IC TRAINING CENTER VIET NAM
FUNDAMENTAL IC DESIGN AND VERIFICATION COURSE

FINAL PROJECT
TIMER IP DESIGN SPECIFICATION

Student: Doan Le Nhan
Class: IC28
Instructor: Kha Kham

Contents

1.Overview 4
1.1. Introduction 4
1.2. Main features 4
1.3. Design Proposal 9
1.3. Block diagram 12
1.4. Interface signals 22
2.Register Specification 26
2.1. Register Summary 26
2.2. Timer Control Register (TCR) 26
2.3. Timer Data Register 0 (TDR0) 27
2.4. Timer Data Register 1 (TDR1) 28
2.5. Timer Compare Register 0 (TCMP0) 28
2.6. Timer Compare Register 1 (TCMP1) 29
2.7. Timer Interrupt Enable Register (TIER) 29
2.8. Timer Interrupt Status Register (TISR) 30
2.9. Timer Halt Control Status Register (THCSR) 31
3. Functional Description 32
3.1. APB slave – Register - Counter 32
3.2. Counter Control 33
3.3. Interrupt 34
4. Sample Waveform 35
5. VPLAN 36
6. URL FILE CODE 38
RTL CODE 38
TESTCASE 38
7. RESULT 39
RESULT WITH TESTBENCH + TESTCASES 39
RESULT WITH GOLDEN MODEL 40
COVERAGE 41

1.Overview

1.1. Introduction

The Timer Counter IP (timer_top) is a configurable 64-bit timer supporting normal counting, divided counting, compare-match interrupt generation, and debug-halt.
Software controls the timer through APB registers, while the hardware provides stable timing and interrupt capability.

Applications include:
• Periodic timer events
• Timeout detection
• Event scheduling
• Real-time counters
• Heartbeat generation
• Low-frequency divided counters
• Debug halt support

1.2. Main features

The Timer Counter IP provides the following key features:

General

- 64-bit up-counter with continuous increment behavior.
- APB slave interface with 32-bit data width with byte-write support (pstrb[3:0]) and one wait states.

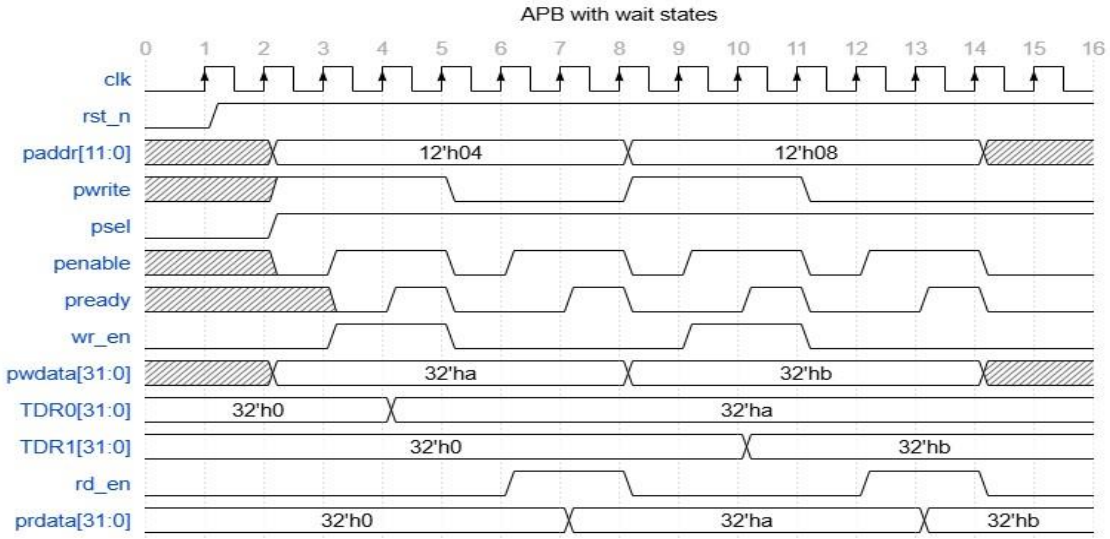


Fig 1. APB with wait states

Counting Modes

- **Normal mode:**
  Counter increments every system clock cycle when TCR.timer_en = 1.
- **Divider mode:**
  When TCR.div_en = 1, the counter increments at a reduced rate determined by TCR.div_val (divide-by 1, 2, 4, 8, …, 256).
- Some example waveform of counter if control mode:

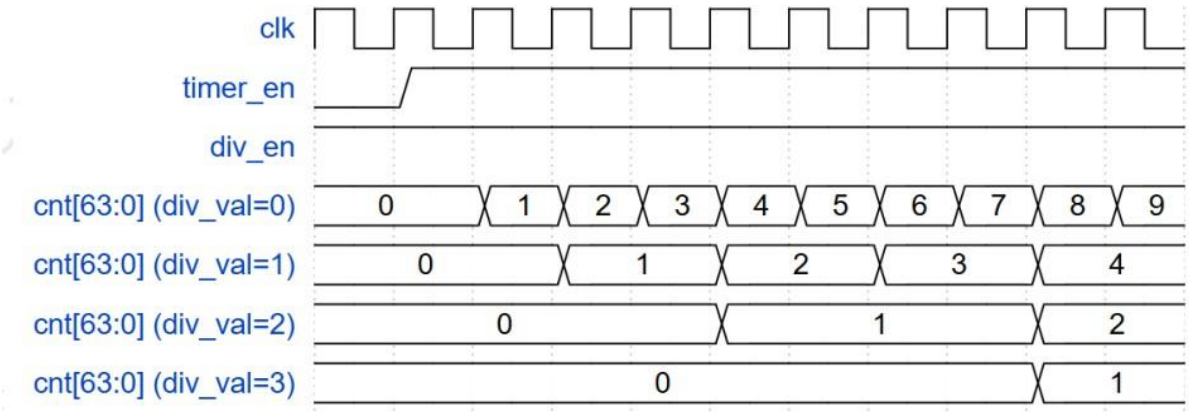


Fig 2. Counter control mode with different div_val

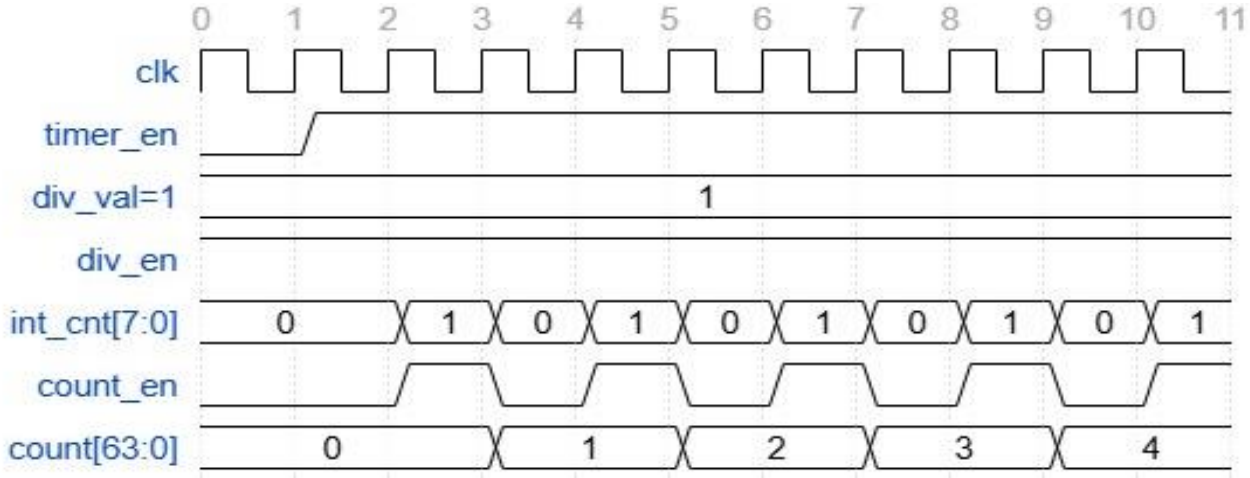


Fig 3. Counter control mode with div_val = 1

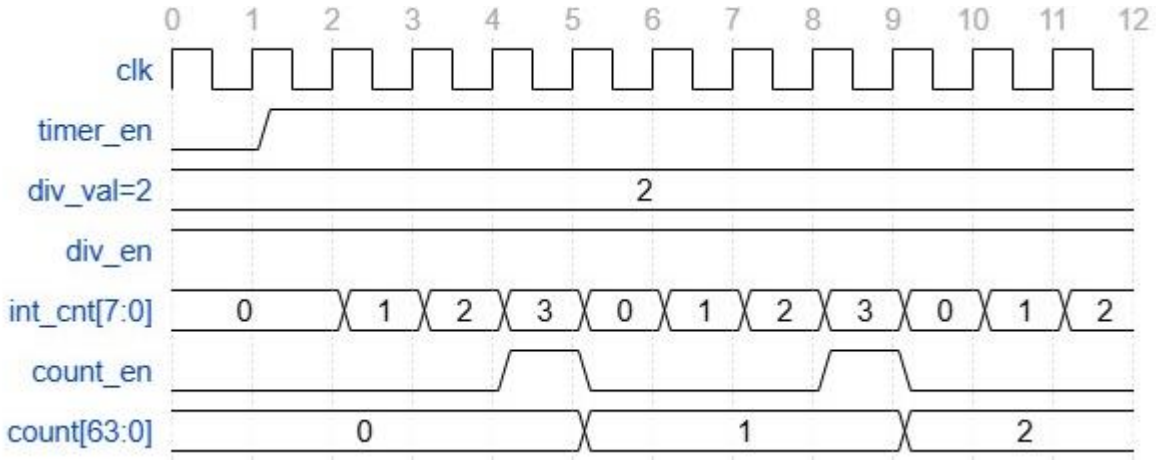


Fig 4. Counter control mode with div_val = 2

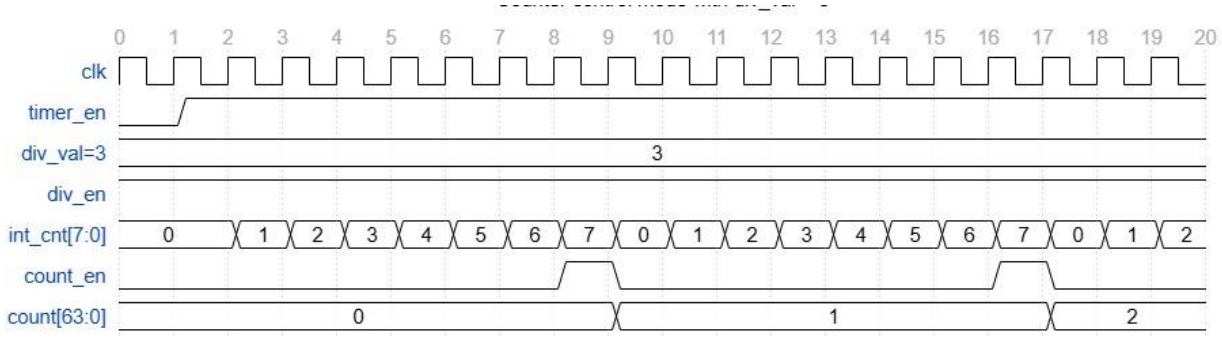


Fig 5. Counter control mode with div_val = 3

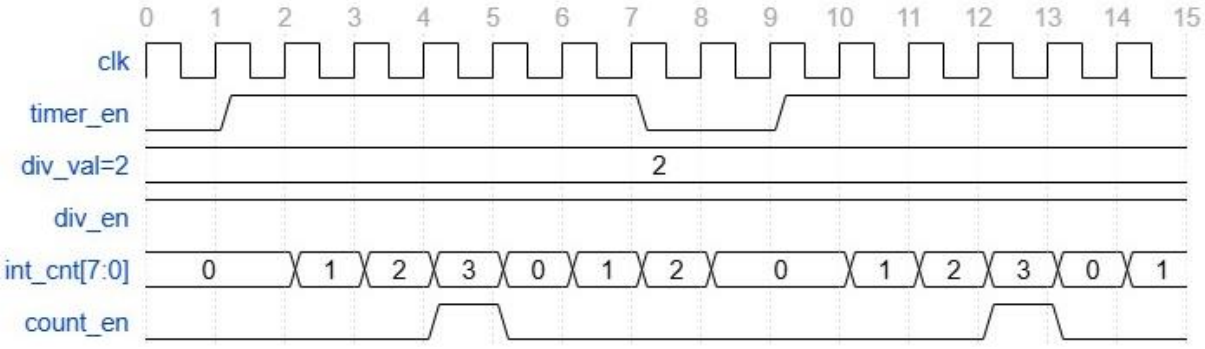


Fig 6. Counter control mode with div_val = 2, timer_en=0 when div_en=1

- **Halt mode (debug freeze):**
  Counter halts only when both dbg_mode = 1 and THCSR.halt_req = 1.
  The counter resumes once halt_req is cleared.

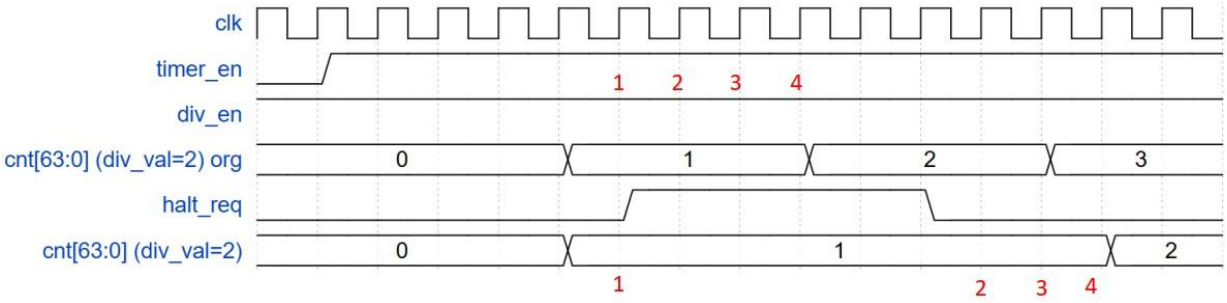


Fig 7. Counter is halted

- **Protection rules:**
  Divider settings (div_en, div_val) cannot be modified while timer is running.
  Invalid divider values (div_val > 8) produce an error response.

Counter Behavior

- *Counter reloads from TDR0/TDR1 when timer is enabled.*
- *Counter clears to zero when timer_en transitions from 1 → 0.*

APB Interface Behavior

- One-cycle wait state inserted automatically (pready = wr_en | rd_en).
- Illegal writes generate APB error (pslverr = 1).
- On error, the write is ignored and the register retains its previous value.
- Supports byte access for partial register update.

Interrupt System

- Interrupt asserted when the counter matches the 64-bit compare value {TCMP1, TCMP0}.
- Interrupt enable via TIER.int_en.
- Interrupt status (TISR.int_st) uses RW1C behavior (write-1-to-clear).
- Disabling int_en masks the output interrupt but preserves the pending flag.

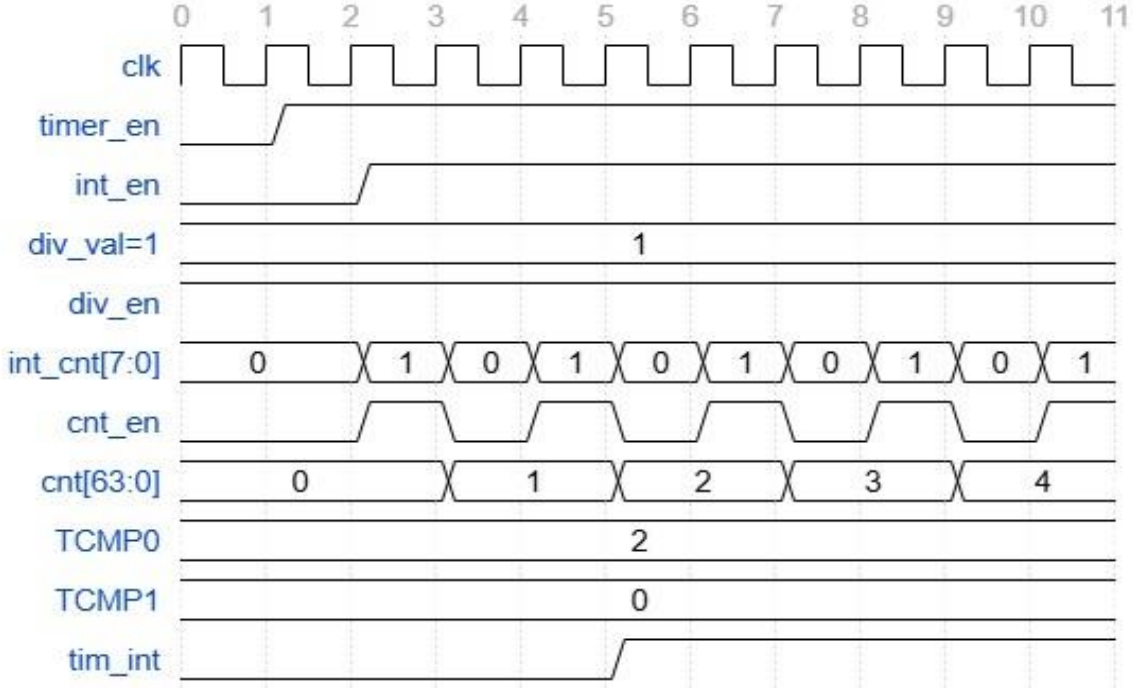


Fig 8. Interrupt assert condition

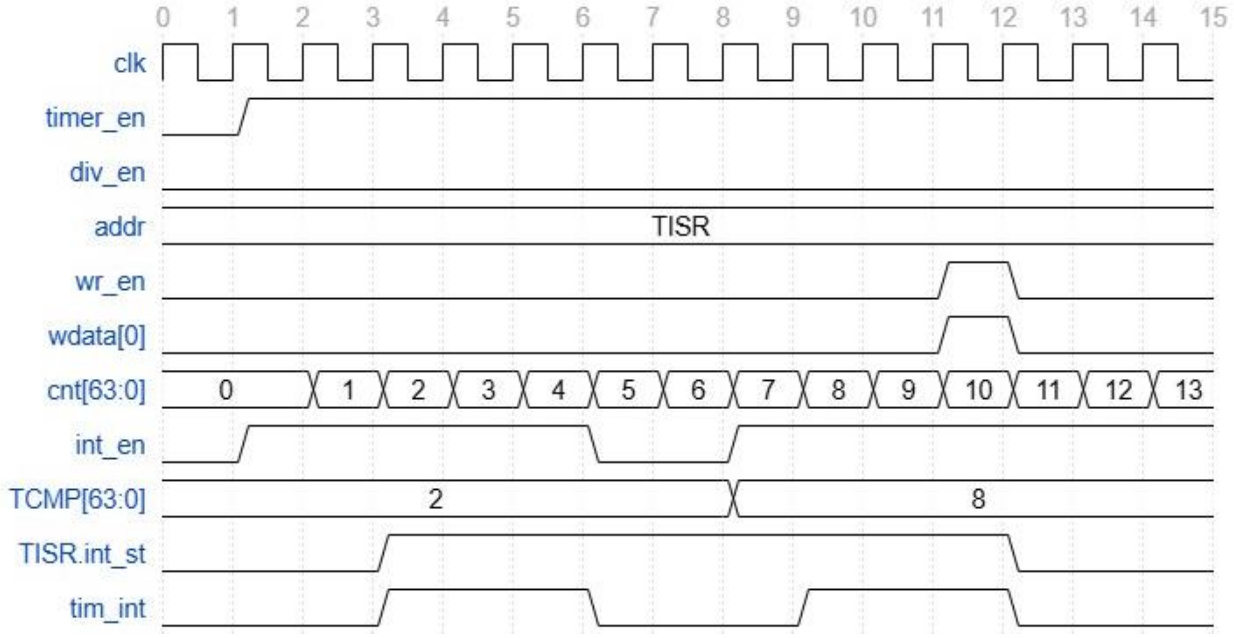


Fig 9. Interrupt clear

1.3. Design Proposal

The Timer IP provides the following features:

APB Slave Interface Behavior

• 32-bit APB slave
• Byte-write via pstrb
• Always inserts 1 wait state (pready = wr_en | rd_en)
• Error detection (pslverr = 1 when):
  – Writing invalid div_val > 8
  – Modifying div_val / div_en while timer_en = 1
• On error → write ignored, registers keep old value

64-bit Counter (CNT64) with Divider Mode

• Up-counter stored in TDR0 (low 32 bits) and TDR1 (high 32 bits)
• Increments when timer_en = 1
• Normal mode: increment every clock
• Divider mode: increment every **2^div_val** cycles
• Overflow wraps to 0
• timer_en: 1→0 → counter cleared
• timer_en: 0→1 → counter reloads from TDR0/TDR1 *Divider Mode (div_en & div_val)* when timer_en = 1 & div_en = 1:

Table 1. Divider Mode of Counter

div_val	Divide Factor
4'b0000	1 (no division)
4'b0001	2
4'b0010	4
4'b0011	8
4'b0100	16
4'b0101	32

div_val	Divide Factor
4'b0110	64
4'b0111	128
4'b1000	256

Reserved values (>8) → PSLVERR.

Protection:

• *div_en and div_val cannot change while running (timer_en=1)*

• *Illegal writes → PSLVERR + write ignored*

Interrupt Mechanism

• *int_en – interrupt enable*

• *int_st – compare-match status*

• *tim_int – final output interrupt*

Trigger:

• *cnt_64b == cmp_64b → int_st = 1*

• *If int_en = 1 → tim_int = 1*

Clear:

• *RW1C: write 1 → clear int_st*

• *Writing 0 → ignored*

• *Clearing int_en masks tim_int*

Debug Halt

Halt occurs only if:

1. dbg_mode = 1
2. halt_req = 1

While halted:
• Counter frozen
• halt_ack = 1

When halted released:
• halt_ack = 0
• Counter resumes exactly

If dbg_mode = 0:
• halt_req ignored
• No halting

1.3. Block diagram

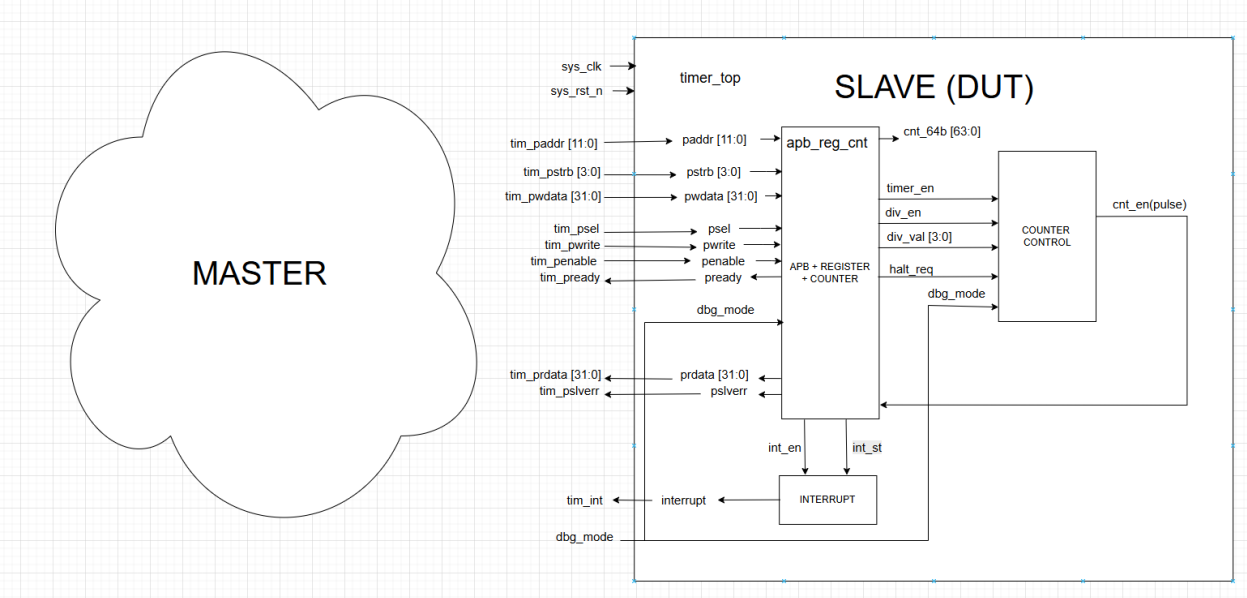


Fig 10. Block diagram of the DUT

• *APB Interface/ Register/Counter Block: APB interface and TCR, TDR0/1, TCMP0/1, TIER, TISR, THCSR, 64-bit counter.*
• *Counter Control Block: generates cnt_en based on dividers & halt logic*
• *Interrupt Block*

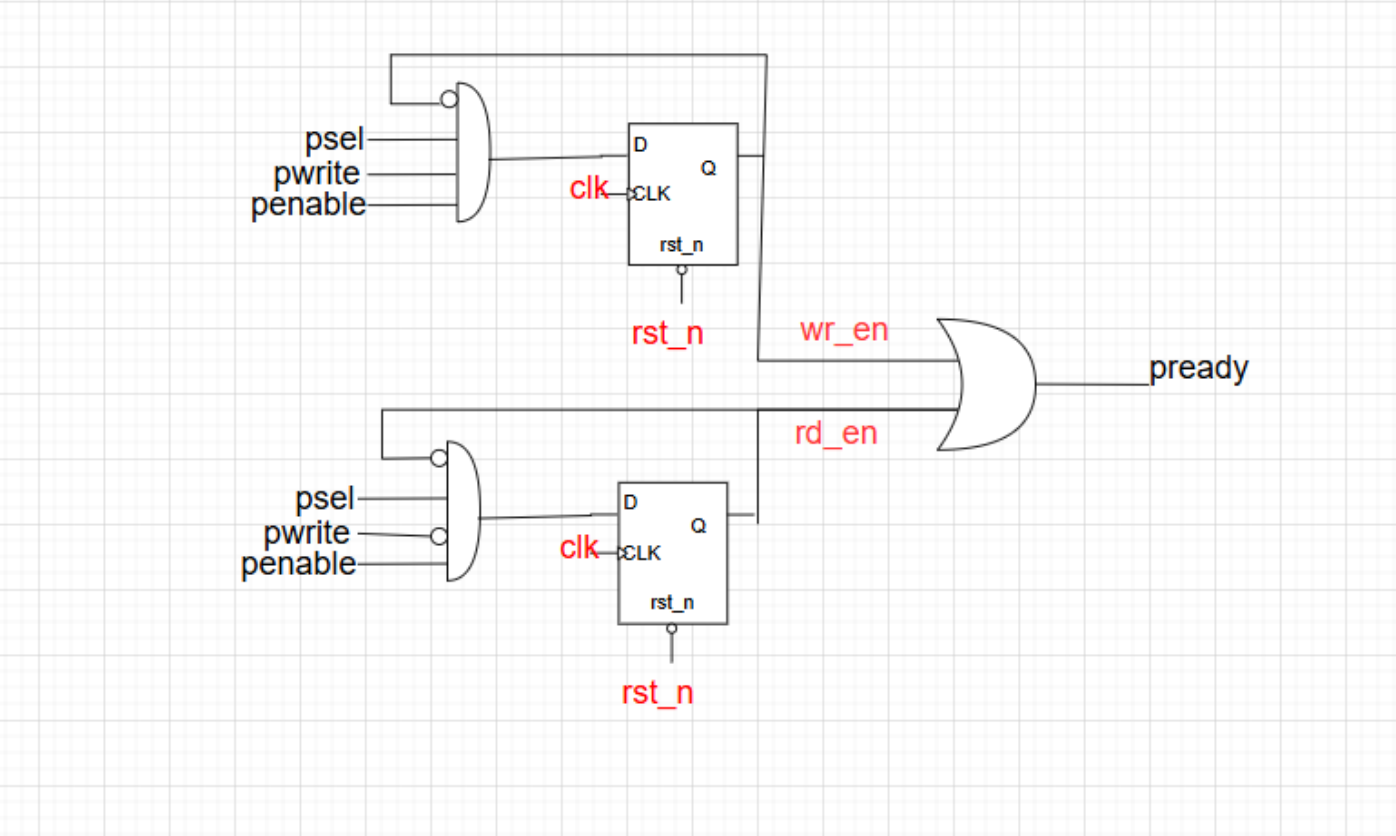


Fig 11. Write/Read Logic and PREADY Logic

- *wr_en and rd_en generated from psel/penable/pwrite*
- *1-cycle pulse for deterministic register access*
- *pready = wr_en | rd_en*

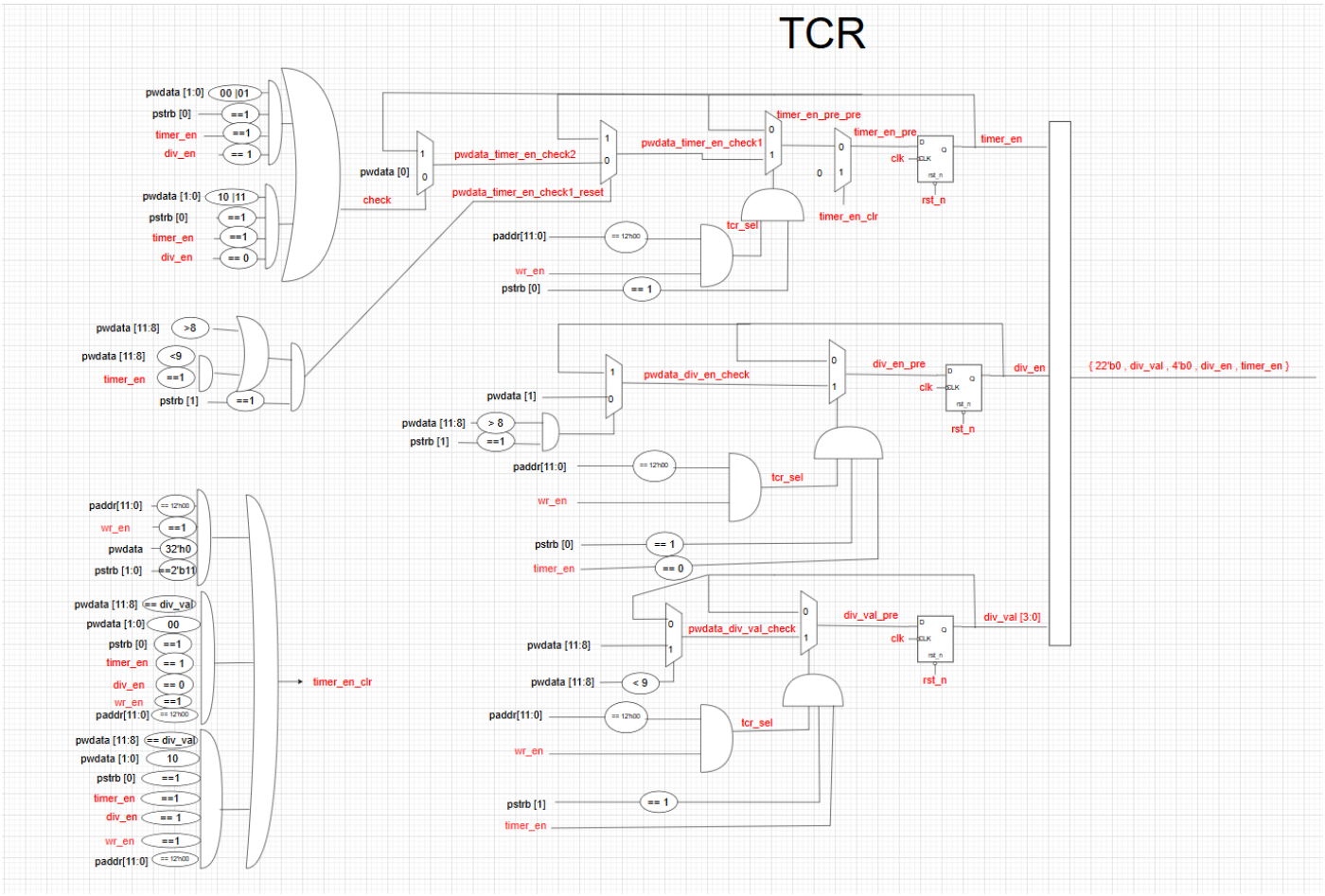


Fig 12. – Timer Control Register (TCR) Logic

- *Handles timer_en, div_en, div_val*
- *Protects divider values when running*
- *Invalid writes flagged by PSLVERR*

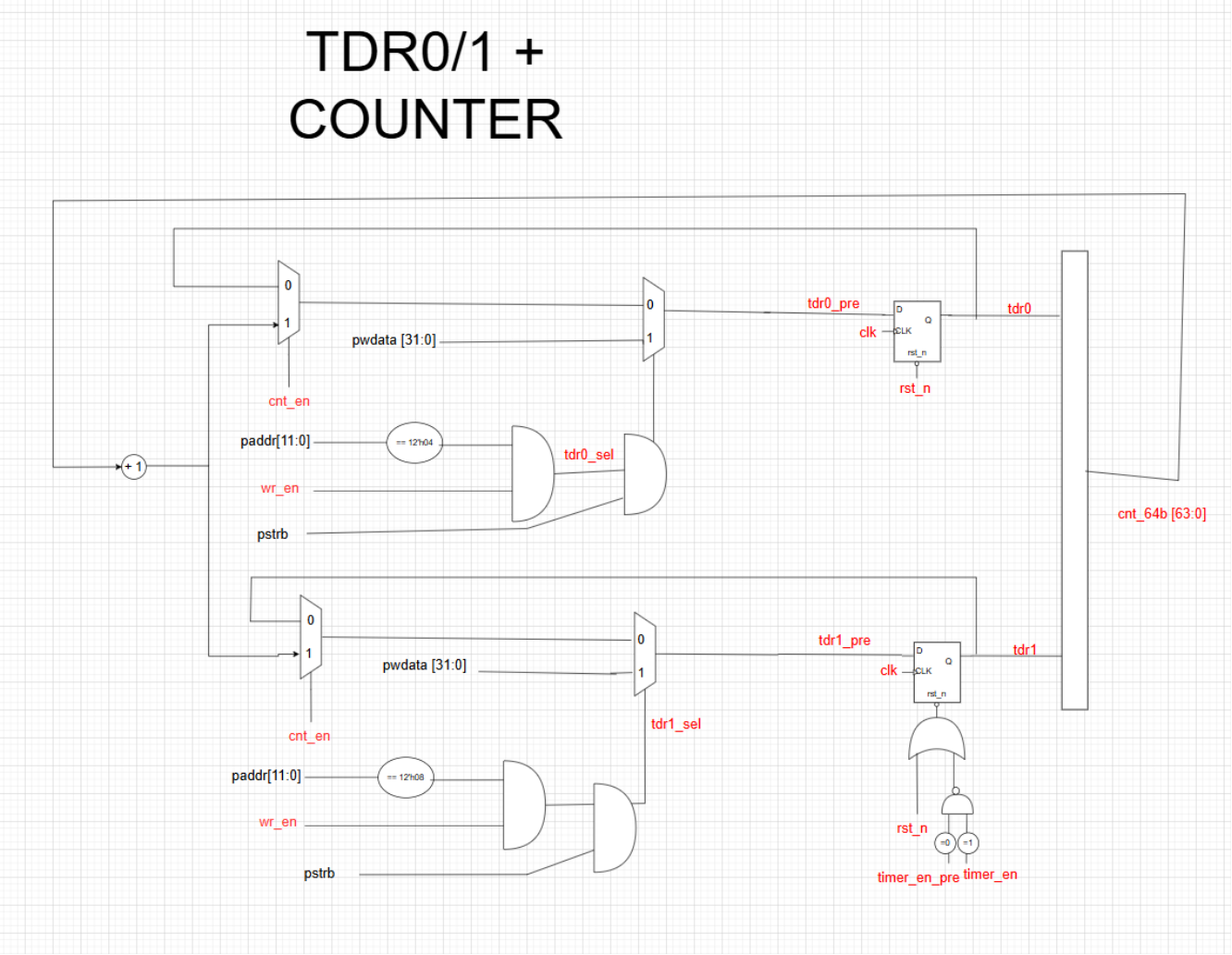


Fig 13. TDR0/TDR1 and 64-bit Counter

- *TDRs mirror CNT64 for readback*
- *Counter reload on timer_en*
- *Counter reset on timer_en high to low*

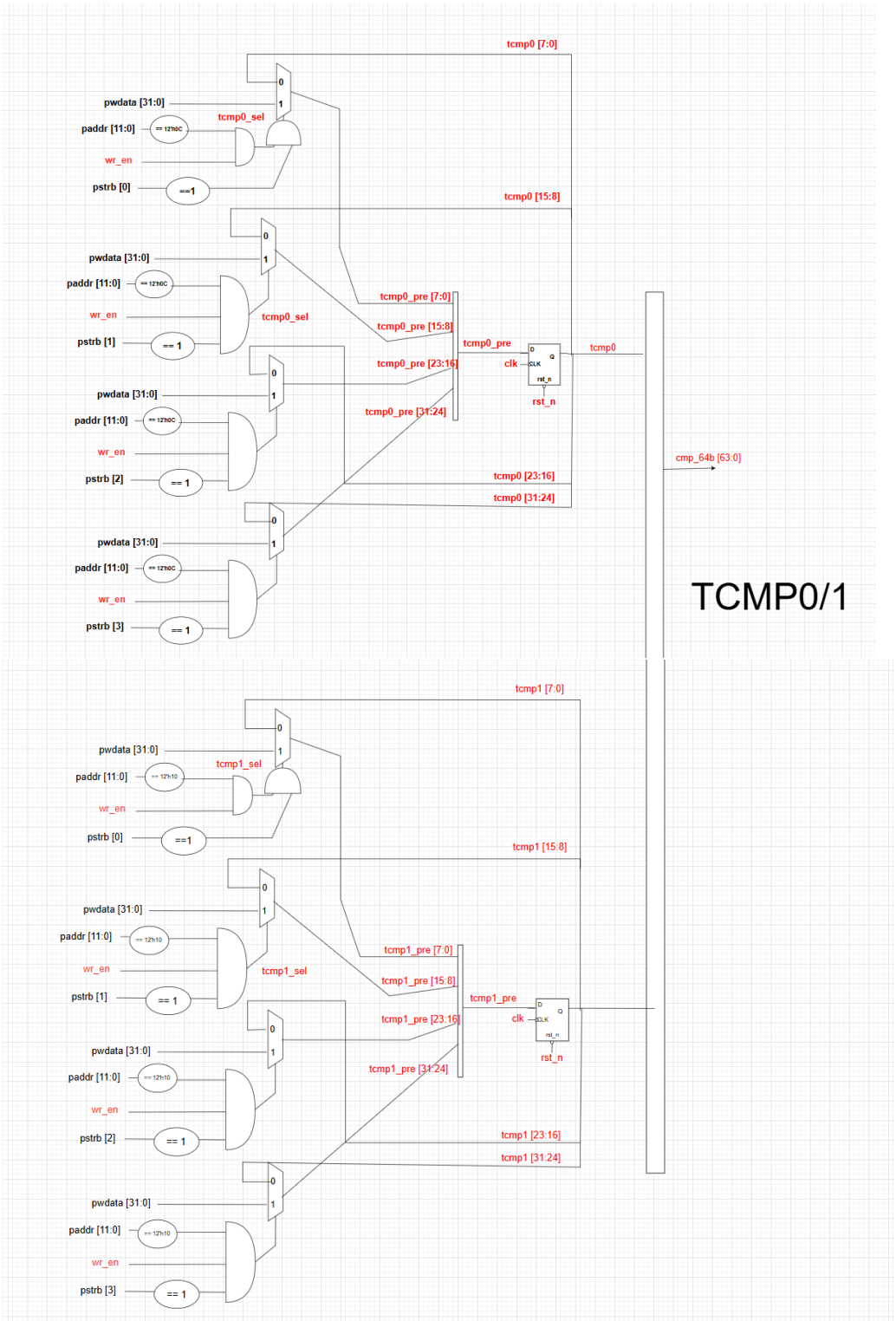


Fig 14. Compare Registers (TCMP0/TCMP1)

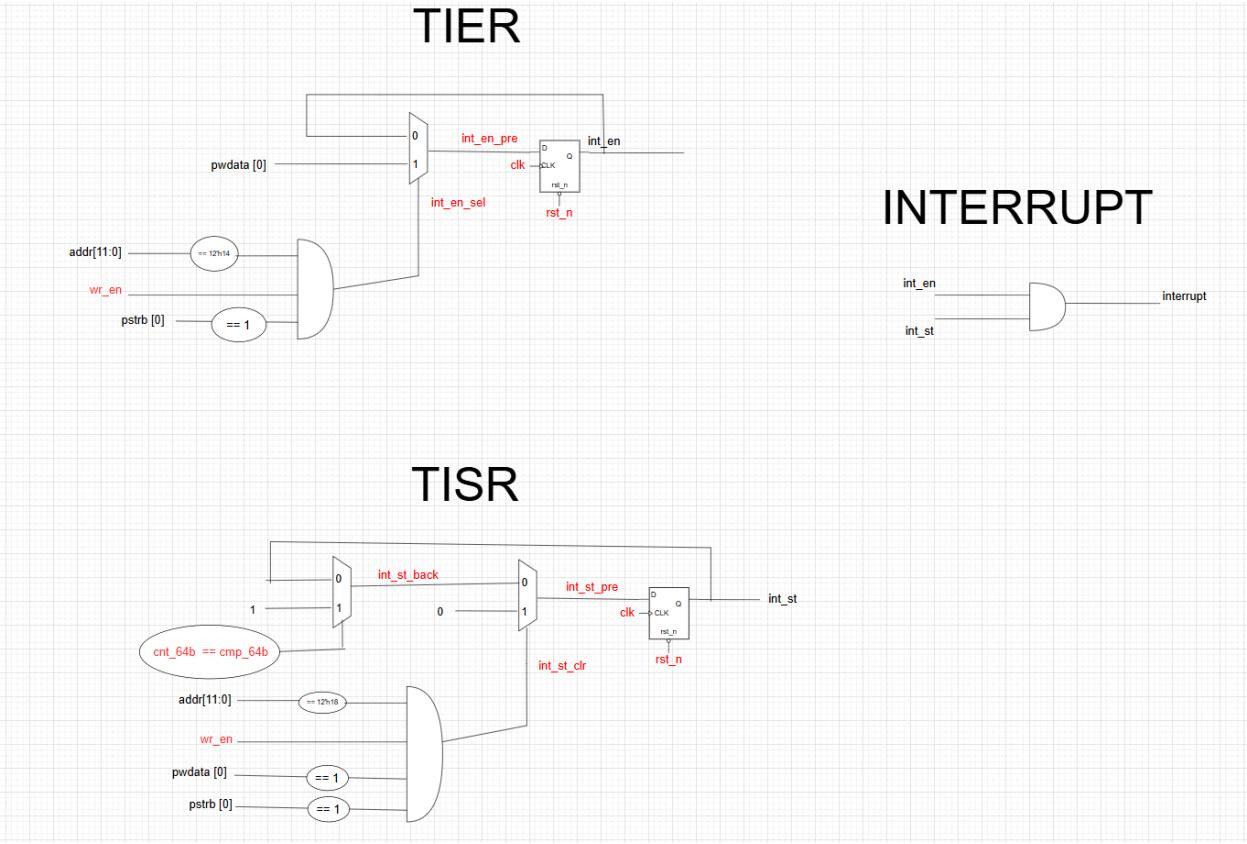


Fig 15. Interrupt Logic (TIER, TISR in Register) & Interrupt Block

- *Compare-match sets int_st and the tim.int will assert if the int_en is turn on too*
- *RW1C clear TISR*
- *tim_int = int_en & int_st*

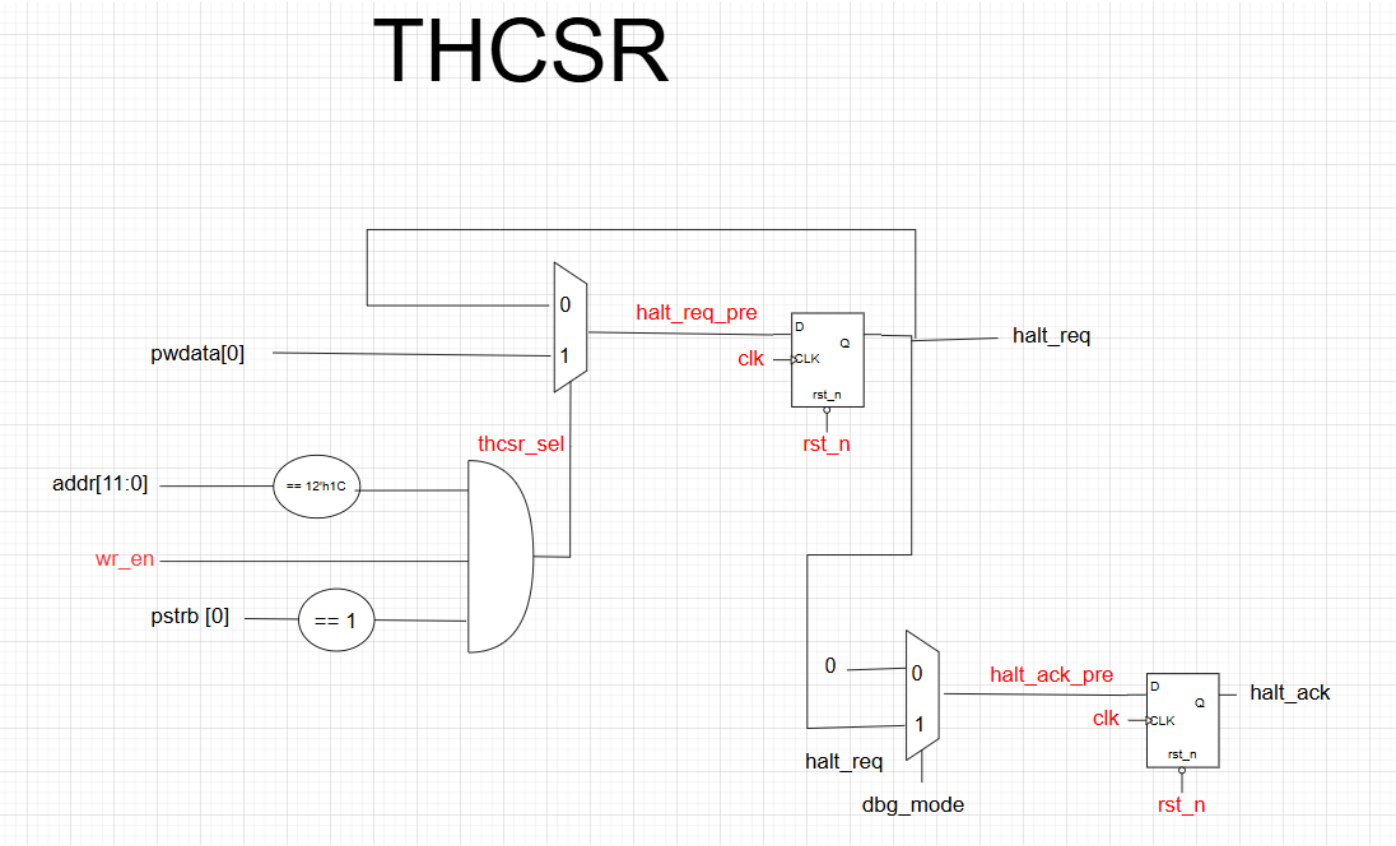


Fig 16. Halt Control Register (THCSR)

- *Works only when dbg_mode = 1*
- *Freeze counter + assert halt_ack*

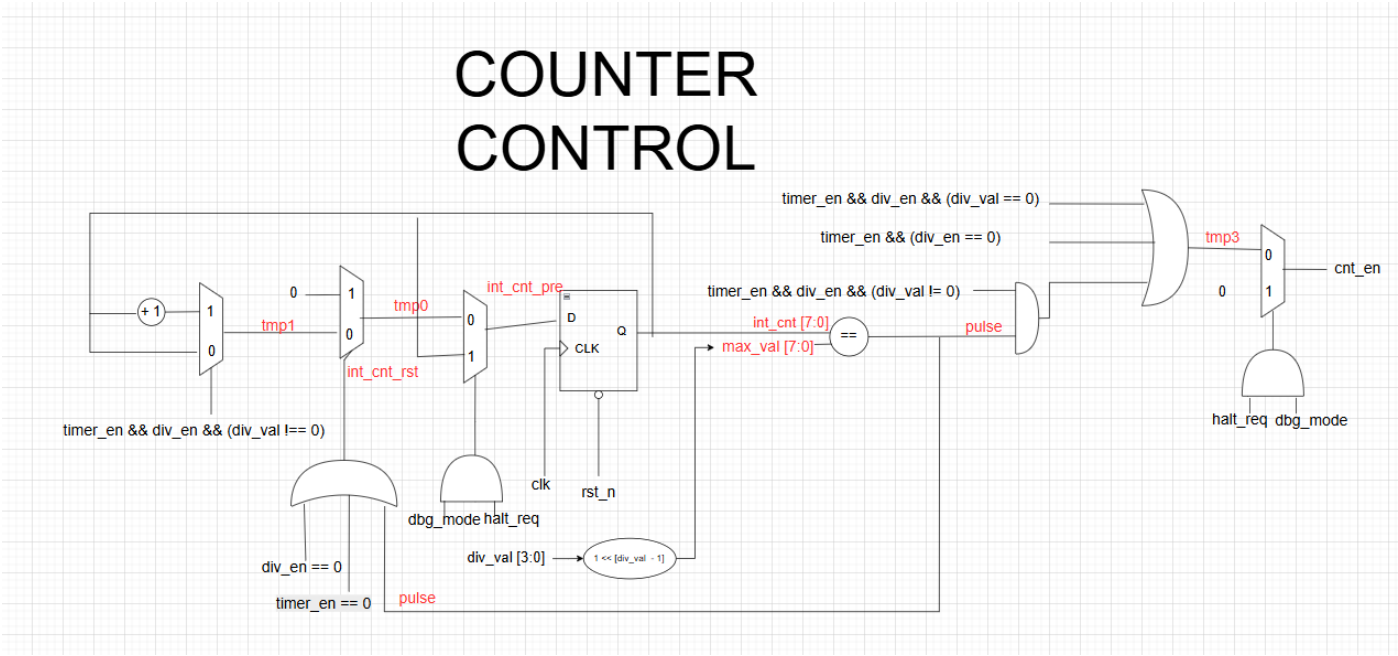


Fig 17. Counter Control Module

- *Generates cnt_en*
- *Normal mode or divided mode*
- *Prescaler resets when timer disabled*

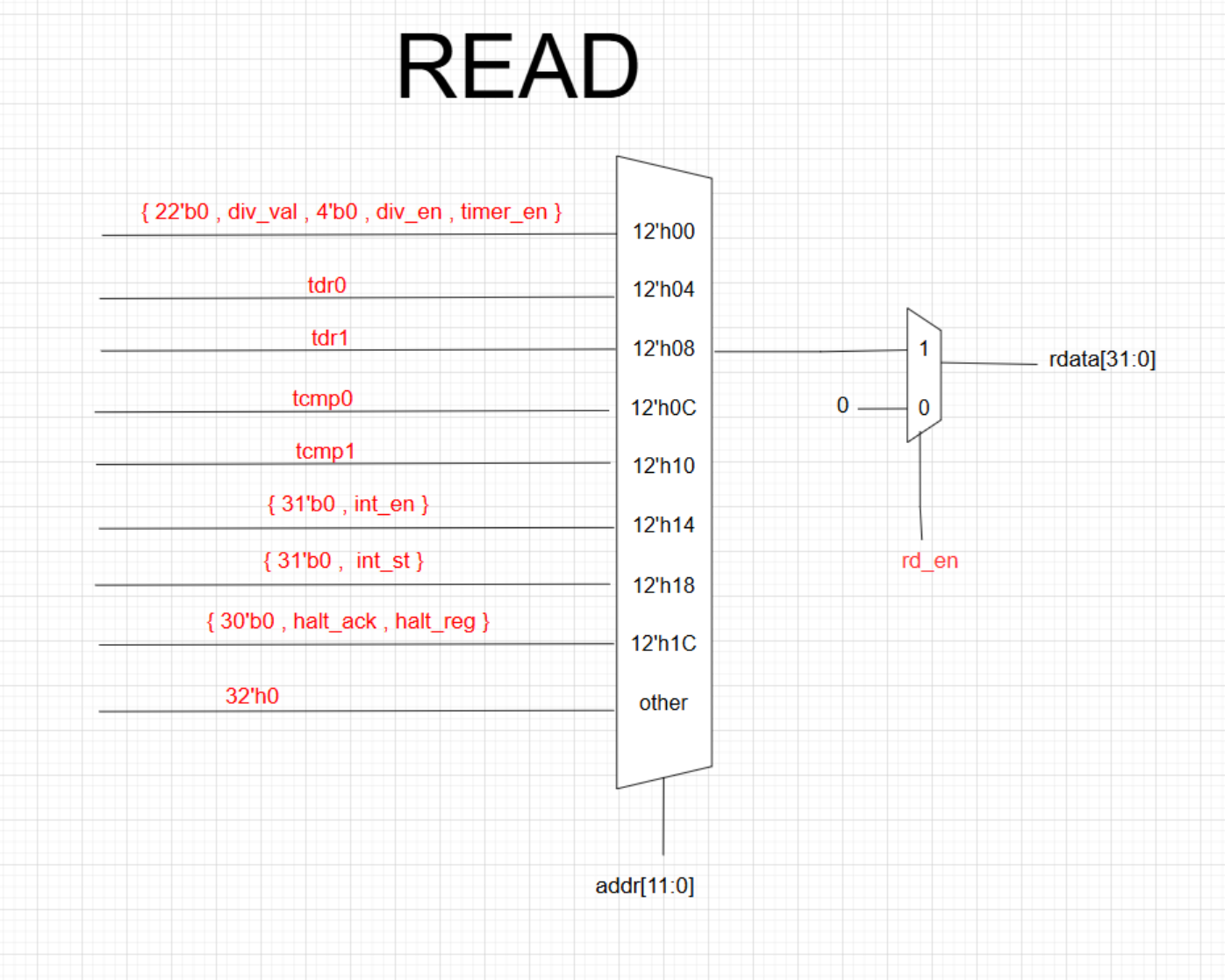


Fig 18. Register Read Path

• *rd_en selects register for PRDATA*

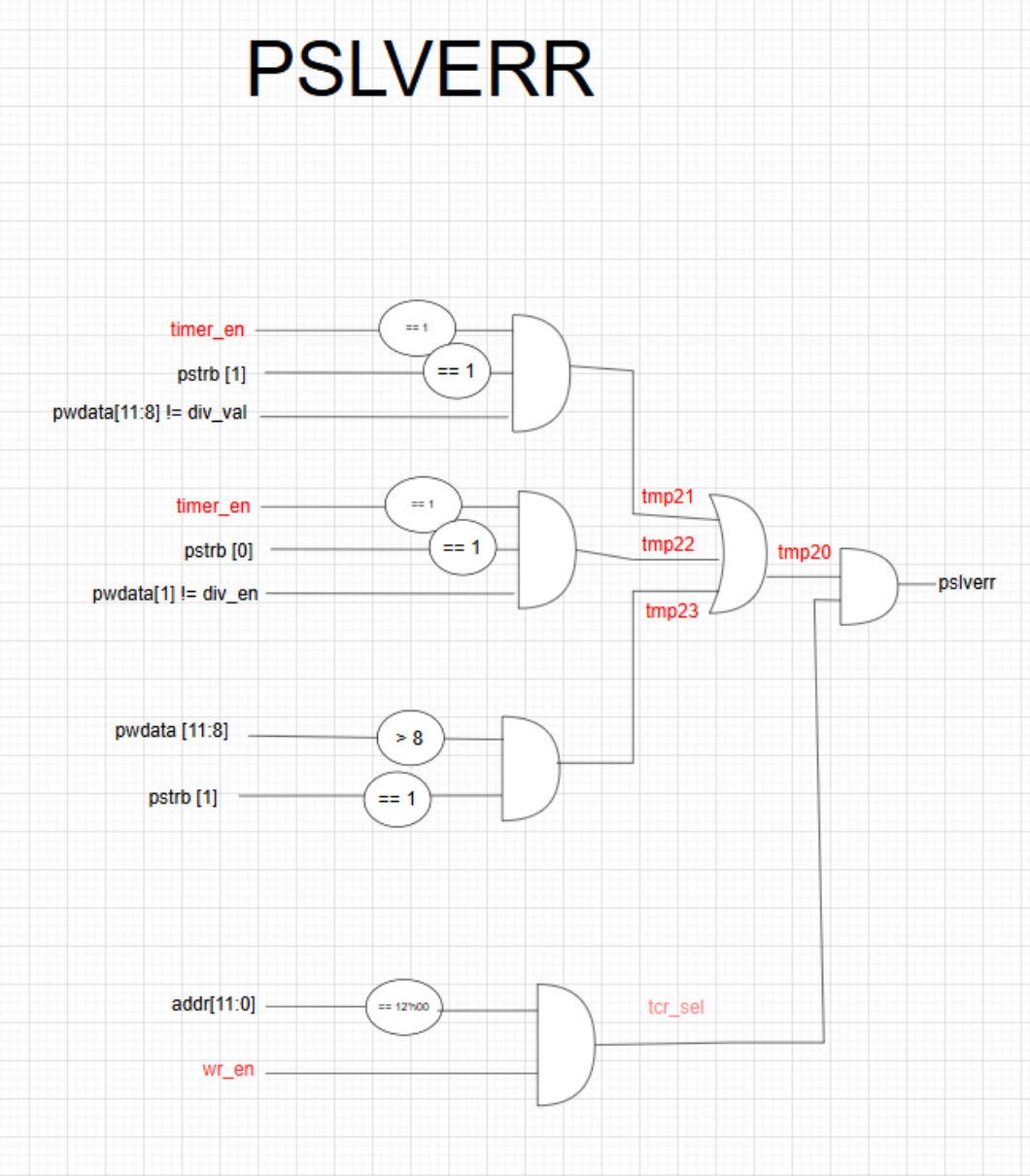


Fig 19. PSLVERR Logic

• *Illegal divider writes → PSLVERR*

1.4. Interface signals

Table 2. Interface signals of timer IP

Signal name	Width	Direction	Description
sys_clk	1	Input	System clock for the entire timer module. All internal logic operates on this clock.
sys_rst_n	1	Input	Active–low system reset. Resets all registers and internal states.
tim_psel	1	Input	APB select signal. Indicates valid APB transfer is requested.
tim_pwrite	1	Input	APB write control (1 = write, 0 = read).
tim_penable	1	Input	APB enable phase indicator.
tim_paddr[11:0]	12	Input	APB address bus selecting timer registers.
tim_pwdata[31:0]	32	Input	APB write data bus.
tim_prdata[31:0]	32	Output	APB read data bus returning data from selected register.
tim_pstrb[3:0]	4	Input	Byte strobe for partial write. Controls which bytes of the register are updated.
tim_pready	1	Output	APB ready signal. Indicates transfer completion.
tim_pslverr	1	Output	APB error response. Asserted when illegal register write occurs.
tim_int	1	Output	Timer interrupt output. Active when compare match occurs AND interrupt enabled.
dbg_mode	1	Input	Debug mode indicator. Enables halt mechanism. Should not change while timer_en = 1 during operation.

Table 3. Interface signals of APB Register Counter

Signal name	Width	Direction	Description
clk	1	Input	System clock.
rst_n	1	Input	Active-low reset.
paddr	12	Input	APB address bus.
pstrb	4	Input	Write byte-strobe for partial update.
pwdata	32	Input	APB write data.
psel	1	Input	APB select signal.
pwrite	1	Input	APB write control.
penable	1	Input	APB enable phase.
pready	1	Output	APB ready response. Indicates transfer completion.
prdata	32	Output	APB read data.
pslverr	1	Output	APB error response for illegal write.
cnt_en	1	Input	Count-enable pulse from counter controller.
timer_en	1	Output	Timer enable decoded from TCR.
div_en	1	Output	Divider enable decoded from TCR.
div_val	4	Output	Clock-divider ratio (TCR bits[11:8]).
halt_req	1	Output	Debug halt request (THCSR).
cnt_64b	64	Output	64-bit timer counter value.

Signal name	Width	Direction	Description
int_en	1	Output	Interrupt enable (TIER).
int_st	1	Output	Interrupt status flag (TISR).
dbg_mode	1	Input	Debug mode indicator.

Table 4. Interface signals of counter control block

Signal name	Width	Direction	Description
clk	1	Input	System clock.
rst_n	1	Input	Active-low reset.
halt_req	1	Input	Request to halt counter in debug mode.
timer_en	1	Input	Timer run control from TCR.
div_en	1	Input	Divider enable.
div_val	4	Input	Divider value (0–8).
dbg_mode	1	Input	Debug mode enable (halts counter when halt_req=1).
cnt_en	1	Output	1-cycle pulse used to increment 64-bit counter.

Table 5. Interface signals of interrupt block

Signal name	Width	Direction	Description
int_en	1	Input	Interrupt enable bit.
int_st	1	Input	Interrupt status flag.
interrupt	1	Output	Final interrupt output (int_en AND int_st).

2.Register Specification

2.1. Register Summary

Table 6. Register Summary

Address	Abbreviation	Register name
0x00	TCR	Timer Control Register
0x04	TDR0	Timer Data Register 0 (lower 32 bits of counter)
0x08	TDR1	Timer Data Register 1 (upper 32 bits of counter)
0x0C	TCMP0	Timer Compare Register 0 (lower 32 bits)
0x10	TCMP1	Timer Compare Register 1 (upper 32 bits)
0x14	TIER	Timer Interrupt Enable Register
0x18	TISR	Timer Interrupt Status Register
0x1C	THCSR	Timer Halt Control Status Register
Others	—	Reserved (RAZ/WI)

2.2. Timer Control Register (TCR)

Offset address: 0x0

Reset value: 0x0000_0000

Bit field

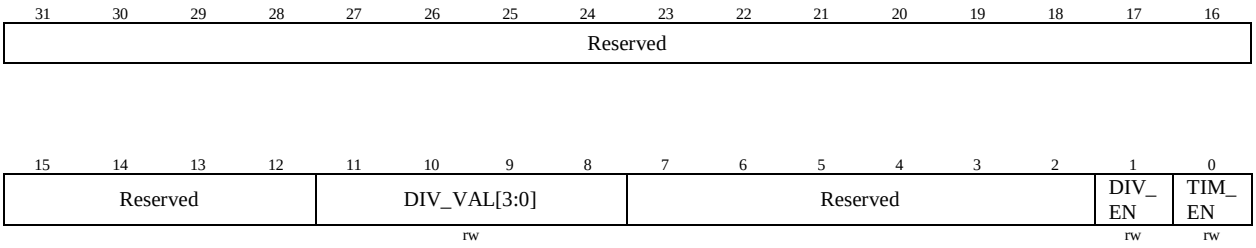


Table 7. Timer Control Register (TCR)

Bit	Name	Type	Default	Description
31:12	Reserved	-	0	Reserved.

Bit	Name	Type	Default	Description
11:8	DIV_VAL[3:0]	RW	4'b0001	Counter division factor. 0000: /1, 0001: /2, … 1000: /256. Others prohibited.
7:2	Reserved	RO	0	Reserved.
1	DIV_EN	RW	0	1: Enable divider mode. 0: Normal counting mode.
0	TIM_EN	RW	0	Timer enable. 1 = start counter.

2.3. Timer Data Register 0 (TDR0)

Offset: 0x04
Reset value: 0x0000_0000

Bit field

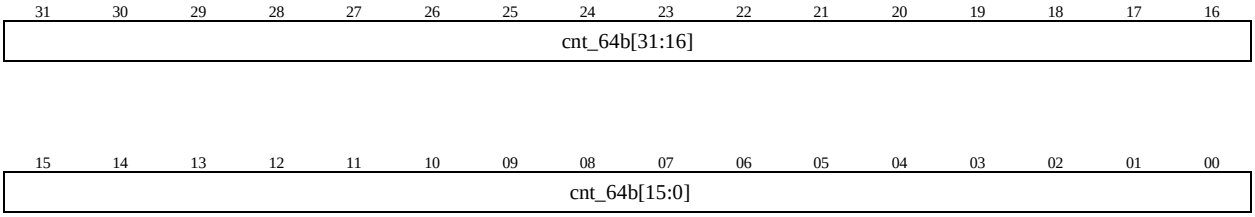


Table 8. Register TDR0

Bit	Name	Type	Default	Description
31:0	TDR0	RW	0	Lower 32 bits of 64-bit counter. Cleared when TIM_EN goes 1→0 (advanced level).

2.4. Timer Data Register 1 (TDR1)

Offset: 0x08
Reset value: 0x0000_0000

Bit field

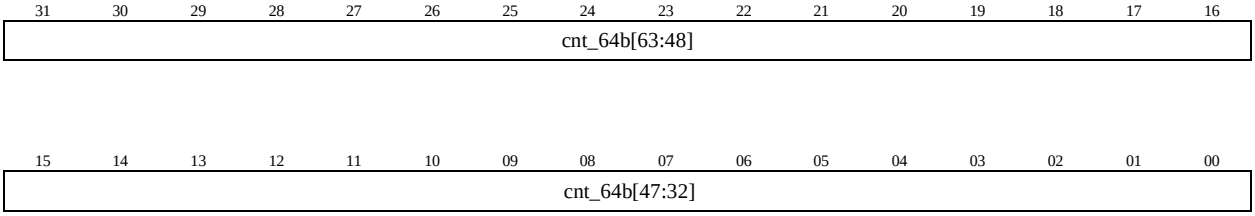


Table 9. Register TDR1

Bit	Name	Type	Default	Description
31:0	**TDR1**	**RW**	**0**	**Upper 32 bits of 64-bit counter. Cleared when TIM_EN goes 1→0 (advanced level).**

2.5. Timer Compare Register 0 (TCMP0)

Offset: 0x0C
Reset value: 0xFFFF_FFFF

Bit field

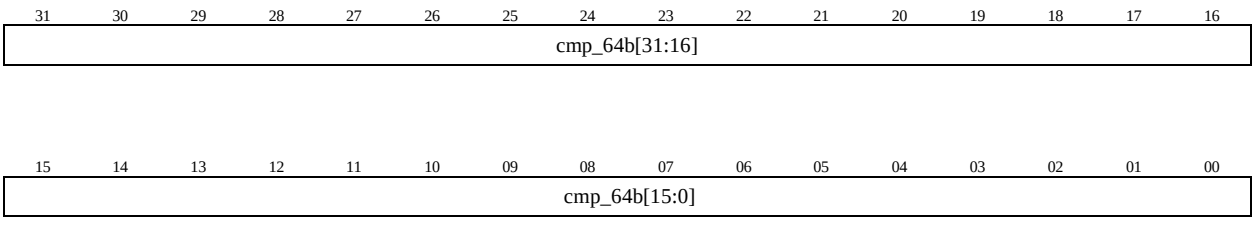


Table 10. Register TCMP0

Bit	Name	Type	Default	Description
31:0	**TCMP0**	**RW**	**0xFFFF_FFFF**	**Lower 32 bits of compare value. Interrupt pending when counter == TCMP.**

2.6. Timer Compare Register 1 (TCMP1)

Offset: 0x10
Reset value: 0xFFFF_FFFF

Bit field

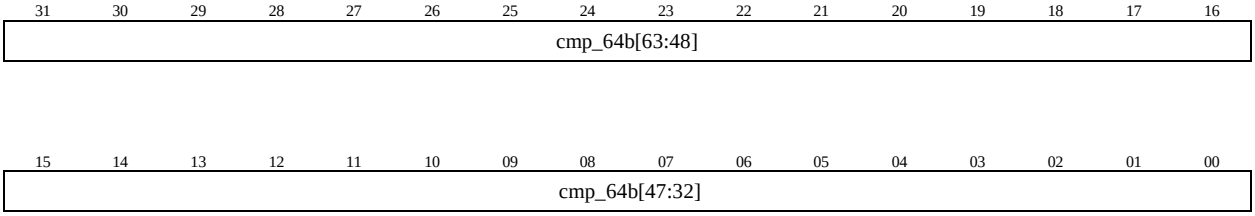


Table 11. Register TCMP1

Bit	Name	Type	Default	Description
31:0	TCMP1	RW	0xFFFF_FFFF	Upper 32 bits of compare value.

2.7. Timer Interrupt Enable Register (TIER)

Offset: 0x14
Reset value: 0x0000_0000

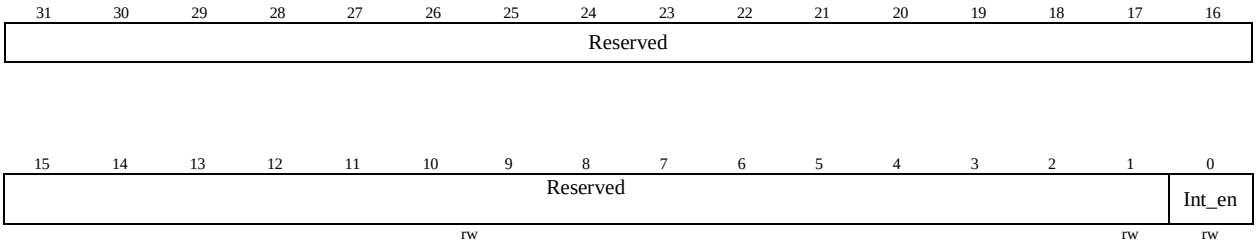


Table 12. Register TIER

Bit	Name	Type	Default	Description
31:1	Reserved	RO	0	Reserved.
0	INT_EN	RW	0	1: Enable timer interrupt. 0: Disable interrupt output.

2.8. Timer Interrupt Status Register (TISR)

Offset: 0x18
Reset value: 0x0000_0000

Bit field

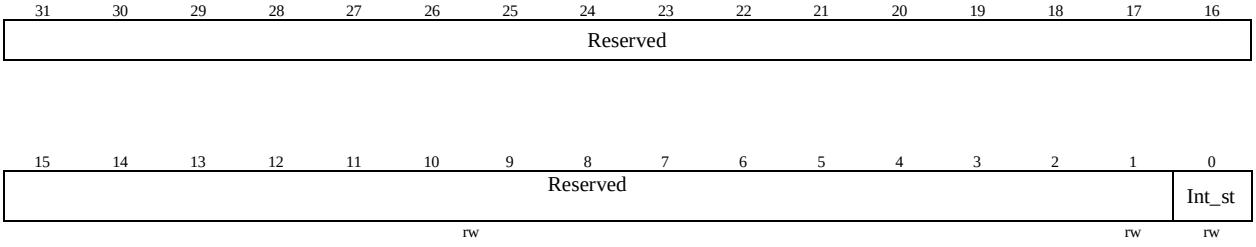


Table 13. Register TISR

Bit	Name	Type	Default	Description
31:1	Reserved	RO	0	Reserved.
0	INT_ST	RW1C	0	Interrupt pending bit. Cleared by writing 1 when set.

2.9. Timer Halt Control Status Register (THCSR)

Offset: 0x1C
Reset value: 0x0000_0000

Bit field

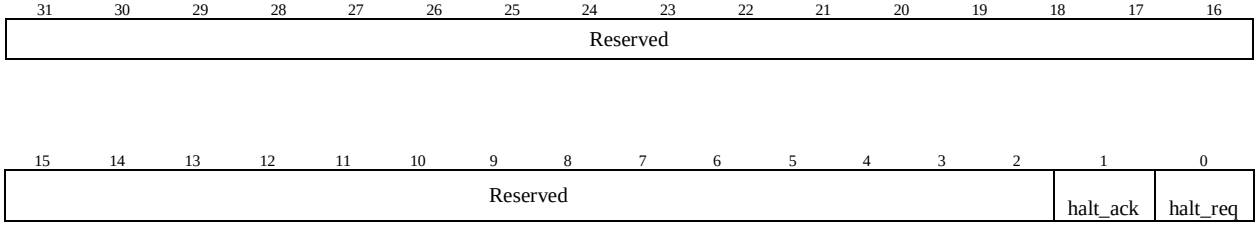


Table 14. Register THCSR

Bit	Name	Type	Default	Description
31:2	Reserved	RO	0	Reserved.
1	HALT_ACK	RO	0	Timer halted acknowledge (advanced level only).
0	HALT_REQ	RW	0	Timer halt request in debug mode.

3. Functional Description

3.1. APB slave – Register - Counter

APB Slave Interface

- Receives APB read/write transactions via psel, penable, and pwrite.
- Generates two internal 1-cycle strobes:
  - **wr_en** – asserted exactly once for each valid write transfer.
  - **rd_en** – asserted exactly once for each valid read transfer.
- Ensures single-update behavior: each APB transaction updates internal logic only once.
- Supports byte-write access using pstrb[3:0].

Register Handling

- Contains all programmable registers:
  TCR, **TDR0**, **TDR1**, **TCMP0**, **TCMP1**, **TIER**, **TISR**, **THCSR**.
- Register values are updated only when wr_en = 1 and the write is legal.
- Illegal writes are ignored while the design asserts pslverr.

64-bit Counter Integration

- TDR0/TDR1 always expose the live 64-bit counter value for APB reads.
- Counter increments whenever cnt_en = 1.
- When timer_en = 0:
  - Counter is cleared and held at **0**.
- When timer_en transitions **0 → 1**:
  - Counter reloads from {TDR1, TDR0}.
- Counter wraps around on overflow (modulo 2^{64}).

Error & Protection

- Divider settings cannot be modified while timer is running (timer_en = 1):
  - Illegal writes → pslverr = 1.

  - Register retains old value.
- Writing invalid divider values (div_val > 8) also asserts pslverr.

APB Ready Behavior

- pready = wr_en | rd_en
  → Ensures a deterministic **1-cycle wait-state** for all APB accesses.

3.2. Counter Control

cnt_en Generation

- Produces the internal increment pulse for the 64-bit counter.
- All logic is synchronous with clk.

Normal Mode (div_en = 0)

- cnt_en = 1 on every clock cycle.
- Counter increments at full system clock speed.

Divider Mode (div_en = 1)

- Uses an internal 8-bit prescaler (int_cnt).
- Prescaler increases every clock cycle.
- When int_cnt == (1 << div_val) - 1:
  - A single-cycle pulse cnt_en = 1 is generated.
  - Prescaler resets to 0.
- Provides programmable division ratios:
  /1, /2, /4, /8, … /256 depending on div_val.

Halt Mode (Debug Freeze)

- Counter halts only when:
  - dbg_mode = 1 **and**
  - halt_req = 1
- During halt:
  - Prescaler freezes.
  - cnt_en = 0 → counter stops.

- When halt_req is cleared:
    - Counter resumes smoothly without losing timing phase.

Reset Behavior

- When rst_n = 0: int_cnt = 0.
- When timer_en high to low: prescaler resets.

3.3. Interrupt

Compare-Match Condition

- Interrupt is triggered when:
- cnt_64b == {TCMP1, TCMP0}
- This sets int_st = 1.

Interrupt Enable Logic

- tim_int = int_en & int_st
- Clearing int_en masks the interrupt output but keeps the pending status internally.

RW1C Status Clear

- TISR.int_st is **write-1-to-clear**:
    - Writing **1** clears the interrupt.
    - Writing **0** has no effect.
- If int_st = 0, writes to the bit are ignored.

Continuous Compare

- The 64-bit counter never stops after a match.
- Compare logic evaluates every cycle.
- Interrupt stays asserted until:
    - Software clears int_st, or
    - Software disables int_en.

4. Sample Waveform

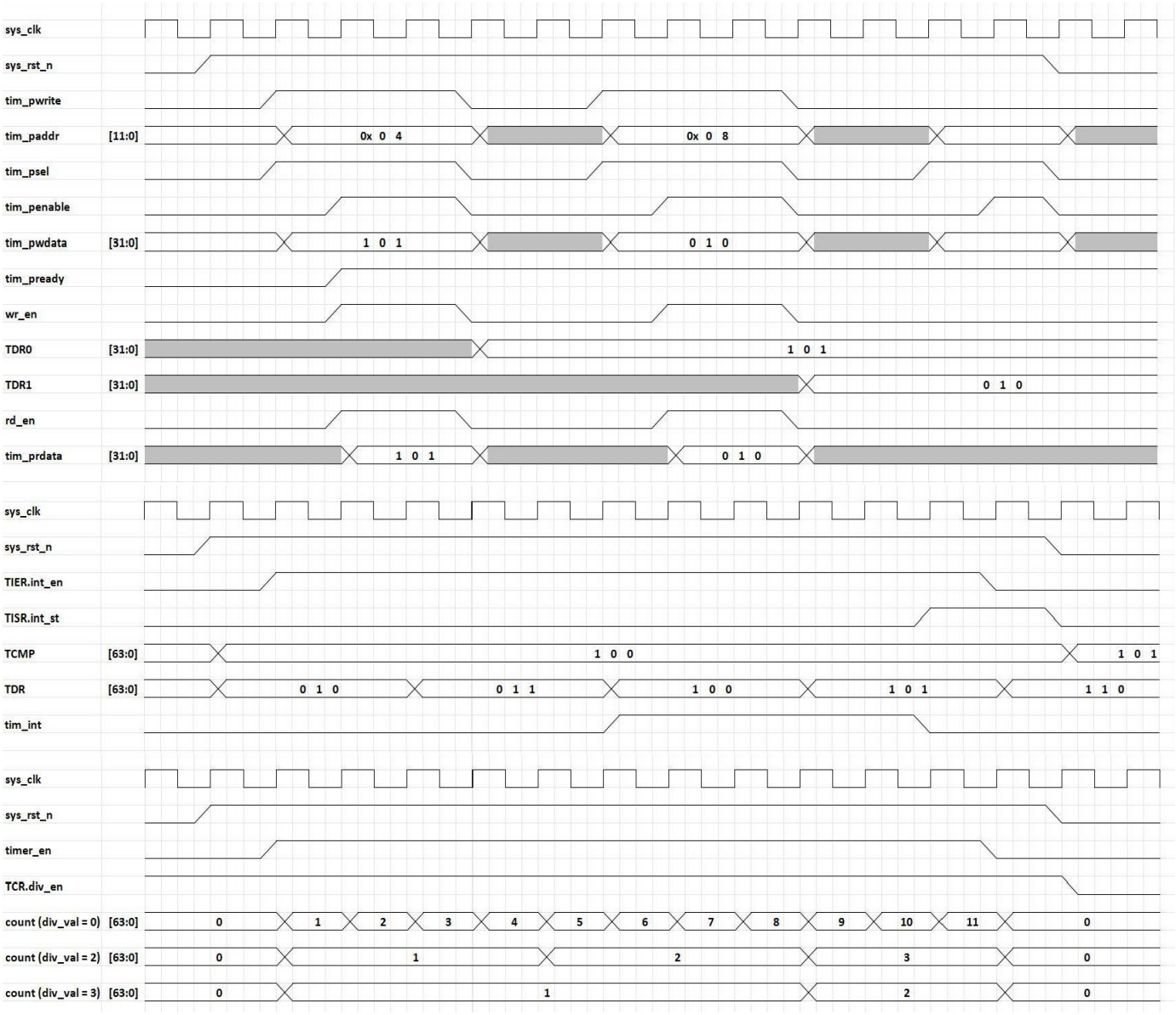


Fig 20. Sample Waveform

5. VPLAN

Table 15. Vplan

ID	Item	Sub item 1	Sub item 2	Test sequence	Pass condition
1	INITIAL VALUE	1.1 Register	TCR REG	After reset is released, perform read access to TCR	Read value = 32'h0000_0100
			TDR0 REG	After system reset is released, perform read access to TDR0	Read value = 32'h0000_0000
			TDR1 REG	After reset is released, perform read access to TDR1	Read value = 32'h0000_0000
			TCMP0 REG	After reset is released, perform read access to TCMP0	Read value = 32'hFFFF_FFFF
			TCMP1 REG	After reset is released, perform read access to TCMP1	Read value = 32'hFFFF_FFFF
			TIER REG	After system reset is released, perform read access to TIER	Read value = 32'hFFFF_FFFF
			TISR REG	After reset is released, perform read access to TISR	Read value = 32'h0000_0000
			THCSR REG	After reset is released, perform read access to THCSR	Read value = 32'hFFFF_FFFF
		1.2 counter		After reset is released, perform read access to TDR0 & TDR1.	TDR0/1 Read value = 32'h0000_0000 (cnt_64b = 64'h0000_0000_0000_0000)
		1.3 intterupt		After reset is released, perform im_int	tim_int = 0
		2.1 Register access	TCR REG	1.Write D0 & read back and compare 2.Write FFFF_F8FF & readback & compare 3.Write DF & read back and compare 4.Write DA & read back and compare	rdata = 32'h0000_0000 rdata = 32'h0000_0803 rdata = 32'h0000_0803 rdata = 32'h0000_0803
			TDR0 REG	1.Write D0 & read back and compare 2.Write D5 & readback & compare 3.Write DA & readback & compare 4.Write DF & readback & compare	read value is same as write value
			TDR1 REG	1.Write D0 & read back and compare 2.Write D5 & readback & compare 3.Write DA & readback & compare 4.Write DF & readback & compare	read value is same as write value
			TCMP0 REG	1.Write D0 & read back and compare 2.Write D5 & readback & compare 3.Write DA & readback & compare 4.Write DF & readback & compare	read value is same as write value
			TCMP1 REG	1.Write D0 & read back and compare 2.Write D5 & readback & compare 3.Write DA & readback & compare 4.Write DF & readback & compare	read value is same as write value
			TIER REG	1.Write D0 & read back and compare 2.Write D5 & readback & compare 3.Write DA & readback & compare 4.Write DF & readback & compare	rdata = 32'h0000_0000 rdata = 32'h0000_0001 rdata = 32'h0000_0000 rdata = 32'h0000_0001
			TISR REG	1.Write D0 & read back and compare 2.Write D5 & readback & compare 3.Write DA & readback & compare 4.Write DF & readback & compare	rdata = 32'h0000_0000
			THCSR REG	1.Write D0 & read back and compare 2.Write D5 & readback & compare 3.Write DA & readback & compare 4.Write DF & readback & compare	rdata = 32'h0000_0000 rdata = 32'h0000_0001 rdata = 32'h0000_0000 rdata = 32'h0000_0001
2	REGISTER	2.2 Byte access	TCR REG	1. Set pstrb = 0001 then write D3 and read back 2. Set pstrb = 0010 then write D3 and read back 3. Set pstrb = 0100 then write D3 and read back 4. Set pstrb = 1000 then write D3 and read back 5. Set pstrb = 0011 then write D3 and read back 6. Set pstrb = 1100 then write D5 and read back	rdata = 32'h0000_0103 rdata = 32'h0000_0103 rdata = 32'h0000_0103 rdata = 32'h0000_0103 rdata = 32'h0000_0103 rdata = 32'h0000_0103
			TDR0 REG	1. Set pstrb = 0001 then write DF and read back 2. Set pstrb = 0010 then write DF and read back 3. Set pstrb = 0100 then write DF and read back 4. Set pstrb = 1000 then write DF and read back 5. Set pstrb = 0011 then write DF and read back 6. Set pstrb = 1100 then write DF and read back	rdata = 32'h0000_00ff rdata = 32'h0000_ff00 rdata = 32'h00ff_0000 rdata = 32'hff00_0000 rdata = 32'h0000_ffff rdata = 32'hffff_0000
			TDR1 REG	1. Set pstrb = 0001 then write DF and read back 2. Set pstrb = 0010 then write DF and read back 3. Set pstrb = 0100 then write DF and read back 4. Set pstrb = 1000 then write DF and read back 5. Set pstrb = 0011 then write DF and read back 6. Set pstrb = 1100 then write DF and read back	rdata = 32'h0000_00ff rdata = 32'h0000_ff00 rdata = 32'h00ff_0000 rdata = 32'hff00_0000 rdata = 32'h0000_ffff rdata = 32'hffff_0000
			TCMP0 REG	1. Set pstrb = 0001 then write DF and read back 2. Set pstrb = 0010 then write DF and read back 3. Set pstrb = 0100 then write DF and read back 4. Set pstrb = 1000 then write DF and read back 5. Set pstrb = 0011 then write DF and read back 6. Set pstrb = 1100 then write DF and read back	rdata = 32'h0000_00ff rdata = 32'h0000_ff00 rdata = 32'h00ff_0000 rdata = 32'hff00_0000 rdata = 32'h0000_ffff rdata = 32'hffff_0000
			TCMP1 REG	1. Set pstrb = 0001 then write DF and read back 2. Set pstrb = 0010 then write DF and read back 3. Set pstrb = 0100 then write DF and read back 4. Set pstrb = 1000 then write DF and read back 5. Set pstrb = 0011 then write DF and read back 6. Set pstrb = 1100 then write DF and read back	rdata = 32'h0000_00ff rdata = 32'h0000_ff00 rdata = 32'h00ff_0000 rdata = 32'hff00_0000 rdata = 32'h0000_ffff rdata = 32'hffff_0000
			TIER REG	1. Set pstrb = 0001 then write DF and read back 2. Set pstrb = 0010 then write DF and read back 3. Set pstrb = 0100 then write DF and read back 4. Set pstrb = 1000 then write DF and read back 5. Set pstrb = 0011 then write DF and read back 6. Set pstrb = 1100 then write DF and read back	rdata = 32'h0000_0001 rdata = 32'h0000_0000 rdata = 32'h0000_0000 rdata = 32'h0000_0000 rdata = 32'h0000_0001 rdata = 32'h0000_0000

		2.2 Byte access	TCMP1 REG	1. Set pstrb = 0001 then write DF and read back 2. Set pstrb = 0010 then write DF and read back 3. Set pstrb = 0100 then write DF and read back 4. Set pstrb = 1000 then write DF and read back 5. Set pstrb = 0011 then write DF and read back 6. Set pstrb = 1100 then write DF and read back	rdata = 32'h0000_00ff rdata = 32'h0000_ff00 rdata = 32'h00ff_0000 rdata = 32'hff00_0000 rdata = 32'h0000_ffff rdata = 32'hffff_0000
			TIER REG	1. Set pstrb = 0001 then write DF and read back 2. Set pstrb = 0010 then write DF and read back 3. Set pstrb = 0100 then write DF and read back 4. Set pstrb = 1000 then write DF and read back 5. Set pstrb = 0011 then write DF and read back 6. Set pstrb = 1100 then write DF and read back	rdata = 32'h0000_0001 rdata = 32'h0000_0000 rdata = 32'h0000_0000 rdata = 32'h0000_0000 rdata = 32'h0000_0001 rdata = 32'h0000_0000
			TISR REG	1. Set pstrb = 0001 then write DF and read back 2. Set pstrb = 0010 then write DF and read back 3. Set pstrb = 0100 then write DF and read back 4. Set pstrb = 1000 then write DF and read back 5. Set pstrb = 0011 then write DF and read back 6. Set pstrb = 1100 then write DF and read back	rdata = 32'h0000_0000
			THCSR REG	1. Set pstrb = 0001 then write DF and read back 2. Set pstrb = 0010 then write DF and read back 3. Set pstrb = 0100 then write DF and read back 4. Set pstrb = 1000 then write DF and read back 5. Set pstrb = 0011 then write DF and read back 6. Set pstrb = 1100 then write DF and read back	rdata = 32'h0000_0001 rdata = 32'h0000_0000 rdata = 32'h0000_0000 rdata = 32'h0000_0000 rdata = 32'h0000_0001 rdata = 32'h0000_0000
		2.3 Reserved access		1.Write DF to 0x100 (first reserved address) and read back 2. Write DF to 0x595 (random middle reserved address) and read back 3. Write DF to 0x3FC (last reserved address) and read back	Read back value is always 0
		2.4 one_hot_check		1. Write 32'h11111112 to TCR 2. Write D2 to TDR0 3. Write D3 to TDR1 4. Write D4 to TCMP0 5. Write D5 to TCMP1 6. Write D9 to TIER 7. Write DA to TIST 8. Write DF to THCSR	1. TCR rdata = 32'h0000_0102 2. TDR0 rdata = 32'h2222_2222 3. TDR1 rdata = 32'h3333_3333 4. TCMP0 rdata = 32'h4444_4444 5. TCMP1 rdata = 32'h5555_5555 6. TIER rdata = 32'h0000_0001 7. TISR rdata = 32'h0000_0000 8. THCSR rdata = 32'h0000_0001
3	COUNTER	3.1 Counter run	normal	1. Set TCR.div_en = 0, div_val = 4'h3 thenTCR.timer_en = 1. 2. Set TCR.div_en = 0, div_val = 4'h3 then TCR.timer_en = 1.	cnt_64b increases by 1 every clock (TDR0/TDR1 reflect cnt value).
			div mode	1. Set TCR.div_en = 1, TCR.div_val = 4'h1 (divide by 2 default), TCR.timer_en = 1. 2. Set TCR.div_en = 1, TCR.div_val = 4'h2 (divide by 4 default), TCR.timer_en = 1. 3. Set TCR.div_en = 1, TCR.div_val = 4'h3 (divide by 8 default), TCR.timer_en = 1. 4. Set TCR.div_en = 1, TCR.div_val = 4'h4 (divide by 16 default), TCR.timer_en = 1. 5. Set TCR.div_en = 1, TCR.div_val = 4'h5 (divide by 32 default), TCR.timer_en = 1. 6. Set TCR.div_en = 1, TCR.div_val = 4'h6 (divide by 64 default), TCR.timer_en = 1. 7. Set TCR.div_en = 1, TCR.div_val = 4'h7 (divide by 128 default), TCR.timer_en = 1. 8. Set TCR.div_en = 1, TCR.div_val = 4'h8 (divide by 256 default), TCR.timer_en = 1. 9. Set TCR.div_en = 1, TCR.div_val = 4'h0 (divide by 1 default), TCR.timer_en = 1.	1. cnt increments at expected rate per div_val (for div_val=1 → period = 2 clocks). 2. cnt increments at expected rate per div_val (for div_val=2 → period = 4 clocks). 3. cnt increments at expected rate per div_val (for div_val=3 → period = 8 clocks). 4. cnt increments at expected rate per div_val (for div_val=4 → period = 16 clocks). 5. cnt increments at expected rate per div_val (for div_val=5 → period = 32 clocks). 6. cnt increments at expected rate per div_val (for div_val=6 → period = 64 clocks). 7. cnt increments at expected rate per div_val (for div_val=7 → period = 128 clocks). 8. cnt increments at expected rate per div_val (for div_val=8 → period = 256 clocks). 9. cnt increments at expected rate per div_val (for div_val=0 → period = 1 clocks).
		3.2 Counter clear		Set timer_en=1, then write timer_en=0, then read TDR0/TDR1	TDR0/TDR1 cleared to initial value
		3.3 Counter after intterupt		Set compare so counter will overflow (increment until wrap) and keep counting	Cnt_64b continues counting; no hang.
4	INTERRUPT	4.1 Interrupt assert & hold pending		1. Program compare so that cmp_64b > cnt_64b. 2. Write TIER.int_en = 1. 3. Enable counter (TCR.timer_en = 1). 4. Wait until cnt_64b == cmp_64b. 5. Read TISR & check tim_int.	TISR.int_st set to 1 and tim_int output asserted when int_en=1 & cmp_64b = cnt_64b
		4.2 Masking by TIER		Set TIER.int_en=0 while TISR.int_st=1	tim_int output becomes 0 (masked) but TISR.int_st remains 1.
		4.3 RW1C clear both TISR.int_st & tim.int		1. Program compare so that cmp_64b > cnt_64b. 2. Write TIER.int_en = 1. 3. Enable counter (TCR.timer_en = 1). 4. Wait until cnt_64b == cmp_64b. 5. Read TISR & check tim_int. 6. Write 1 to TISR.int_st, then read back TISR & TIER	TISR.int_st becomes 0; tim_int clear to 0
5	APB	5 Pslverr	Set prohibited div_val	Write prohibited div_val (e.g: 4'hF, 4'hA…) to TCR	Slave asserts PSLVERR=1, register unchanged.
			Change div_val while timer_en=1	With timer_en=1, write 32'h000_0401 at TCR to change div_val	Slave asserts PSLVERR=1 and ignores write.
			Change div_en while timer_en=1	With timer_en=1, write 32'h000_0403 at TCR to change div_val	Slave asserts PSLVERR=1 and ignores write.
6	HALT	6 Counter run normal & div mode	No halt when dbg_mode=0	With dbg_mode=0 write THCSR.halt_req=1	No halt: counter continues; halt_ack remains 0.
			Halt request in debug mode	Set dbg_mode=1, write THCSR.halt_req = 1 (and timer_en=1)	cnt_64b freezes and THCSR.halt_ack becomes 1.
			clearing halt_req	After THCSR.halt_ack = 1, write THCSR.halt_req = 0	Counter resumes counting; halt_ack clears to 0.

6. URL FILE CODE

File RTL + testbench + testcases + coverage:
https://drive.google.com/drive/folders/1UxvluI8XcAPy9teDnGtUBw6CFXaZmL5v?usp=sharing

RTL CODE

apb_reg_cnt.v

cnt_ctrl.v

interrupt.v

timer_top.v

TESTCASE

1_initial_value.v

21_register_access.v

22_register_byte_acces

23_register_reserved_access.v

24_register_one_hot.v

31_counter_run.v

32_counter_clear.v

33_counter_after_interrupt.v

4_interrupt_check.v

5_apb_pslverr.v

6_halt.v

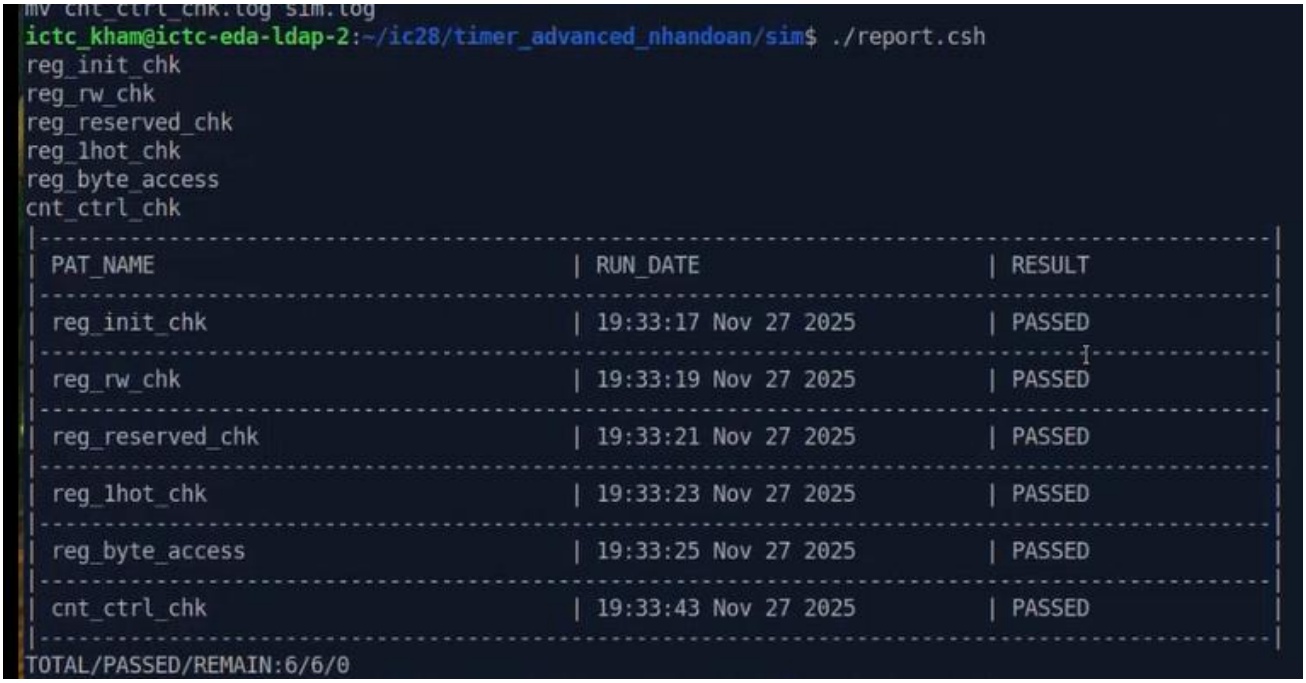
7. RESULT

RESULT WITH TESTBENCH + TESTCASES

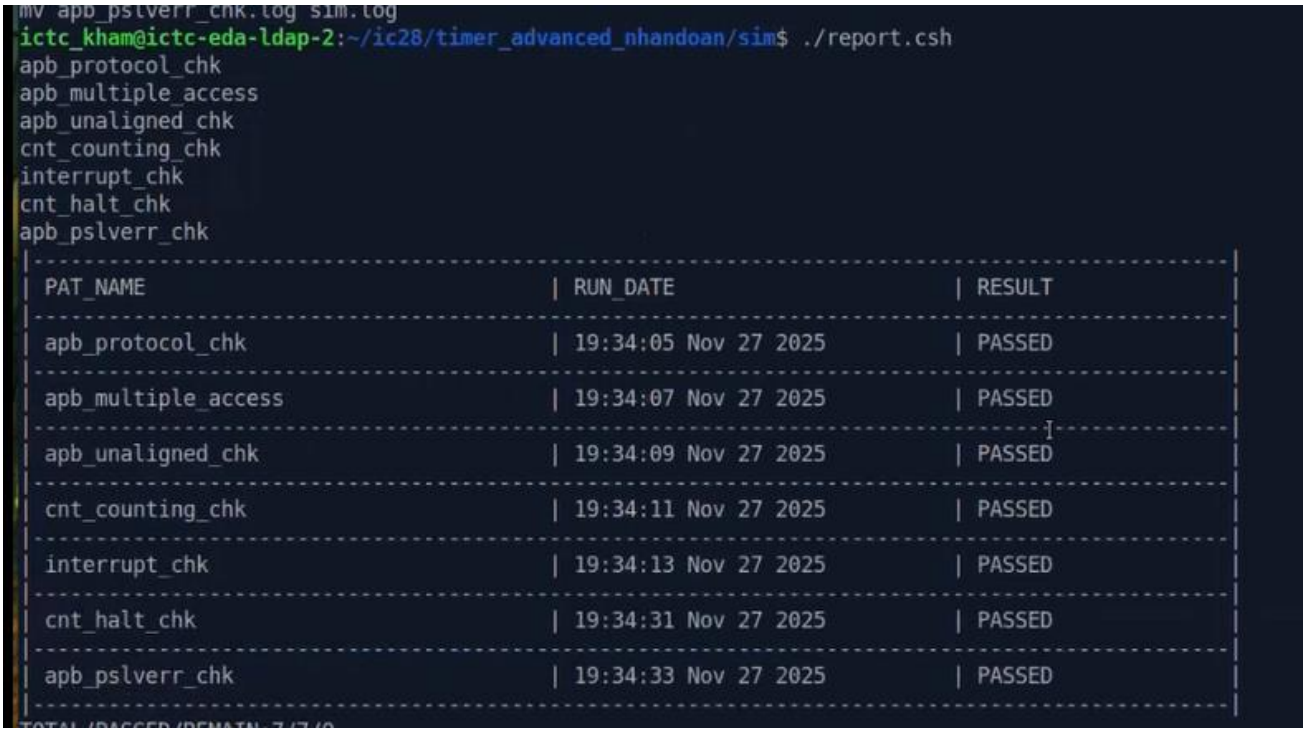
```
lenhan_1201@ictc-eda-ldap-2:~/final_project/sim$ ./report.csh
1_initial_value
21_register_access
22_register_byte_access
23_register_reserved_access
24_register_one_hot
31_counter_run
32_counter_clear
33_counter_after_interrupt
4_interrupt_check
5_apb_pslverr
6_halt
|-----------------------------------------------------------------------------------------|
| PAT_NAME                          | RUN_DATE                    | RESULT           |
|-----------------------------------------------------------------------------------------|
| 1_initial_value                   | 21:37:32 Nov 28 2025        | PASSED           |
|-----------------------------------------------------------------------------------------|
| 21_register_access                | 21:37:34 Nov 28 2025        | PASSED           |
|-----------------------------------------------------------------------------------------|
| 22_register_byte_access           | 21:37:36 Nov 28 2025        | PASSED           |
|-----------------------------------------------------------------------------------------|
| 23_register_reserved_access       | 21:37:38 Nov 28 2025        | PASSED           |
|-----------------------------------------------------------------------------------------|
| 24_register_one_hot               | 21:37:40 Nov 28 2025        | PASSED           |
|-----------------------------------------------------------------------------------------|
| 31_counter_run                    | 21:37:42 Nov 28 2025        | PASSED           |
|-----------------------------------------------------------------------------------------|
| 32_counter_clear                  | 21:37:43 Nov 28 2025        | PASSED           |
|-----------------------------------------------------------------------------------------|
| 33_counter_after_interrupt        | 21:37:45 Nov 28 2025        | PASSED           |
|-----------------------------------------------------------------------------------------|
| 4_interrupt_check                 | 21:37:47 Nov 28 2025        | PASSED           |
|-----------------------------------------------------------------------------------------|
| 5_apb_pslverr                     | 21:37:49 Nov 28 2025        | PASSED           |
|-----------------------------------------------------------------------------------------|
| 6_halt                            | 21:37:52 Nov 28 2025        | PASSED           |
|-----------------------------------------------------------------------------------------|
lenhan_1201@ictc-eda-ldap-2:~/final_project/sim$
```

Fig 21. Testcase Report

RESULT WITH GOLDEN MODEL



```
mv cnt_ctrl_chk.log sim.log
ictc_kham@ictc-eda-ldap-2:~/ic28/timer_advanced_nhandoan/sim$ ./report.csh
reg_init_chk
reg_rw_chk
reg_reserved_chk
reg_1hot_chk
reg_byte_access
cnt_ctrl_chk
|-----------------------------------------------------------------------------------------|
| PAT_NAME                          | RUN_DATE                    | RESULT       |
|-----------------------------------------------------------------------------------------|
| reg_init_chk                      | 19:33:17 Nov 27 2025        | PASSED       |
|-----------------------------------------------------------------------------------------|
| reg_rw_chk                        | 19:33:19 Nov 27 2025        | PASSED       |
|-----------------------------------------------------------------------------------------|
| reg_reserved_chk                  | 19:33:21 Nov 27 2025        | PASSED       |
|-----------------------------------------------------------------------------------------|
| reg_1hot_chk                      | 19:33:23 Nov 27 2025        | PASSED       |
|-----------------------------------------------------------------------------------------|
| reg_byte_access                   | 19:33:25 Nov 27 2025        | PASSED       |
|-----------------------------------------------------------------------------------------|
| cnt_ctrl_chk                      | 19:33:43 Nov 27 2025        | PASSED       |
|-----------------------------------------------------------------------------------------|
TOTAL/PASSED/REMAIN:6/6/0
```



```
mv apb_pslverr_chk.log sim.log
ictc_kham@ictc-eda-ldap-2:~/ic28/timer_advanced_nhandoan/sim$ ./report.csh
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       |
|-----------------------------------------------------------------------------------------|
| apb_protocol_chk                  | 19:34:05 Nov 27 2025        | PASSED       |
|-----------------------------------------------------------------------------------------|
| apb_multiple_access               | 19:34:07 Nov 27 2025        | PASSED       |
|-----------------------------------------------------------------------------------------|
| apb_unaligned_chk                 | 19:34:09 Nov 27 2025        | PASSED       |
|-----------------------------------------------------------------------------------------|
| cnt_counting_chk                  | 19:34:11 Nov 27 2025        | PASSED       |
|-----------------------------------------------------------------------------------------|
| interrupt_chk                     | 19:34:13 Nov 27 2025        | PASSED       |
|-----------------------------------------------------------------------------------------|
| cnt_halt_chk                      | 19:34:31 Nov 27 2025        | PASSED       |
|-----------------------------------------------------------------------------------------|
| apb_pslverr_chk                   | 19:34:33 Nov 27 2025        | PASSED       |
|-----------------------------------------------------------------------------------------|
```

Fig 22. Golden Model Report

COVERAGE

BEFORE EXCLUDE:

```
Coverage Report Summary Data by instance

================================================================================
=== Instance: /test_bench/timer_top_dut/u_apb_reg_cnt
=== Design Unit: work.apb_reg_cnt
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        97        97         0   100.00%
    Conditions                      38        38         0   100.00%
    Expressions                     97        94         3    96.90%
    Statements                     101       101         0   100.00%
    Toggles                        928       928         0   100.00%

================================================================================
=== Instance: /test_bench/timer_top_dut/u_cnt_ctrl
=== Design Unit: work.cnt_ctrl
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        10        10         0   100.00%
    Conditions                       5         5         0   100.00%
    Expressions                     12        11         1    91.66%
    Statements                      12        12         0   100.00%
    Toggles                         86        86         0   100.00%

================================================================================
=== Instance: /test_bench/timer_top_dut/u_interrupt
=== 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/timer_top_dut
=== Design Unit: work.timer_top
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
```

Fig.23 Sumary coverage report

Testbench never have 100% coverage because :

1. wr_en & rd_en never asserts when PSEL=0

```
================================Expression Details================================

Expression Coverage for instance /test_bench/timer_top_dut/u_apb_reg_cnt --

  File ../rtl/apb_reg_cnt.v
----------------Focused Expression View-----------------
Line       54 Item    1  (((~wr_en & psel) & pwrite) & penable)
Expression totals: 3 of 4 input terms covered = 75.00%

  Input Term   Covered  Reason for no coverage   Hint
 -----------  --------  -----------------------  --------------
        psel         N  '_0' not hit             Hit '_0'

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:         10  wr_en_0               (penable && pwrite && psel)
  Row   2:         10  wr_en_1               (penable && pwrite && psel)
  Row   3:    ***0***  psel_0                (penable && pwrite && ~wr_en)
  Row   4:         10  psel_1                (penable && pwrite && ~wr_en)
  Row   5:         10  pwrite_0              (penable && (~wr_en & psel))
  Row   6:         10  pwrite_1              (penable && (~wr_en & psel))
  Row   7:         10  penable_0             ((~wr_en & psel) & pwrite)
  Row   8:         10  penable_1             ((~wr_en & psel) & pwrite)
```

```
----------------Focused Expression View-----------------
Line       62 Item    1  (((pwrite ~| rd_en) & psel) & penable)
Expression totals: 3 of 4 input terms covered = 75.00%

  Input Term   Covered  Reason for no coverage   Hint
 -----------  --------  -----------------------  --------------
        psel         N  '_0' not hit             Hit '_0'

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:         10  pwrite_0              (penable && psel && ~rd_en)
  Row   2:         10  pwrite_1              (penable && psel && ~rd_en)
  Row   3:         10  rd_en_0               (penable && psel && ~pwrite)
  Row   4:         10  rd_en_1               (penable && psel && ~pwrite)
  Row   5:    ***0***  psel_0                (penable && (pwrite ~| rd_en))
  Row   6:         10  psel_1                (penable && (pwrite ~| rd_en))
  Row   7:         10  penable_0             ((pwrite ~| rd_en) & psel)
  Row   8:         10  penable_1             ((pwrite ~| rd_en) & psel)
```

2. *This pwdata[11:8] is not div_val: this is just an condition of TCR register to prevent write data*
   Row 5 corresponds to the input term (**pwdata[11:8] < 9) = FALSE**, meaning the condition pwdata < 9 is evaluated and is FALSE.
   However, in the RTL expression ((pwdata > 8) | ((pwdata < 9) && timer_en)), whenever pwdata ≥ 9, the left operand (pwdata > 8) becomes TRUE, and the OR operator short-circuits the right operand. As a result, the condition (pwdata < 9) is never evaluated for the FALSE case.
   When pwdata < 9, the condition is always TRUE.
   When pwdata ≥ 9, the condition is not evaluated.
   Therefore, the term (pwdata < 9)_0 is **unreachable** and Row 5 naturally has 0 hits.

```
----------------Focused Expression View-----------------
Line       94 Item    1  (pstrb[1] & ((pwdata[11:8] > 8) | ((pwdata[11:8] < 9) && timer_en)))
Expression totals: 3 of 4 input terms covered = 75.00%

          Input Term   Covered  Reason for no coverage   Hint
         -----------  --------  ----------------------  --------------
  (pwdata[11:8] < 9)         N  '_0' not hit             Hit '_0'

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  --------------------------
  Row   1:          8  pstrb[1]_0            ((pwdata[11:8] > 8) | ((pwdata[11:8] < 9) && timer_en))
  Row   2:         10  pstrb[1]_1            ((pwdata[11:8] > 8) | ((pwdata[11:8] < 9) && timer_en))
  Row   3:          6  (pwdata[11:8] > 8)_0  (pstrb[1] && ~((pwdata[11:8] < 9) && timer_en))
  Row   4:          6  (pwdata[11:8] > 8)_1  (pstrb[1] && ~((pwdata[11:8] < 9) && timer_en))
  Row   5:    ***0***  (pwdata[11:8] < 9)_0  (pstrb[1] && ~(pwdata[11:8] > 8))
  Row   6:          8  (pwdata[11:8] < 9)_1  (pstrb[1] && ~(pwdata[11:8] > 8) && timer_en)
  Row   7:          6  timer_en_0            (pstrb[1] && ~(pwdata[11:8] > 8) && (pwdata[11:8] < 9))
  Row   8:          8  timer_en_1            (pstrb[1] && ~(pwdata[11:8] > 8) && (pwdata[11:8] < 9))
```


```
----------------Focused Expression View-----------------
Line       94 Item    1  (pstrb[1] & ((pwdata[11:8] > 8) | ((pwdata[11:8] < 9) && timer_en)))
Expression totals: 3 of 4 input terms covered = 75.00%

          Input Term   Covered  Reason for no coverage   Hint
         -----------  --------  ----------------------  --------------
  (pwdata[11:8] < 9)         N  '_0' not hit             Hit '_0'

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  --------------------------
  Row   1:          8  pstrb[1]_0            ((pwdata[11:8] > 8) | ((pwdata[11:8] < 9) && timer_en))
  Row   2:         10  pstrb[1]_1            ((pwdata[11:8] > 8) | ((pwdata[11:8] < 9) && timer_en))
  Row   3:          6  (pwdata[11:8] > 8)_0  (pstrb[1] && ~((pwdata[11:8] < 9) && timer_en))
  Row   4:          6  (pwdata[11:8] > 8)_1  (pstrb[1] && ~((pwdata[11:8] < 9) && timer_en))
  Row   5:    ***0***  (pwdata[11:8] < 9)_0  (pstrb[1] && ~(pwdata[11:8] > 8))
  Row   6:          8  (pwdata[11:8] < 9)_1  (pstrb[1] && ~(pwdata[11:8] > 8) && timer_en)
  Row   7:          6  timer_en_0            (pstrb[1] && ~(pwdata[11:8] > 8) && (pwdata[11:8] < 9))
  Row   8:          8  timer_en_1            (pstrb[1] && ~(pwdata[11:8] > 8) && (pwdata[11:8] < 9))
```


```
----------Focused Expression View (Bimodal)------------
Line       96 Item    1  ((((((pstrb[1:0] == 3) && (pwdata == 0)) && wr_en) && (paddr == 0)) | ((((((~div_en && pstrb[0]) && timer_en) && (pwdata[1:0] == 0)) && wr_en)
Expression totals: 9 of 10 input terms covered = 90.00%
```

3. The FALSE case of the condition (div_val != 0) is unreachable in this expression.
The term belongs to the third OR-branch: ((timer_en && div_en) && (div_val != 0) && pulse).

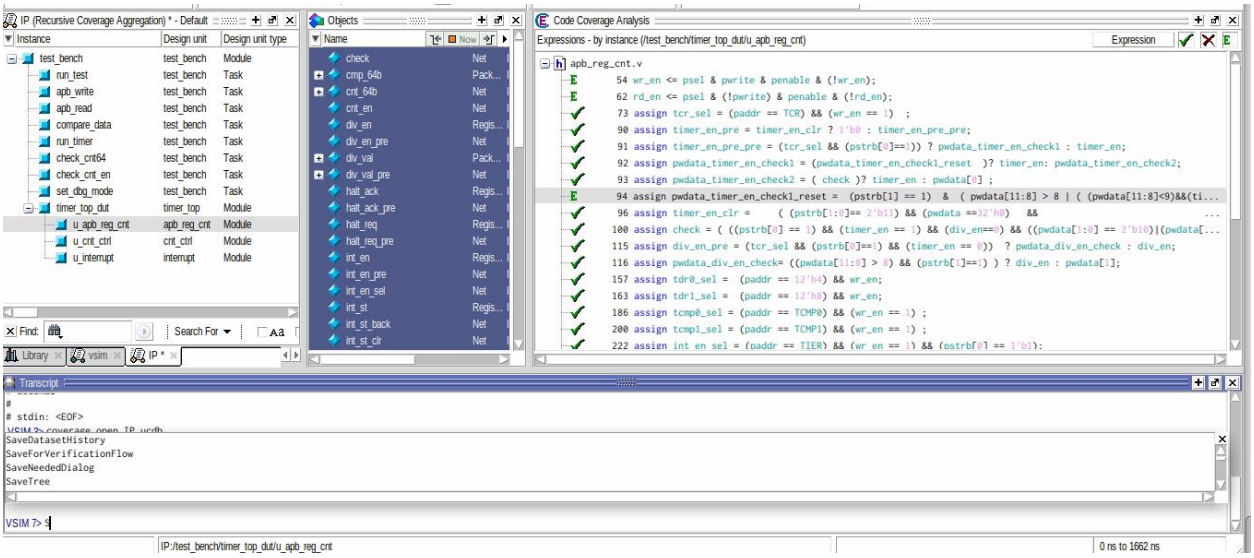
For the expression to evaluate this term in the FALSE case, all of the following must hold simultaneously:

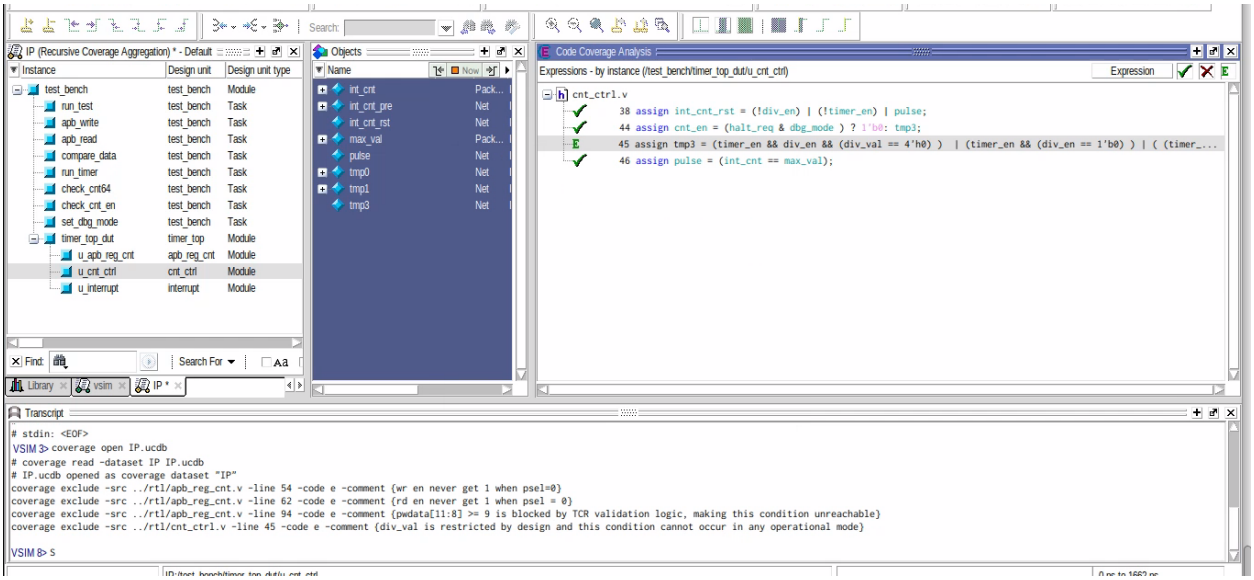
- timer_en = 1
- div_en = 1
- pulse = 1
- div_val == 0

However, pulse is only generated when div_val != 0.
Therefore, when div_val = 0, the gating condition for this expression never becomes TRUE, and the RTL never evaluates (div_val != 0) in the FALSE path.

```
================================Expression Details================================

Expression Coverage for instance /test_bench/timer_top_dut/u_cnt_ctrl --

  File ../rtl/cnt_ctrl.v


-----------Focused Expression View (Bimodal)------------
Line       45 Item    1  ((((timer_en && div_en) && (div_val == 0)) | (timer_en && ~div_en)) | (((timer_en & div_en) & (div_val != 0)) & pulse))
Expression totals: 4 of 5 input terms covered = 80.00%

       Input Term   Covered  Reason for no coverage                 Hint
       ----------  --------  --------------------------------------  --------------
    (div_val != 0)         N  '_1' hit but '_0' is not hit           Hit '_0' for output ->0

    Rows:   Hits(->0)   Hits(->1)  FEC Target            Non-masking condition(s)

---------  ----------  ----------  --------------------  ------------------------
 Row   1:          22           0  timer_en_0            (pulse && (div_val != 0) && div_en)
 Row   2:           0          16  timer_en_1            (~(((timer_en & div_en) & (div_val != 0)) & pulse) && (div_val == 0) && div_en), ~div_en, (~(((timer_en && div
 Row   3:           0          13  div_en_0              timer_en
 Row   4:           6           8  div_en_1              (~(((timer_en & div_en) & (div_val != 0)) & pulse) && (div_val == 0) && timer_en), (~(((timer_en & div_en) & (
 Row   5:           6           0  (div_val == 0)_0      (~(((timer_en & div_en) & (div_val != 0)) & pulse) && ~(timer_en && ~div_en) && (timer_en && div_en))
 Row   6:           0           6  (div_val == 0)_1      (~(((timer_en & div_en) & (div_val != 0)) & pulse) && ~(timer_en && ~div_en) && (timer_en && div_en))
 Row   7:           0           0  (div_val != 0)_0      (~(((timer_en && div_en) && (div_val == 0)) | (timer_en && ~div_en)) && pulse && (timer_en & div_en))
 Row   8:           0           6  (div_val != 0)_1      (~(((timer_en && div_en) && (div_val == 0)) | (timer_en && ~div_en)) && pulse && (timer_en & div_en))
 Row   9:           6           0  pulse_0               (~(((timer_en && div_en) && (div_val == 0)) | (timer_en && ~div_en)) && ((timer_en & div_en) & (div_val != 0))
Row   10:           0           6  pulse_1               (~(((timer_en && div_en) && (div_val == 0)) | (timer_en && ~div_en)) && ((timer_en & div_en) & (div_val != 0))
```

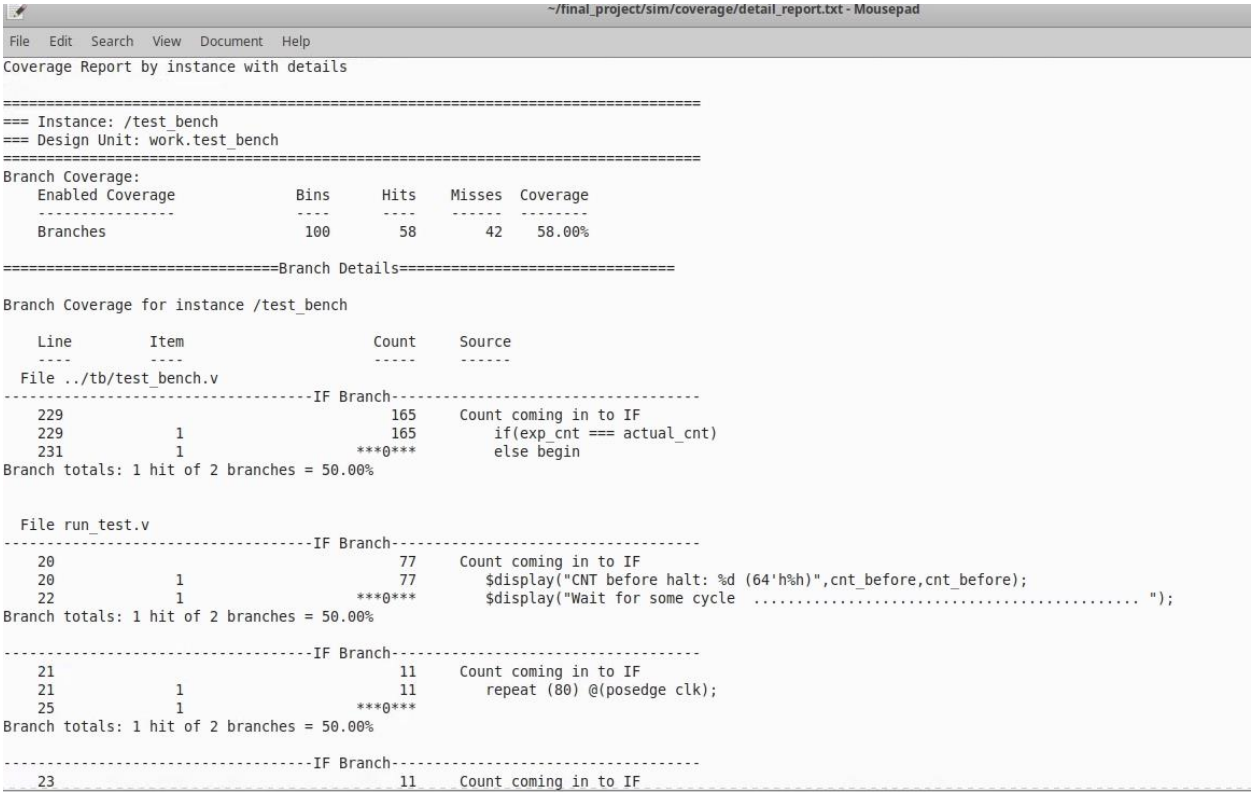
AFTER EXCLUDE:





```
Coverage Report Summary Data by instance

================================================================================
=== Instance: /test_bench/timer_top_dut/u_apb_reg_cnt
=== Design Unit: work.apb_reg_cnt
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        97        97         0   100.00%
    Conditions                      38        38         0   100.00%
    Expressions                     85        85         0   100.00%
    Statements                     101       101         0   100.00%
    Toggles                        928       928         0   100.00%

================================================================================
=== Instance: /test_bench/timer_top_dut/u_cnt_ctrl
=== Design Unit: work.cnt_ctrl
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        10        10         0   100.00%
    Conditions                       5         5         0   100.00%
    Expressions                      7         7         0   100.00%
    Statements                      12        12         0   100.00%
    Toggles                         86        86         0   100.00%

================================================================================
=== Instance: /test_bench/timer_top_dut/u_interrupt
=== 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/timer_top_dut
=== Design Unit: work.timer_top
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                        148       148         0   100.00%

================================================================================
=== Instance: /test_bench
=== Design Unit: work.test_bench
================================================================================
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                       103        58        45    56.31%
    Conditions                      25         3        22    12.00%
    Statements                     453       450         3    99.33%
    Toggles                        244       180        64    73.77%


Total Coverage By Instance (filtered view): 88.13%
```

Fig 24. Summary report - Coverage after exclude



~/final_project/sim/coverage/detail_report.txt - Mousepad

File Edit Search View Document Help

```
Coverage Report by instance with details

================================================================================
=== Instance: /test_bench
=== Design Unit: work.test_bench
================================================================================
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                       100        58        42    58.00%

================================Branch Details================================

Branch Coverage for instance /test_bench

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File ../tb/test_bench.v
------------------------------------IF Branch------------------------------------
    229                                      165     Count coming in to IF
    229             1                        165         if(exp_cnt === actual_cnt)
    231             1                     ***0***        else begin
Branch totals: 1 hit of 2 branches = 50.00%


  File run_test.v
------------------------------------IF Branch------------------------------------
    20                                       77     Count coming in to IF
    20              1                        77        $display("CNT before halt: %d (64'h%h)",cnt_before,cnt_before);
    22              1                     ***0***        $display("Wait for some cycle  ..................................... ");
Branch totals: 1 hit of 2 branches = 50.00%

------------------------------------IF Branch------------------------------------
    21                                       11     Count coming in to IF
    21              1                        11        repeat (80) @(posedge clk);
    25              1                     ***0***
Branch totals: 1 hit of 2 branches = 50.00%

------------------------------------IF Branch------------------------------------
    23                                       11     Count coming in to IF
```

Fig 25. Detail report - Coverage after exclude