("========== Access Byte 0 ==========");
88          test_bench.apb_write_register(12'h00C, 32'h1111_1111, 4'b0001);
89          test_bench.apb_read_register(12'h00C, read_data);
90          test_bench.check_result(32'hffff_ff11, read_data);
91
92
93          $display("========== Access Byte 1 ==========");
94          test_bench.apb_write_register(12'h00C, 32'hffff_ffff, 4'b0010);
95          test_bench.apb_read_register(12'h00C, read_data);
96          test_bench.check_result(32'hffff_ff11, read_data);
97
98
99          $display("========== Access Byte 2 ==========");
100         test_bench.apb_write_register(12'h00C, 32'haaaa_aaaa, 4'b0100);
101         test_bench.apb_read_register(12'h00C, read_data);
102         test_bench.check_result(32'hffaa_ff11, read_data);
103
104
105         $display("========== Access Byte 3 ==========");
106         test_bench.apb_write_register(12'h00C, 32'h0505_0505, 4'b1000);
107         test_bench.apb_read_register(12'h00C, read_data);
108         test_bench.check_result(32'h05aa_ff11, read_data);
109
110
111
112         $display("\n------- TCMP1 Check -------");
113         $display("========== Access Byte 0 ==========");
114         test_bench.apb_write_register(12'h010, 32'h1111_1111, 4'b0001);
115         test_bench.apb_read_register(12'h010, read_data);
116         test_bench.check_result(32'hffff_ff11, read_data);
117
118
119         $display("========== Access Byte 1 ==========");
120         test_bench.apb_write_register(12'h010, 32'hffff_ffff, 4'b0010);
121         test_bench.apb_read_register(12'h010, read_data);
122         test_bench.check_result(32'hffff_ff11, read_data);
123
124
125         $display("========== Access Byte 2 ==========");
126         test_bench.apb_write_register(12'h010, 32'haaaa_aaaa, 4'b0100);
127         test_bench.apb_read_register(12'h010, read_data);
128         test_bench.check_result(32'hffaa_ff11, read_data);
129
130
```

```
130
131         $display("========== Access Byte 3 ==========");
132         test_bench.apb_write_register(12'h010, 32'h0505_0505, 4'b1000);
133         test_bench.apb_read_register(12'h010, read_data);
134         test_bench.check_result(32'h05aa_ff11, read_data);
135
136
137
138         $display("\n------- TIER Check -------");
139         $display("========== Access Byte 0 ==========");
140         test_bench.apb_write_register(12'h014, 32'h1111_1111, 4'b0001);
141         test_bench.apb_read_register(12'h014, read_data);
142         test_bench.check_result(32'h0000_0001, read_data);
143
144
145         $display("========== Access Byte 1 ==========");
146         test_bench.apb_write_register(12'h014, 32'hffff_ffff, 4'b0010);
147         test_bench.apb_read_register(12'h014, read_data);
148         test_bench.check_result(32'h0000_0001, read_data);
149
150
151         $display("========== Access Byte 2 ==========");
152         test_bench.apb_write_register(12'h014, 32'haaaa_aaaa, 4'b0100);
153         test_bench.apb_read_register(12'h014, read_data);
154         test_bench.check_result(32'h0000_0001, read_data);
155
156
157         $display("========== Access Byte 3 ==========");
158         test_bench.apb_write_register(12'h014, 32'h0505_0505, 4'b1000);
159         test_bench.apb_read_register(12'h014, read_data);
160         test_bench.check_result(32'h0000_0001, read_data);
161
162
163
164         $display("\n------- TISR Check -------");
165         $display("========== Access Byte 0 ==========");
166         test_bench.apb_write_register(12'h018, 32'h1111_1111, 4'b0001);
167         test_bench.apb_read_register(12'h018, read_data);
168         test_bench.check_result(32'h0000_0000, read_data);
169
170
171         $display("========== Access Byte 1 ==========");
172         test_bench.apb_write_register(12'h018, 32'hffff_ffff, 4'b0010);
173         test_bench.apb_read_register(12'h018, read_data);
174         test_bench.check_result(32'h0000_0000, read_data);
```

```
174         test_bench.check_result(32'h0000_0000, read_data);
175
176
177         $display("========== Access Byte 2 ==========");
178         test_bench.apb_write_register(12'h018, 32'haaaa_aaaa, 4'b0100);
179         test_bench.apb_read_register(12'h018, read_data);
180         test_bench.check_result(32'h0000_0000, read_data);
181
182
183         $display("========== Access Byte 3 ==========");
184         test_bench.apb_write_register(12'h018, 32'h0505_0505, 4'b1000);
185         test_bench.apb_read_register(12'h018, read_data);
186         test_bench.check_result(32'h0000_0000, read_data);
187
188
189         $display("\n------- THCSR Check -------");
190         $display("========== Access Byte 0 ==========");
191         test_bench.apb_write_register(12'h01C, 32'h1111_1111, 4'b0001);
192         test_bench.apb_read_register(12'h01C, read_data);
193         test_bench.check_result(32'h0000_0001, read_data);
194
195
196         $display("========== Access Byte 1 ==========");
197         test_bench.apb_write_register(12'h01C, 32'hffff_ffff, 4'b0010);
198         test_bench.apb_read_register(12'h01C, read_data);
199         test_bench.check_result(32'h0000_0001, read_data);
200
201
202         $display("========== Access Byte 2 ==========");
203         test_bench.apb_write_register(12'h01C, 32'haaaa_aaaa, 4'b0100);
204         test_bench.apb_read_register(12'h01C, read_data);
205         test_bench.check_result(32'h0000_0001, read_data);
206
207
208         $display("========== Access Byte 3 ==========");
209         test_bench.apb_write_register(12'h01C, 32'h0505_0505, 4'b1000);
210         test_bench.apb_read_register(12'h01C, read_data);
211         test_bench.check_result(32'h0000_0001, read_data);
212
213
214         #100;
215         test_bench.sys_rst_n = 1'b0;
216         #100;
217         @(posedge test_bench.sys_clk);
218         #1;
219         sys_rst_n = 1'b1;
```

```
188
189         $display("\n------- THCSR Check -------");
190         $display("========== Access Byte 0 ==========");
191         test_bench.apb_write_register(12'h01C, 32'h1111_1111, 4'b0001);
192         test_bench.apb_read_register(12'h01C, read_data);
193         test_bench.check_result(32'h0000_0001, read_data);
194
195
196         $display("========== Access Byte 1 ==========");
197         test_bench.apb_write_register(12'h01C, 32'hffff_ffff, 4'b0010);
198         test_bench.apb_read_register(12'h01C, read_data);
199         test_bench.check_result(32'h0000_0001, read_data);
200
201
202         $display("========== Access Byte 2 ==========");
203         test_bench.apb_write_register(12'h01C, 32'haaaa_aaaa, 4'b0100);
204         test_bench.apb_read_register(12'h01C, read_data);
205         test_bench.check_result(32'h0000_0001, read_data);
206
207
208         $display("========== Access Byte 3 ==========");
209         test_bench.apb_write_register(12'h01C, 32'h0505_0505, 4'b1000);
210         test_bench.apb_read_register(12'h01C, read_data);
211         test_bench.check_result(32'h0000_0001, read_data);
212
213
214         #100;
215         test_bench.sys_rst_n = 1'b0;
216         #100;
217         @(posedge test_bench.sys_clk);
218         #1;
219         sys_rst_n = 1'b1;
220
221         $display(" check byte access completed\n");
222
223     end
224
225
226 endtask
```

```
task run_test();
    reg [31:0]  read_data;
    begin

            $display("====================================");
            $display("==== reg_init_chk ===");
            $display("====================================");

               $display("------- TCR -------");
               test_bench.apb_read_register(12'h000, read_data);
               test_bench.check_result(32'h0000_0100, read_data);


               $display("------- TDR0 -------");
               test_bench.apb_read_register(12'h004, read_data);
               test_bench.check_result(32'h0000_0000, read_data);

               $display("------- TDR1 -------");
               test_bench.apb_read_register(12'h008, read_data);
               test_bench.check_result(32'h0000_0000, read_data);

               $display("------- TCMP0 -------");
               test_bench.apb_read_register(12'h00C, read_data);
               test_bench.check_result(32'hffff_ffff, read_data);

               $display("------- TCMP1 -------");
               test_bench.apb_read_register(12'h010, read_data);
               test_bench.check_result(32'hffff_ffff, read_data);

               $display("------- TIER -------");
               test_bench.apb_read_register(12'h014, read_data);
               test_bench.check_result(32'h0000_0000, read_data);

               $display("------- TISR -------");
               test_bench.apb_read_register(12'h018, read_data);
               test_bench.check_result(32'h0000_0000, read_data);

               $display("------- THCSR -------");
               test_bench.apb_read_register(12'h01C, read_data);
               test_bench.check_result(32'h0000_0000, read_data);

               #100;
               test_bench.sys_rst_n = 1'b0;
               #100;
               @(posedge test_bench.sys_clk);
               #1;
```

```
                #100;
                test_bench.sys_rst_n = 1'b0;
                #100;
                @(posedge test_bench.sys_clk);
                #1;
                sys_rst_n = 1'b1;

                $display(" reg_init_chk completed\n");

     end
    endtask
```

```
task run_test();

    reg [31:0] read_data;
    begin

            $display("====================================");
            $display("==== Check Reserved =====");
            $display("====================================");


                    $display("------- Random Address -------");
                    test_bench.apb_write_register(12'h015, 32'h0000_1234, 4'b1111);
                    test_bench.apb_read_register(12'h015, read_data);
                    test_bench.check_result(32'h0000_0000, read_data);

                    test_bench.apb_write_register(12'haaa, 32'hffff_ffff, 4'b1111);
                    test_bench.apb_read_register(12'haaa, read_data);
                    test_bench.check_result(32'h0000_0000,read_data);

                    test_bench.apb_write_register(12'h567, 32'h5555_5555, 4'b1111);
                    test_bench.apb_read_register(12'h567, read_data);
                    test_bench.check_result(32'h0000_0000,read_data);

                    test_bench.apb_write_register(12'h111, 32'hAAAA_AAAA, 4'b1111);
                    test_bench.apb_read_register(12'h111, read_data);
                    test_bench.check_result(32'h0000_0000,read_data);

                    test_bench.apb_write_register(12'hfff, 32'h0000_0500, 4'b1111);
                    test_bench.apb_read_register(12'hfff, read_data);
                    test_bench.check_result(32'h00000_0000,read_data);


                    #100;
                    test_bench.sys_rst_n = 1'b0;
                    #100;
                    @(posedge test_bench.sys_clk);
                    #1;
                    sys_rst_n = 1'b1;

                    $display(" Check Reserved completed\n");


    end
endtask
```

```
task run_test;
        reg [31:0] read_data;
        begin
                $display("====== Check_register_write_read =====");

                $display("------- TCR Check -------");
                test_bench.apb_write_register(12'h000, 32'h0000_0000, 4'b1111);
                test_bench.apb_read_register(12'h000, read_data);
                test_bench.check_result(32'h0000_0000, read_data);

                test_bench.apb_write_register(12'h000, 32'hffff_ffff, 4'b1111);
                test_bench.apb_read_register(12'h000, read_data);
                test_bench.check_result(32'h0000_0000,read_data);

                test_bench.apb_write_register(12'h000, 32'h5555_5555, 4'b1111);
                test_bench.apb_read_register(12'h000, read_data);
                test_bench.check_result(32'h0000_0501,read_data);

                test_bench.apb_write_register(12'h000, 32'hAAAA_AAAA, 4'b1111);
                test_bench.apb_read_register(12'h000, read_data);
                test_bench.check_result(32'h0000_0501,read_data);

                test_bench.apb_write_register(12'h000, 32'h0000_0500, 4'b1111);
                test_bench.apb_read_register(12'h000, read_data);
                test_bench.check_result(32'h00000_0500,read_data);


                $display("------- TDR0 -------");
                test_bench.apb_write_register(12'h004, 32'h0000_0000, 4'b1111);
                test_bench.apb_read_register(12'h004, read_data);
                test_bench.check_result(32'h0000_0000, read_data);

                test_bench.apb_write_register(12'h004, 32'hffff_ffff, 4'b1111);
                test_bench.apb_read_register(12'h004, read_data);
                test_bench.check_result(32'hffff_ffff,read_data);

                test_bench.apb_write_register(12'h004, 32'h5555_5555, 4'b1111);
                test_bench.apb_read_register(12'h004, read_data);
                test_bench.check_result(32'h5555_5555,read_data);

                test_bench.apb_write_register(12'h004, 32'hAAAA_AAAA, 4'b1111);
                test_bench.apb_read_register(12'h004, read_data);
                test_bench.check_result(32'hAAAA_AAAA, read_data);

```

```
51
52                      $display("------- TDR1 -------");
53                      test_bench.apb_write_register(12'h008, 32'h0000_0000, 4'b1111);
54                      test_bench.apb_read_register(12'h008, read_data);
55                      test_bench.check_result(32'h0000_0000, read_data);
56
57                      test_bench.apb_write_register(12'h008, 32'hffff_ffff, 4'b1111);
58                      test_bench.apb_read_register(12'h008, read_data);
59                      test_bench.check_result(32'hffff_ffff,read_data);
60
61                      test_bench.apb_write_register(12'h008, 32'h5555_5555, 4'b1111);
62                      test_bench.apb_read_register(12'h008, read_data);
63                      test_bench.check_result(32'h5555_5555,read_data);
64
65                      test_bench.apb_write_register(12'h008, 32'hAAAA_AAAA, 4'b1111);
66                      test_bench.apb_read_register(12'h008, read_data);
67                      test_bench.check_result(32'hAAAA_AAAA, read_data);
68
69                      test_bench.apb_write_register(12'h008, 32'h3333_3333, 4'b1111);
70                      test_bench.apb_read_register(12'h008, read_data);
71                      test_bench.check_result(32'h3333_3333,read_data);
72
73                      $display("------- TCMP0 -------");
74                      test_bench.apb_write_register(12'h00C, 32'h0000_0000, 4'b1111);
75                      test_bench.apb_read_register(12'h00C, read_data);
76                      test_bench.check_result(32'h0000_0000, read_data);
77
78                      test_bench.apb_write_register(12'h00C, 32'hffff_ffff, 4'b1111);
79                      test_bench.apb_read_register(12'h00C, read_data);
80                      test_bench.check_result(32'hffff_ffff,read_data);
81
82                      test_bench.apb_write_register(12'h00C, 32'h5555_5555, 4'b1111);
83                      test_bench.apb_read_register(12'h00C, read_data);
84                      test_bench.check_result(32'h5555_5555,read_data);
85
86                      test_bench.apb_write_register(12'h00C, 32'hAAAA_AAAA, 4'b1111);
87                      test_bench.apb_read_register(12'h00C, read_data);
88                      test_bench.check_result(32'hAAAA_AAAA, read_data);
89
90                      test_bench.apb_write_register(12'h00C, 32'h3333_333a, 4'b1111);
91                      test_bench.apb_read_register(12'h00C, read_data);
92                      test_bench.check_result(32'h3333_333a,read_data);
93
94                      $display("------- TCMP1 -------");
95                      test_bench.apb_write_register(12'h010, 32'h0000_0000, 4'b1111);
96                      test_bench.apb_read_register(12'h010, read_data);
```

```
94                      $display("------- TCMP1 -------");
95                      test_bench.apb_write_register(12'h010, 32'h0000_0000, 4'b1111);
96                      test_bench.apb_read_register(12'h010, read_data);
97                      test_bench.check_result(32'h0000_0000, read_data);
98
99                      test_bench.apb_write_register(12'h010, 32'hffff_ffff, 4'b1111);
100                     test_bench.apb_read_register(12'h010, read_data);
101                     test_bench.check_result(32'hffff_ffff,read_data);
102
103                     test_bench.apb_write_register(12'h010, 32'h5555_5555, 4'b1111);
104                     test_bench.apb_read_register(12'h010, read_data);
105                     test_bench.check_result(32'h5555_5555,read_data);
106
107                     test_bench.apb_write_register(12'h010, 32'hAAAA_AAAA, 4'b1111);
108                     test_bench.apb_read_register(12'h010, read_data);
109                     test_bench.check_result(32'hAAAA_AAAA, read_data);
110
111                     test_bench.apb_write_register(12'h010, 32'h3333_3333, 4'b1111);
112                     test_bench.apb_read_register(12'h010, read_data);
113                     test_bench.check_result(32'h3333_3333,read_data);
114
115                     $display("------- TIER -------");
116                     test_bench.apb_write_register(12'h014, 32'h0000_0000, 4'b1111);
117                     test_bench.apb_read_register(12'h014, read_data);
118                     test_bench.check_result(32'h0000_0000, read_data);
119
120                     test_bench.apb_write_register(12'h014, 32'hffff_ffff, 4'b1111);
121                     test_bench.apb_read_register(12'h014, read_data);
122                     test_bench.check_result(32'h0000_0001,read_data);
123
124                     test_bench.apb_write_register(12'h014, 32'h5555_5555, 4'b1111);
125                     test_bench.apb_read_register(12'h014, read_data);
126                     test_bench.check_result(32'h0000_0001,read_data);
127
128                     test_bench.apb_write_register(12'h014, 32'hAAAA_AAAA, 4'b1111);
129                     test_bench.apb_read_register(12'h014, read_data);
130                     test_bench.check_result(32'h0000_0000,read_data);
131
132                     test_bench.apb_write_register(12'h014, 32'h3333_3333, 4'b1111);
133                     test_bench.apb_read_register(12'h014, read_data);
134                     test_bench.check_result(32'h0000_0001,read_data);
135
```

```
136
137                     $display("------- TISR -------");
138                     test_bench.apb_write_register(12'h018, 32'h0000_0000, 4'b1111);
139                     test_bench.apb_read_register(12'h018, read_data);
140                     test_bench.check_result(32'h0000_0000, read_data);
141
142                     test_bench.apb_write_register(12'h018, 32'hffff_ffff, 4'b1111);
143                     test_bench.apb_read_register(12'h018, read_data);
144                     test_bench.check_result(32'h0000_0000,read_data);
145
146                     test_bench.apb_write_register(12'h018, 32'h5555_5555, 4'b1111);
147                     test_bench.apb_read_register(12'h018, read_data);
148                     test_bench.check_result(32'h0000_0000,read_data);
149
150                     test_bench.apb_write_register(12'h018, 32'hAAAA_AAAA, 4'b1111);
151                     test_bench.apb_read_register(12'h018, read_data);
152                     test_bench.check_result(32'h0000_0000,read_data);
153
154                     test_bench.apb_write_register(12'h018, 32'h3333_3333, 4'b1111);
155                     test_bench.apb_read_register(12'h018, read_data);
156                     test_bench.check_result(32'h0000_0000,read_data);
157
158
159                     $display("------- THCSR -------");
160                     test_bench.apb_write_register(12'h01C, 32'h0000_0000, 4'b1111);
161                     test_bench.apb_read_register(12'h01C, read_data);
162                     test_bench.check_result(32'h0000_0000, read_data);
163
164                     test_bench.apb_write_register(12'h01C, 32'hffff_ffff, 4'b1111);
165                     test_bench.apb_read_register(12'h01C, read_data);
166                     test_bench.check_result(32'h0000_0001,read_data);
167
168                     test_bench.apb_write_register(12'h01C, 32'h5555_5555, 4'b1111);
169                     test_bench.apb_read_register(12'h01C, read_data);
170                     test_bench.check_result(32'h0000_0001,read_data);
171
172                     test_bench.apb_write_register(12'h01C, 32'hAAAA_AAAA, 4'b1111);
173                     test_bench.apb_read_register(12'h01C, read_data);
174                     test_bench.check_result(32'h0000_0000,read_data);
175
176                     test_bench.apb_write_register(12'h01C, 32'h3333_3333, 4'b1111);
177                     test_bench.apb_read_register(12'h01C, read_data);
178                     test_bench.check_result(32'h0000_0001,read_data);
179
180                     #100;
181                     test_bench.sys_rst_n = 1'b0;
180                       #100;
181                       test_bench.sys_rst_n = 1'b0;
182                       #100;
183                       @(posedge test_bench.sys_clk);
184                       #1;
185                       test_bench.sys_rst_n = 1'b1;
186
187                       $display(" Check_register_write_read completed\n");
188                 end
189         endtask
190
```

Golden module:

```
haonguyen_1905@ictc-eda-ldap:~/timer_advanced_haonguyen/sim$ ./report.csh
reg_init_chk
reg_rw_chk
reg_reserved_chk
reg_1hot_chk
reg_byte_access
cnt_ctrl_chk
apb_protocol_chk
apb_multiple_access
apb_unaligned_chk
cnt_counting_chk
interrupt_chk
cnt_halt_chk
apb_pslverr_chk
|-----------------------------------------------------------------------------------------|
| PAT_NAME                          | RUN_DATE                    | RESULT            |
|-----------------------------------------------------------------------------------------|
| reg_init_chk                      | 21:41:36 Dec 13 2025        | PASSED            |
|-----------------------------------------------------------------------------------------|
| reg_rw_chk                        | 21:41:38 Dec 13 2025        | PASSED            |
|-----------------------------------------------------------------------------------------|
| reg_reserved_chk                  | 21:41:40 Dec 13 2025        | PASSED            |
|-----------------------------------------------------------------------------------------|
| reg_1hot_chk                      | 21:41:42 Dec 13 2025        | PASSED            |
|-----------------------------------------------------------------------------------------|
| reg_byte_access                   | 21:41:44 Dec 13 2025        | PASSED            |
|-----------------------------------------------------------------------------------------|
| cnt_ctrl_chk                      | 21:42:02 Dec 13 2025        | PASSED            |
|-----------------------------------------------------------------------------------------|
| apb_protocol_chk                  | 21:42:04 Dec 13 2025        | PASSED            |
|-----------------------------------------------------------------------------------------|
| apb_multiple_access               | 21:42:06 Dec 13 2025        | PASSED            |
|-----------------------------------------------------------------------------------------|
| apb_unaligned_chk                 | 21:42:08 Dec 13 2025        | PASSED            |
|-----------------------------------------------------------------------------------------|
| cnt_counting_chk                  | 21:42:10 Dec 13 2025        | PASSED            |
|-----------------------------------------------------------------------------------------|
| interrupt_chk                     | 21:42:12 Dec 13 2025        | PASSED            |
|-----------------------------------------------------------------------------------------|
| cnt_halt_chk                      | 21:42:30 Dec 13 2025        | PASSED            |
|-----------------------------------------------------------------------------------------|
| apb_pslverr_chk                   | 21:42:32 Dec 13 2025        | PASSED            |
|-----------------------------------------------------------------------------------------|
TOTAL/PASSED/REMAIN:13/13/0
haonguyen_1905@ictc-eda-ldap:~/timer_advanced_haonguyen/sim$
```

cov_test bench:

```
haonguyen_1905@ictc-eda-ldap:~/final_project_advanced/sim$ ./report.csh
reg_init_chk
reg_rw_chk
reg_reserved_chk
reg_1hot_chk
reg_byte_access
cnt_ctrl_chk
apb_protocol_chk
apb_multiple_access
apb_unaligned_chk
cnt_counting_chk
interrupt_chk
cnt_halt_chk
apb_pslverr_chk
apb_pready_chk
|-------------------------------------------------------------------------------------|
| PAT_NAME                          | RUN_DATE                    | RESULT        |
|-------------------------------------------------------------------------------------|
| reg_init_chk                      | 21:46:54 Dec 13 2025        | PASSED        |
|-------------------------------------------------------------------------------------|
| reg_rw_chk                        | 21:46:56 Dec 13 2025        | PASSED        |
|-------------------------------------------------------------------------------------|
| reg_reserved_chk                  | 21:46:58 Dec 13 2025        | PASSED        |
|-------------------------------------------------------------------------------------|
| reg_1hot_chk                      | 21:47:00 Dec 13 2025        | PASSED        |
|-------------------------------------------------------------------------------------|
| reg_byte_access                   | 21:47:02 Dec 13 2025        | PASSED        |
|-------------------------------------------------------------------------------------|
| cnt_ctrl_chk                      | 21:47:04 Dec 13 2025        | PASSED        |
|-------------------------------------------------------------------------------------|
| apb_protocol_chk                  | 21:47:06 Dec 13 2025        | PASSED        |
|-------------------------------------------------------------------------------------|
| apb_multiple_access               | 21:47:08 Dec 13 2025        | PASSED        |
|-------------------------------------------------------------------------------------|
| apb_unaligned_chk                 | 21:47:10 Dec 13 2025        | PASSED        |
|-------------------------------------------------------------------------------------|
| cnt_counting_chk                  | 21:47:12 Dec 13 2025        | PASSED        |
|-------------------------------------------------------------------------------------|
| interrupt_chk                     | 21:47:14 Dec 13 2025        | PASSED        |
|-------------------------------------------------------------------------------------|
| cnt_halt_chk                      | 21:47:16 Dec 13 2025        | PASSED        |
|-------------------------------------------------------------------------------------|
| apb_pslverr_chk                   | 21:47:18 Dec 13 2025        | PASSED        |
|-------------------------------------------------------------------------------------|
| apb_pready_chk                    | 21:47:21 Dec 13 2025        | PASSED        |
|-------------------------------------------------------------------------------------|
TOTAL/PASSED/REMAIN:14/14/0
haonguyen_1905@ictc-eda-ldap:~/final_project_advanced/sim$
```

detail_report.txt **summary_report.txt**

```
Coverage Report Summary Data by instance

================================================================================
=== Instance: /test_bench/dut/apb
=== Design Unit: work.apb_slave
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         7         7         0   100.00%
    Conditions                       9         9         0   100.00%
    Expressions                      8         8         0   100.00%
    Statements                      13        13         0   100.00%
    Toggles                        350       350         0   100.00%

================================================================================
=== Instance: /test_bench/dut/register
=== Design Unit: work.reg_module
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        78        78         0   100.00%
    Conditions                      67        67         0   100.00%
    Expressions                     17        17         0   100.00%
    Statements                      77        77         0   100.00%
    Toggles                        940       940         0   100.00%

================================================================================
=== Instance: /test_bench/dut
=== Design Unit: work.timer_top
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                        358       358         0   100.00%

================================================================================
=== Instance: /test_bench
=== Design Unit: work.test_bench
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        41        19        22    46.34%
    Conditions                      16         0        16     0.00%
    Statements                     508       442        66    87.00%
    Toggles                        242       178        64    73.55%


Total Coverage By Instance (filtered view): 89.98%

```