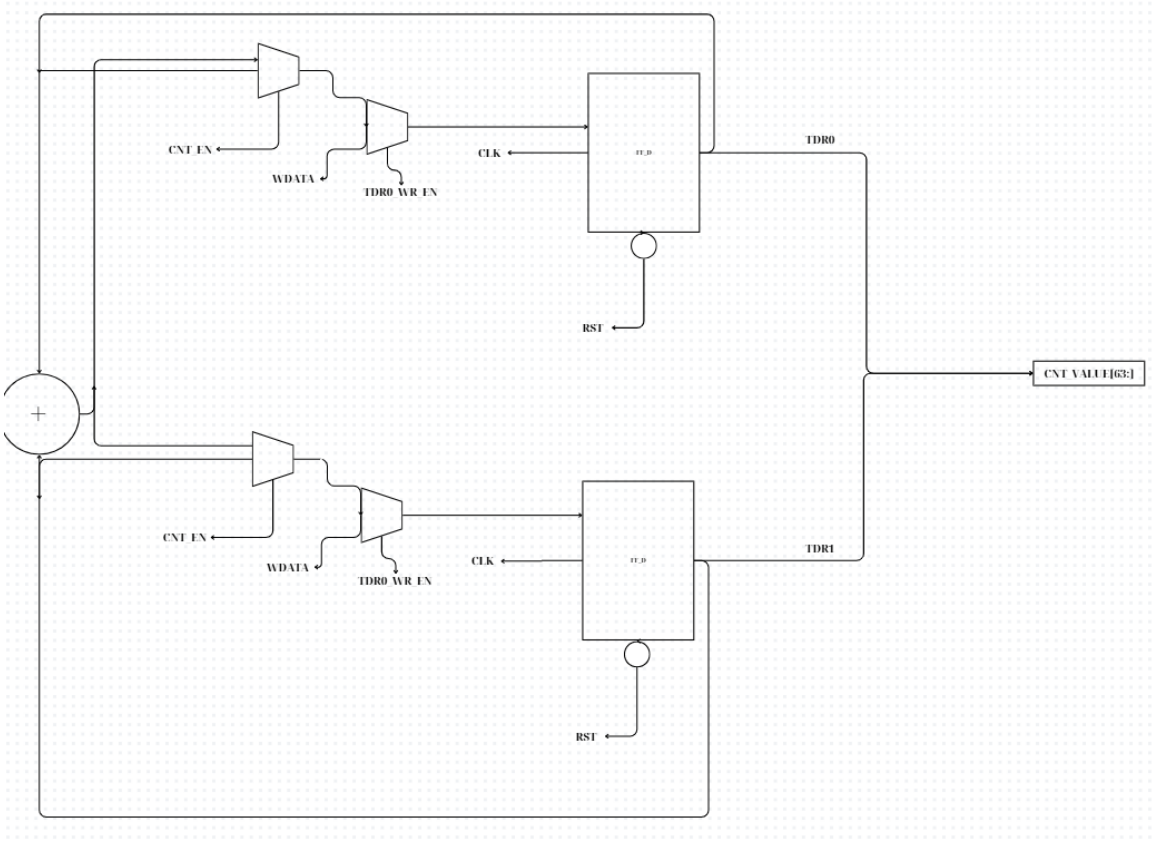
Báo Cáo Đồ Án Timer IP - Standard Level

I. MÔ TẢ:

Module counter



Khối counter là bộ đếm 64-bit dùng để đếm thời gian. Mỗi khi tín hiệu cnt_en bật (ở mức 1), bộ đếm sẽ tăng lên 1 giá trị theo từng chu kỳ xung clk. Kết quả đếm được xuất ra qua cnt_value để các khối khác sử dụng, đặc biệt là để so sánh với giá trị TCMP0, TCMP1 nhằm kích hoạt ngắt tim_int.

Trong phiên bản standard level, mặc dù counter có cổng đầu vào tdr0 và tdr1, nhưng không sử dụng chúng trong logic hoạt động. Việc khởi tạo lại counter (nếu cần) phải được thực hiện bằng cách ghi giá trị 0 vào cnt_value thông qua tín hiệu reset (rst_n = 0). Counter không tự nạp lại giá trị từ TDR0/TDR1 khi timer_en xuống thấp.

Tín hiệu I/O:

Tín hiệu	I/O	Độ rộng	Mô tả
Clk	In	1 bit	Xung clock hệ thống
rst_n	In	1 bit	Tín hiệu reset bất đồng bộ, mức thấp
cnt_en	In	1 bit	Cho phép đếm; khi = 1 thì bộ đếm hoạt động
tdr0	In	32 bit	Giá trị phần thấp (không dùng trong standard)
tdr1	In	32 bit	Giá trị phần cao (không dùng trong standard)
cnt_value	Out	64 bit	Giá trị hiện tại của bộ đếm

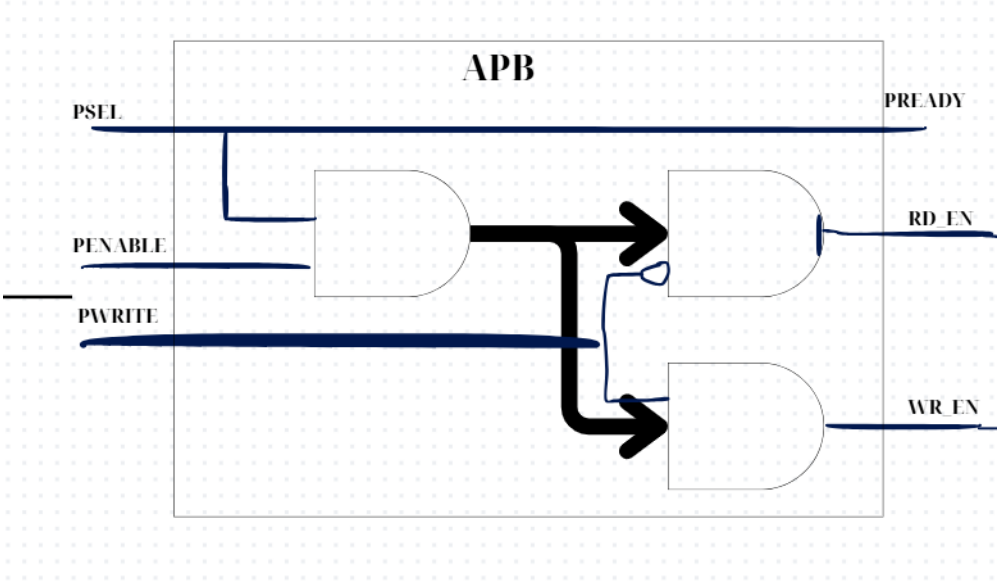
Tín hiệu kết nối với khối khác:

Tín hiệu	Tới/Từ khối
Clk	timer_top
rst_n	timer_top
cnt_en	Từ control_counter
tdr0, tdr1	Từ register_file
cnt_value	Tới register_file

Nguyên lý hoạt động:

Khi rst_n = 0, bộ đếm được đưa về 0.
Khi cnt_en = 1, mỗi chu kỳ xung clock, giá trị cnt_value sẽ tăng thêm 1.
Nếu cnt_en = 0, bộ đếm giữ nguyên giá trị hiện tại.

Module apb_slave



Khối apb_slave nhận tín hiệu điều khiển từ bus APB và tạo ra tín hiệu ghi (wr_en) hoặc đọc (rd_en) tới thanh ghi bên trong. Khối này không chứa thanh ghi lưu trữ mà chỉ xử lý logic truy cập.

Tín hiệu I/O:

Tín hiệu	I/O	Độ rộng	Mô tả
Clk	In	1 bit	Clock hệ thống
rst_n	In	1 bit	Reset bất đồng bộ, mức thấp
Psel	In	1 bit	Chọn thiết bị
Pwrite	In	1 bit	1: ghi, 0: đọc
Penable	In	1 bit	Bắt đầu truyền dữ liệu
Paddr	In	12 bit	Địa chỉ thanh ghi
Pwdata	In	32 bit	Dữ liệu cần ghi
Pready	Out	1 bit	Luôn ở mức 1 (không có wait-state)
wr_en	Out	1 bit	Tín hiệu cho phép ghi dữ liệu
rd_en	Out	1 bit	Tín hiệu cho phép đọc dữ liệu

Tín hiệu kết nối với khối khác:

Tín hiệu	Tới/Từ khối

clk, rst_n	Từ timer_top
psel...	Từ timer_top (APB bus)
wr_en, rd_en	Tới register_file

Nguyên lý hoạt động:

Khi psel = 1, penable = 1 và pwrite = 1 → wr_en = 1.
Khi psel = 1, penable = 1 và pwrite = 0 → rd_en = 1.
pready luôn bằng 1 vì phiên bản standard không có wait-state.

Module register_file

Khối register_file chứa các thanh ghi điều khiển: TCR, TDR0/1, TCMP0/1, TIER, TISR. Nó nhận tín hiệu đọc/ghi từ APB và trả dữ liệu về hoặc cập nhật các giá trị thanh ghi.

Khối register_file chứa các thanh ghi điều khiển của Timer IP như: TCR, TDR0/1, TCMP0/1, TIER, và TISR. Khối này cho phép ghi hoặc đọc giá trị từ phần mềm thông qua giao tiếp APB, đồng thời tạo ra các tín hiệu điều khiển nội bộ như timer_en, div_en, div_val, và tín hiệu tạo ngắt.

Trong đó:

TDR0 và TDR1 dùng để lưu giá trị hiện tại của bộ đếm 64-bit, có thể ghi hoặc đọc từ phần mềm. Tuy nhiên trong standard level, counter không tự nạp lại từ các thanh ghi này – việc khởi tạo giá trị bộ đếm nếu cần phải do phần mềm thực hiện thông qua reset.

TCMP0 và TCMP1 là các thanh ghi so sánh, chứa giá trị mà counter sẽ so sánh để tạo tín hiệu ngắt. Khi {TDR1, TDR0} == {TCMP1, TCMP0}, tín hiệu int_st_set được bật, và nếu int_en = 1, hệ thống sẽ sinh ra tín hiệu ngắt tim_int.

Tín hiệu I/O:

Tín hiệu	I/O	Độ rộng	Mô tả
Clk	In	1 bit	Clock hệ thống
rst_n	In	1 bit	Reset bất đồng bộ
Addr	In	12 bit	Địa chỉ thanh ghi
Wdata	In	32 bit	Dữ liệu ghi
wr_en	In	1 bit	Cho phép ghi
rd_en	In	1 bit	Cho phép đọc
Rdata	Out	32 bit	Dữ liệu đọc trả về

cnt_value	In	64 bit	Giá trị đếm từ counter
div_en	Out	1 bit	Cho phép chia tần số
div_val	Out	4 bit	Giá trị chia
timer_en	Out	1 bit	Bật/tắt counter
TDR0	Out	32 bit	Giá trị counter ban đầu phần thấp
TDR1	Out	32 bit	Giá trị counter ban đầu phần cao
TCMP0	Out	32 bit	Giá trị so sánh phần thấp
TCMP1	Out	32 bit	Giá trị so sánh phần cao
int_en	Out	1 bit	Cho phép ngắt
int_st	Out	1 bit	Trạng thái ngắt
int_st_set	Out	1 bit	Bật trạng thái ngắt khi counter = compare
int_st_clear	Out	1 bit	Xóa trạng thái ngắt khi ghi vào TISR

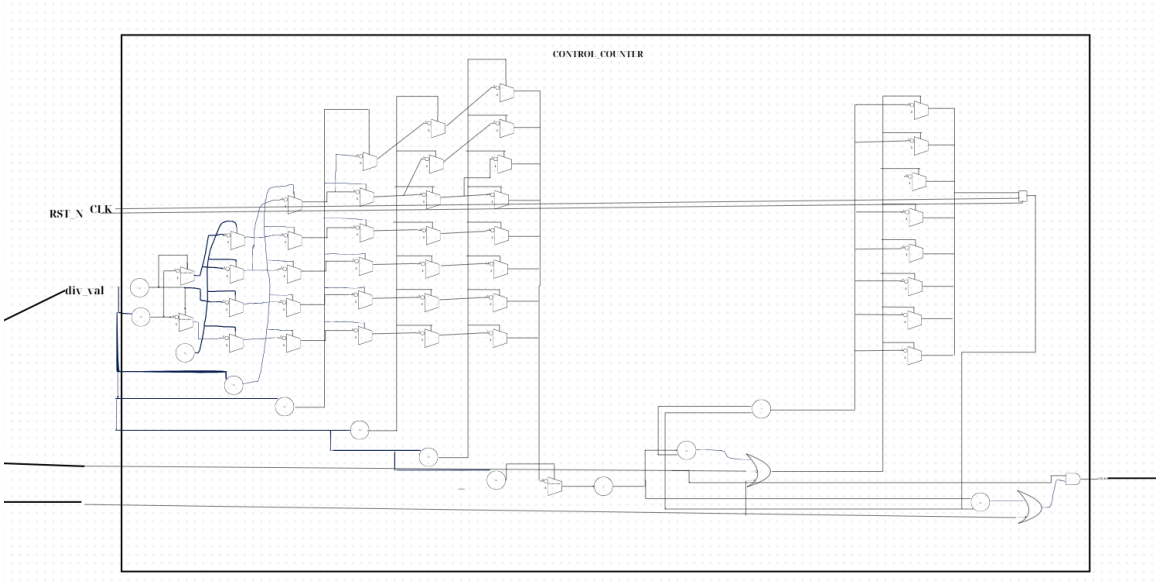
Tín hiệu kết nối với khối khác:

Tín hiệu	Tới/Từ khối
wr_en, rd_en	Từ apb_slave
cnt_value	Từ counter
Rdata	Tới timer_top
TDR0/1, TCMP0/1	Tới counter
int_st_set/clear	Tới interrupt

Nguyên lý hoạt động:

Khi wr_en = 1, ghi giá trị vào thanh ghi tương ứng theo địa chỉ.
Khi rd_en = 1, đọc giá trị của thanh ghi tại địa chỉ tương ứng.
Khi cnt_value == {TCMP1, TCMP0} → tạo int_st_set = 1.
Khi ghi 1 vào bit 0 của TISR → tạo int_st_clear = 1.

Module control_counter



Khối control_counter điều khiển tín hiệu cnt_en dựa vào cấu hình chia tần số (div_val, div_en) và trạng thái bật timer (timer_en).

Tín hiệu I/O:

Tín hiệu	I/O	Độ rộng	Mô tả
Clk	In	1 bit	Clock hệ thống
rst_n	In	1 bit	Reset bất đồng bộ
div_en	In	1 bit	Cho phép chia
div_val	In	4 bit	Giá trị chia (1 đến 256)
timer_en	In	1 bit	Bật counter
cnt_en	Out	1 bit	Cho phép counter tăng

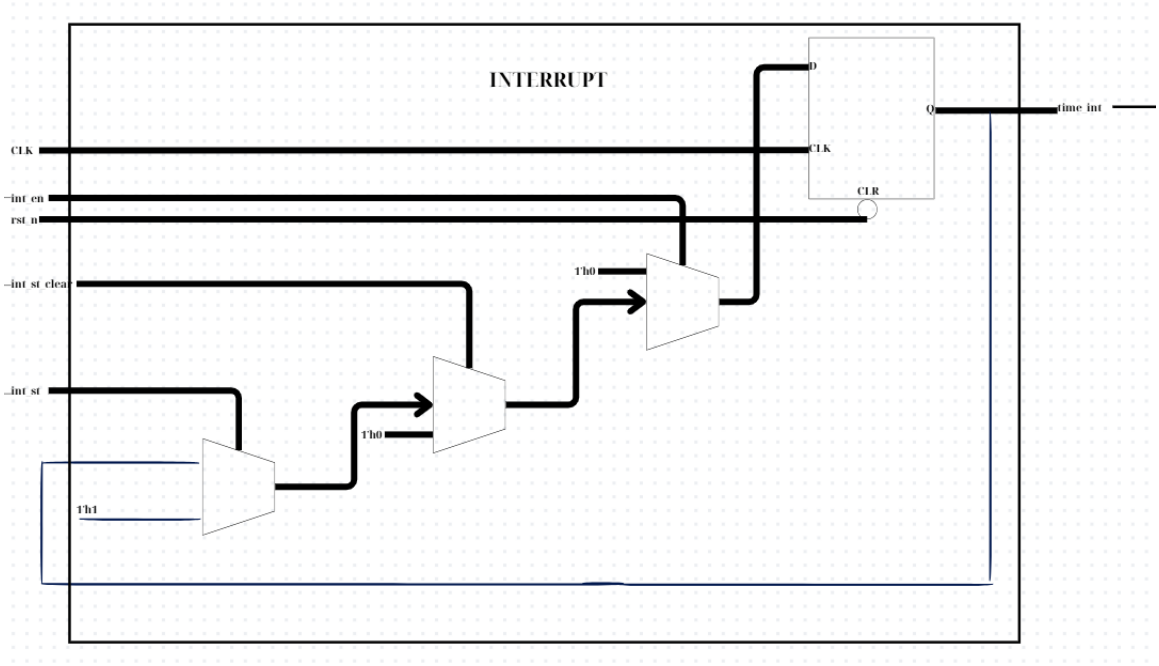
Tín hiệu kết nối với khối khác:

Tín hiệu	Tới/Từ khối
div_en, div_val	Từ register_file
timer_en	Từ register_file
cnt_en	Tới counter

Nguyên lý hoạt động:

Nếu timer_en = 0 → cnt_en = 0.
Nếu div_en = 0 → cnt_en = 1 mỗi chu kỳ.
Nếu div_en = 1 → cnt_en = 1 mỗi khi bộ đếm nội bộ đạt div_val.

Module interrupt



Khối interrupt tạo tín hiệu ngắt tim_int khi có điều kiện phù hợp.

Tín hiệu I/O:

Tín hiệu	I/O	Độ rộng	Mô tả
Clk	In	1 bit	Clock hệ thống
rst_n	In	1 bit	Reset bất đồng bộ
int_st_set	In	1 bit	Bật trạng thái ngắt (khi counter = compare)
int_st_clear	In	1 bit	Xóa ngắt (ghi vào TISR)
int_en	In	1 bit	Cho phép ngắt
int_st	In	1 bit	Trạng thái pending
tim_int	Out	1 bit	Tín hiệu ngắt ra ngoài

Tín hiệu kết nối với khối khác:

Tín hiệu	Tới/Từ khối
int_en/st/set/clear	Từ register_file
tim_int	Tới timer_top (xuất ra)

Nguyên lý hoạt động:

Nếu rst_n = 0 → tim_int = 0.
Nếu int_en = 1 và int_st = 1 → tim_int = 1.
Nếu int_en = 0 hoặc int_st_clear = 1 → tim_int = 0.

II. CODE RTL:

```
duy_ic22@ictc-eda-ldap:~/16_ss16/rtl$ pwd
/ictc/student_data/duy_ic22/16_ss16/rtl
```

```
module counter(
        input wire clk,
        input wire rst_n,
        input wire cnt_en,
        input wire [31:0] tdr0,
        input wire [31:0] tdr1,
        output reg [63:0] cnt_value
);

always @(posedge clk or negedge rst_n) begin
        if(!rst_n) begin
                cnt_value <= 64'd0;
        end
        else if (cnt_en) begin
                cnt_value <= cnt_value + 1;
        end
end
endmodule
```

```
module control_counter(
        input wire clk,
        input wire rst_n,
        input wire div_en,
        input wire [3:0] div_val,
        input wire timer_en,
        output wire cnt_en
);

reg [7:0] int_cnt;
wire [7:0] div_factor;

assign div_factor = (div_val == 4'd0) ? 8'd1 :
                    (div_val == 4'd1) ? 8'd2 :
                    (div_val == 4'd2) ? 8'd4 :
                    (div_val == 4'd3) ? 8'd8 :
                    (div_val == 4'd4) ? 8'd16:
                    (div_val == 4'd5) ? 8'd32:
                    (div_val == 4'd6) ? 8'd64:
                    (div_val == 4'd7) ? 8'd128 : 8'd255;
assign cnt_en = (timer_en && (!div_en || (int_cnt == div_factor - 1)));
always @(posedge clk or negedge rst_n) begin
        if(!rst_n || !timer_en || !div_en || (int_cnt == div_factor - 1))
                int_cnt <= 0;
        else if (div_en && timer_en)
                int_cnt <= int_cnt + 1;
end
endmodule
```

```
module register(
        input wire clk,
        input wire rst_n,
        input wire [11:0] addr,
        input wire [31:0] wdata,
        input wire wr_en,
        input wire rd_en,
        output reg [31:0] rdata,
        input wire [63:0] cnt_value,
        output reg div_en,
        output reg timer_en,
        output reg [3:0] div_val,
        output reg [31:0] TDR0,
        output reg [31:0] TDR1,
        output reg [31:0] TCMP0,
        output reg [31:0] TCMP1,
        output reg int_st,
        output reg int_en,
        output wire int_st_set,
        output wire int_st_clear
);

parameter TCR_ADDR = 12'h000;
parameter TDR0_ADDR = 12'h004;
parameter TDR1_ADDR = 12'h008;
parameter TCMP0_ADDR = 12'h00C;
parameter TCMP1_ADDR = 12'h010;
parameter TIER_ADDR = 12'h014;
parameter TISR_ADDR = 12'h018;
```

```
assign int_st_set = ({TDR1, TDR0} == {TCMP1, TCMP0});
assign int_st_clear = (addr == TISR_ADDR && wr_en && wdata[0]);

always @(posedge clk or negedge rst_n) begin
        if(!rst_n) begin
                div_en <= 0;
                timer_en <= 0;
                div_val <= 4'd1;
                TDR0 <= 32'd0;
                TDR1 <= 32'd0;
                TCMP0 <= 32'hFFFF_FFFF;
                TCMP1 <= 32'hFFFF_FFFF;
                int_en <= 0;
                int_st <= 0; end
        else begin
                if(wr_en) begin
                        case(addr)
                                TCR_ADDR: begin
                                        timer_en <= wdata[0];
                                        div_en <= wdata[1];
                                        div_val <= wdata [11:8]; end
                                TDR0_ADDR: TDR0 <= wdata;
                                TDR1_ADDR: TDR1 <= wdata;
                                TCMP0_ADDR: TCMP0 <= wdata;
                                TCMP1_ADDR: TCMP1 <= wdata;
                                TIER_ADDR: int_en <= wdata[0];
                                TISR_ADDR: begin
                                        if (wdata[0])
```

```
                                TISR_ADDR: begin
                                        if (wdata[0])
                                                int_st <= 1'b0; end
                                endcase
                        end
                        if(int_st_set)
                                int_st <= 1;
                end
        end

        always @(*) begin
                if(rd_en) begin
                        case(addr)
                                TCR_ADDR: rdata = {20'b0,div_val,6'b0,div_en,timer_en};
                                TDR0_ADDR: rdata = TDR0;
                                TDR1_ADDR: rdata = TDR1;
                                TCMP0_ADDR: rdata = TCMP0;
                                TCMP1_ADDR: rdata = TCMP1;
                                TIER_ADDR: rdata = {31'b0, int_en};
                                TISR_ADDR: rdata = {31'b0,int_st};
                                default: rdata = 32'b0;
                        endcase
                end
                else begin
                        rdata = 32'b0;
                end
        end
        endmodule
```

```
module apb(
        input wire clk,
        input wire rst_n,
        input wire psel,
        input wire pwrite,
        input wire penable,
        input wire [11:0] paddr,
        input wire [31:0] pwdata,
        output wire pready,
        output reg wr_en,
        output reg rd_en
);

assign pready = psel;

always @(*) begin
        wr_en = (psel && penable && pwrite);
        rd_en = (psel && penable && !pwrite);
end
endmodule
```

```
module interrupt(
        input wire clk,
        input wire rst_n,
        input wire int_st_set,
        input wire int_st_clear,
        input wire int_en,
        input wire int_st,
        output reg tim_int
);

always @(posedge clk or negedge rst_n)begin
        if(!rst_n)
                tim_int <= 0;
        else if (!int_en)
                tim_int <= 0;
        else if (int_st_clear)
                tim_int <= 0;
        else if (int_st)
                tim_int <= 1;
end
endmodule
```

```
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, rd_en;
wire [63:0] cnt_value;
wire div_en, timer_en;
wire [3:0] div_val;
wire cnt_en;
wire [31:0] tdr0, tdr1;
wire [31:0] tcmp0, tcmp1;
wire int_st,int_en,int_st_set,int_st_clear;

assign tim_pslverr = 1'b0;
```

```
apb U0 (
          .clk(sys_clk),
          .rst_n(sys_rst_n),
          .psel(tim_psel),
          .pwrite(tim_pwrite),
          .penable(tim_penable),
          .paddr(tim_paddr),
          .pwdata(tim_pwdata),
          .pready(tim_pready),
          .wr_en(wr_en),
          .rd_en(rd_en)
);
```

```
register U1 (
        .clk(sys_clk),
        .rst_n(sys_rst_n),
        .addr (tim_paddr),
        .wdata (tim_pwdata),
        .wr_en(wr_en),
        .rd_en(rd_en),
        .rdata(tim_prdata),
        .cnt_value(cnt_value),
        .div_en(div_en),
        .div_val(div_val),
        .timer_en(timer_en),
        .TDR0(tdr0),
        .TDR1(tdr1),
        .TCMP0(tcmp0),
        .TCMP1(tcmp1),
        .int_st(int_st),
        .int_en(int_en),
        .int_st_clear(int_st_clear)
);
```

```
control_counter U2 (
        .clk(sys_clk),
        .rst_n(sys_rst_n),
        .div_en(div_en),
        .div_val(div_val),
        .timer_en(timer_en),
        .cnt_en(cnt_en)
);

counter U3 (
        .clk(sys_clk),
        .rst_n(sys_rst_n),
        .cnt_en(cnt_en),
        .tdr0(tdr0),
        .tdr1(tdr1),
        .cnt_value(cnt_value)
);

interrupt U4 (
        .clk(sys_clk),
        .rst_n(sys_rst_n),
        .int_st_set(int_st_set),
        .int_st_clear(int_st_clear),
        .int_en(int_en),
        .int_st(int_st),
        .tim_int(tim_int)
);
endmodule
```

III. **TEST BENCH:**

```
duy_ic22@ictc-eda-ldap:~/16_ss16/tb$ pwd
/ictc/student_data/duy_ic22/16_ss16/tb
```

VPLAN:

TC ID	Mô tả	Địa chỉ	Ghi giá trị	Đọc giá trị	Giá trị kỳ vọng
TC01	Ghi TCR để bật timer	0x000	0x00000001	0x000	timer_en = 1
TC02	Ghi TCMP0 và TCMP1	0x00C/0x010	0x12345678 / 0x9ABCDEF0	0x00C/0x010	Giá trị đã ghi
TC03	Ghi TDR0 và TDR1	0x004/0x008	0x89ABCDEF / 0x01234567	0x004/0x008	Giá trị đã ghi
TC04	Ghi TIER để bật int_en	0x014	0x00000001	0x014	int_en = 1
TC05	Ghi TISR để xóa int_st	0x018	0x00000001	0x018	int_st = 0
TC0 6	Ghi THCSR để bật halt_req	0x01C	0x00000001	0x01C	halt_req expected 1
TC0 7	Ghi TIER để tắt int_en	0x014	0x00000000	0x014	nt_en expected 0
TC0 8	Ghi TISR với wdata[0	0x018	0x00000000	0x018	int_st should remain 1

	] = 0				
TC0 9	Ghi TCR với div_val > 8	0x000	0x00000F03	0x000	div_val out of range
TC0 10	Ghi địa chỉ không hợp lệ	0x0FFF	DEADBEEF	0x0FFF	default rdata expected 0

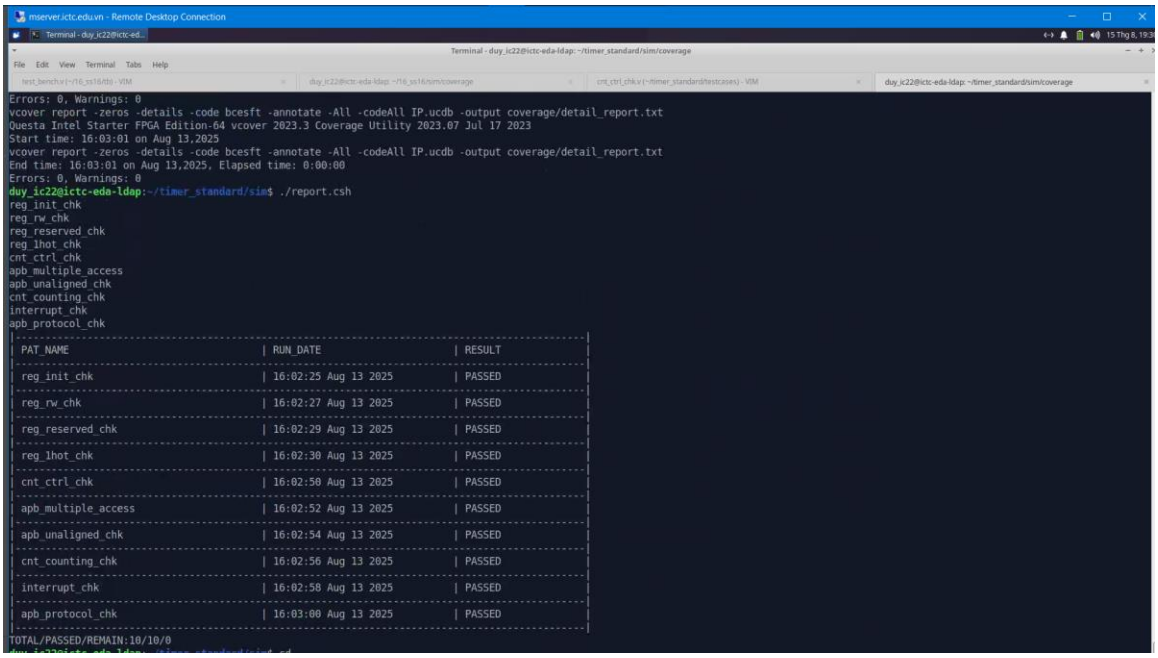
COVERAGE:

```
duy_ic22@ictc-eda-ldap:~/16_ss16/sim/coverage$ cat summary_report.txt
Coverage Report Summary Data by instance

================================================================================
=== Instance: /test_bench/uut/U0
=== Design Unit: work.apb
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Expressions                      6         6         0   100.00%
    Statements                       2         2         0   100.00%

================================================================================
=== Instance: /test_bench/uut/U1
=== Design Unit: work.register
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        30        30         0   100.00%
    Conditions                       1         1         0   100.00%
    Expressions                      4         4         0   100.00%
    Statements                      39        39         0   100.00%
    Toggles                        338       338         0   100.00%

================================================================================
=== Instance: /test_bench/uut/U2
=== Design Unit: work.control_counter
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        25        24         1    96.00%
    Conditions                      12        11         1    91.66%
    Statements                       6         6         0   100.00%
    Toggles                         36        36         0   100.00%

================================================================================
```

```
================================================================================
=== Instance: /test_bench/uut/U3
=== Design Unit: work.counter
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         7         6         1    85.71%
    Conditions                       5         4         1    80.00%
    Statements                       5         5         0   100.00%
    Toggles                        128       128         0   100.00%

================================================================================
=== Instance: /test_bench/uut/U4
=== Design Unit: work.interrupt
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Expressions                      2         2         0   100.00%
    Statements                       1         1         0   100.00%

================================================================================
=== Instance: /test_bench/uut
=== Design Unit: work.timer_top
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                        498       498         0   100.00%

================================================================================
=== Instance: /test_bench
=== Design Unit: work.test_bench
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                     540       540         0   100.00%
    Toggles                        174       163        11    93.67%


Total Coverage By Instance (filtered view): 96.94%
```

IV.GOLDEN MODEL:

```
duy_ic22@ictc-eda-ldap:~/timer_standard/sim/coverage$ pwd
/ictc/student_data/duy_ic22/timer_standard/sim/coverage
```


mserver.ictc.edu.vn - Remote Desktop Connection
Terminal - duy_ic22@ictc-ed...
15 Thg 8, 19:3
Terminal - duy_ic22@ictc-eda-ldap: ~/timer_standard/sim/coverage
File Edit View Terminal Tabs Help
test_bench.v (~/16_ss16/tb) - VIM
duy_ic22@ictc-eda-ldap: ~/16_ss16/sim/coverage
cnt_ctrl_chk.v (~/timer_standard/testcases) - VIM
duy_ic22@ictc-eda-ldap: ~/timer_standard/sim/coverage
Errors: 0, Warnings: 0
vcover report -zeros -details -code bcesft -annotate -All -codeAll IP.ucdb -output coverage/detail_report.txt
Questa Intel Starter FPGA Edition-64 vcover 2023.3 Coverage Utility 2023.07 Jul 17 2023
Start time: 16:03:01 on Aug 13,2025
vcover report -zeros -details -code bcesft -annotate -All -codeAll IP.ucdb -output coverage/detail_report.txt
End time: 16:03:01 on Aug 13,2025, Elapsed time: 0:00:00
Errors: 0, Warnings: 0
duy_ic22@ictc-eda-ldap:~/timer_standard/sim$ ./report.csh
reg_init_chk
reg_rw_chk
reg_reserved_chk
reg_1hot_chk
cnt_ctrl_chk
apb_multiple_access
apb_unaligned_chk
cnt_counting_chk
interrupt_chk
apb_protocol_chk
PAT_NAME	RUN_DATE	RESULT
reg_init_chk	16:02:25 Aug 13 2025	PASSED
reg_rw_chk	16:02:27 Aug 13 2025	PASSED
reg_reserved_chk	16:02:29 Aug 13 2025	PASSED
reg_1hot_chk	16:02:30 Aug 13 2025	PASSED
cnt_ctrl_chk	16:02:50 Aug 13 2025	PASSED
apb_multiple_access	16:02:52 Aug 13 2025	PASSED
apb_unaligned_chk	16:02:54 Aug 13 2025	PASSED
cnt_counting_chk	16:02:56 Aug 13 2025	PASSED
interrupt_chk	16:02:58 Aug 13 2025	PASSED
apb_protocol_chk	16:03:00 Aug 13 2025	PASSED
TOTAL/PASSED/REMAIN:10/10/0