**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](#_Toc215349862)

[1.1. Introduction 4](#_Toc215349863)

[1.2. Main features 4](#_Toc215349864)

[1.3. Design Proposal 9](#_Toc215349865)

[1.3. Block diagram 12](#_Toc215349866)

[1.4. Interface signals 22](#_Toc215349867)

[2.Register Specification 26](#_Toc215349868)

[2.1. Register Summary 26](#_Toc215349869)

[2.2. Timer Control Register (TCR) 26](#_Toc215349870)

[2.3. Timer Data Register 0 (TDR0) 27](#_Toc215349871)

[2.4. Timer Data Register 1 (TDR1) 28](#_Toc215349872)

[2.5. Timer Compare Register 0 (TCMP0) 28](#_Toc215349873)

[2.6. Timer Compare Register 1 (TCMP1) 29](#_Toc215349874)

[2.7. Timer Interrupt Enable Register (TIER) 29](#_Toc215349875)

[2.8. Timer Interrupt Status Register (TISR) 30](#_Toc215349876)

[2.9. Timer Halt Control Status Register (THCSR) 31](#_Toc215349877)

[3. Functional Description 32](#_Toc215349878)

[3.1. APB slave – Register - Counter 32](#_Toc215349879)

[3.2. Counter Control 33](#_Toc215349880)

[3.3. Interrupt 34](#_Toc215349881)

[4. Sample Waveform 35](#_Toc215349882)

[5. VPLAN 36](#_Toc215349883)

[6. URL FILE CODE 38](#_Toc215349884)

[RTL CODE 38](#_Toc215349885)

[TESTCASE 38](#_Toc215349886)

[7. RESULT 39](#_Toc215349887)

[RESULT WITH TESTBENCH + TESTCASES 39](#_Toc215349888)

[RESULT WITH GOLDEN MODEL 40](#_Toc215349889)

[COVERAGE 41](#_Toc215349890)

# 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.

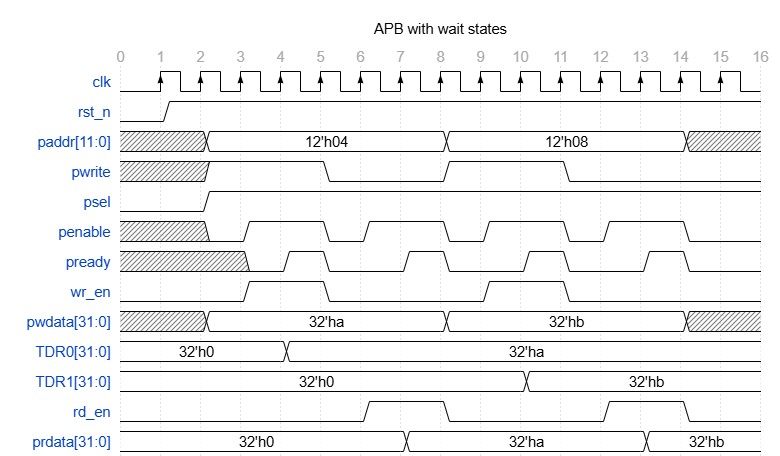


Fig 1. APB with wait states

**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:

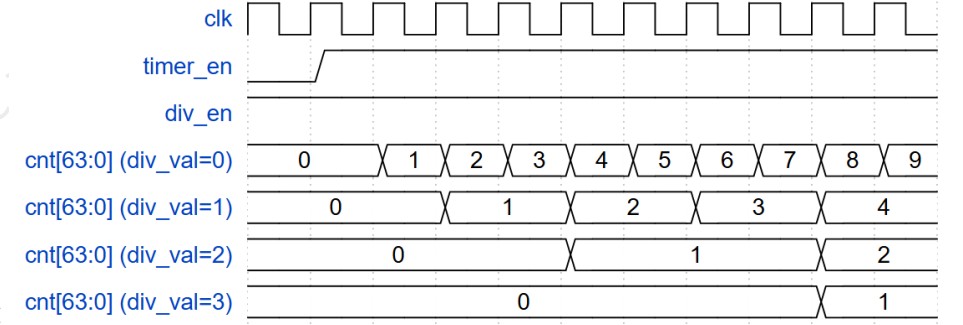


Fig 2. Counter control mode with different div\_val

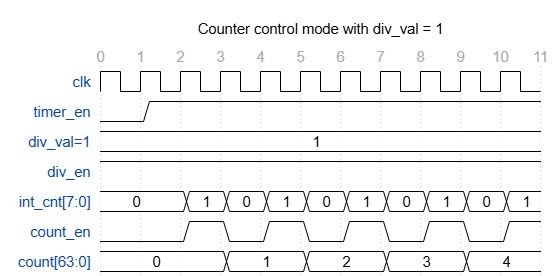


Fig 3. Counter control mode with div\_val = 1

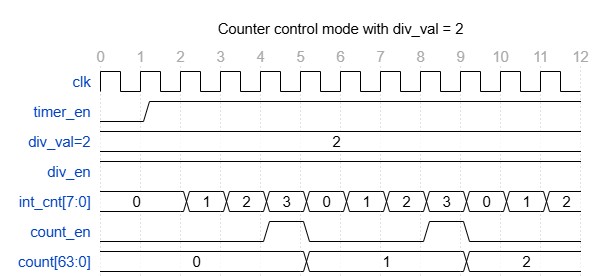


Fig 4. Counter control mode with div\_val = 2

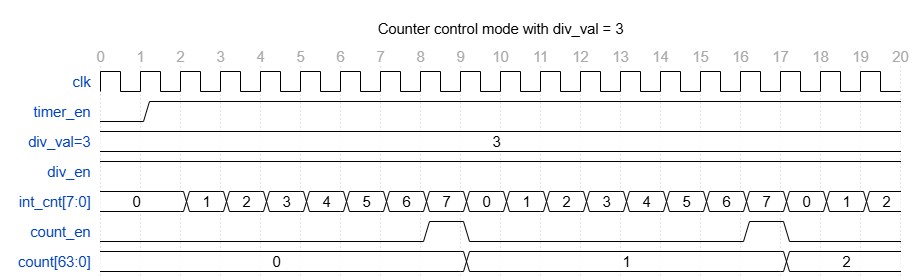


Fig 5. Counter control mode with div\_val = 3

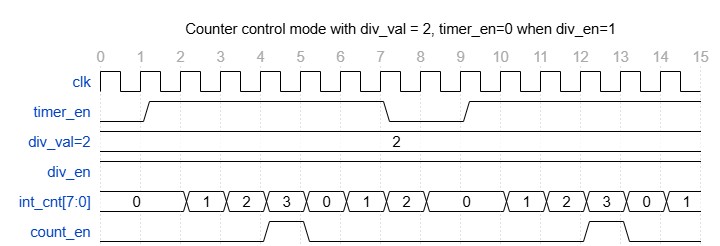


Fig 6. Counter control mode with div\_val = 2, timer\_en=0 when div\_en=1

* **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.

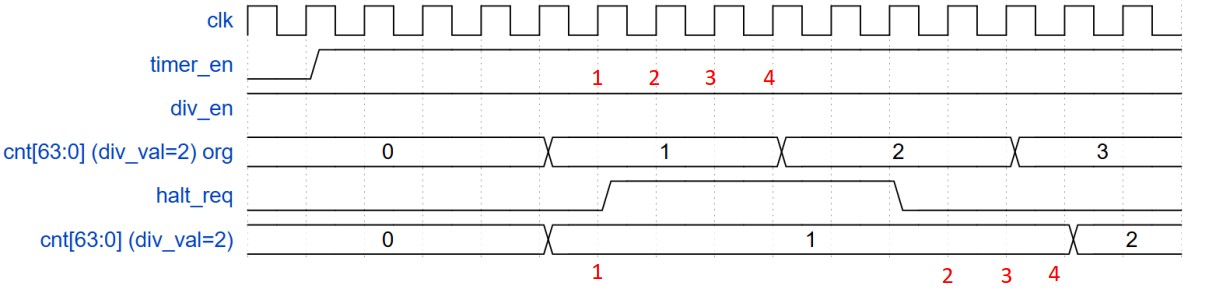


Fig 7. Counter is halted

* **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.

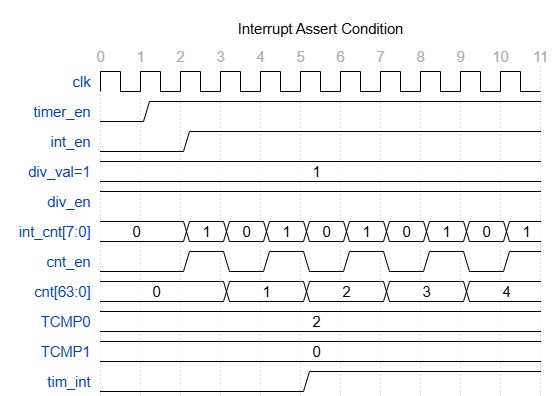


Fig 8. Interrupt assert condition

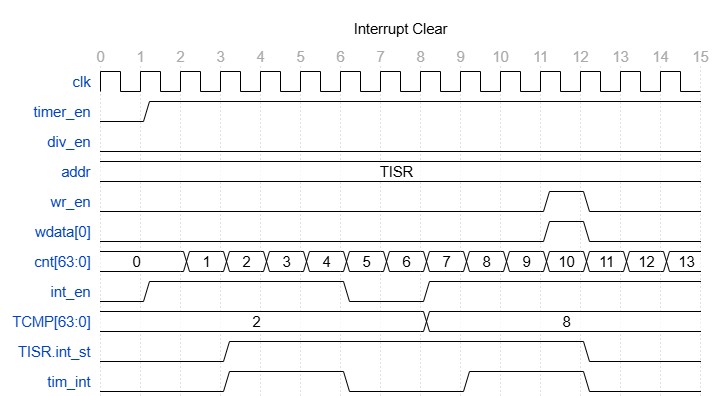


Fig 9. Interrupt clear

## 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 |
| 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

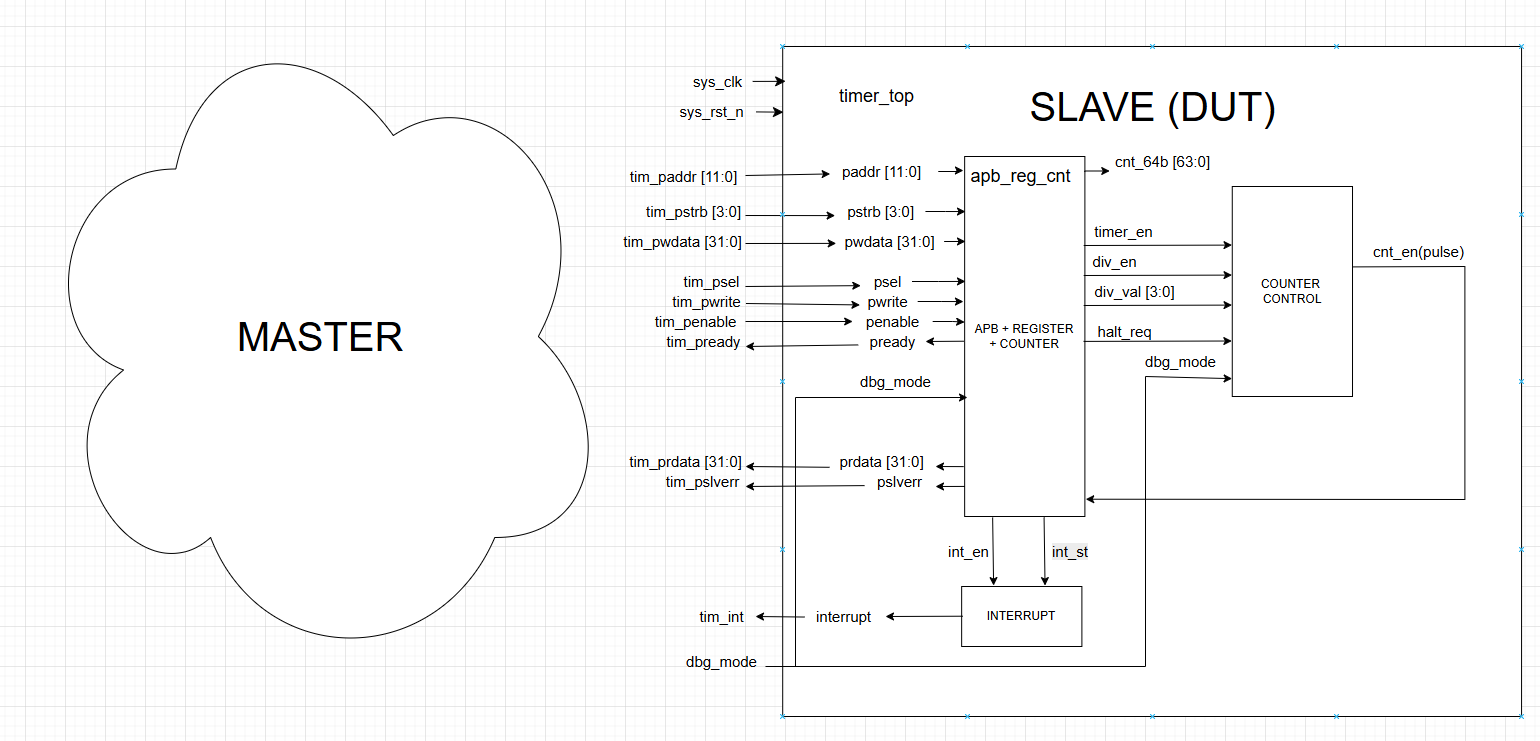
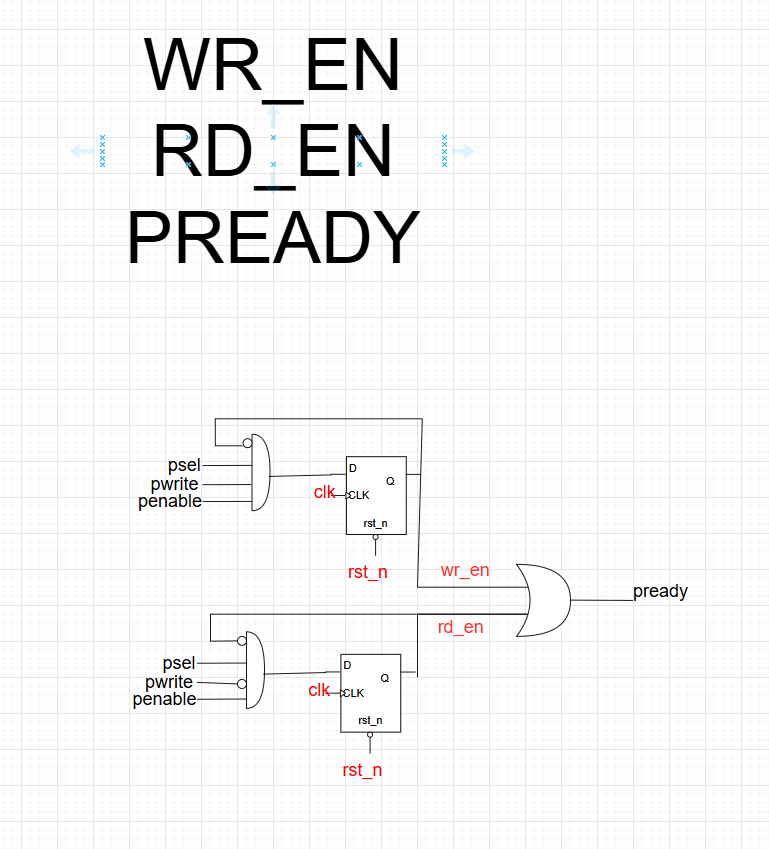


Fig 10. Block diagram of the DUT

*• 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*

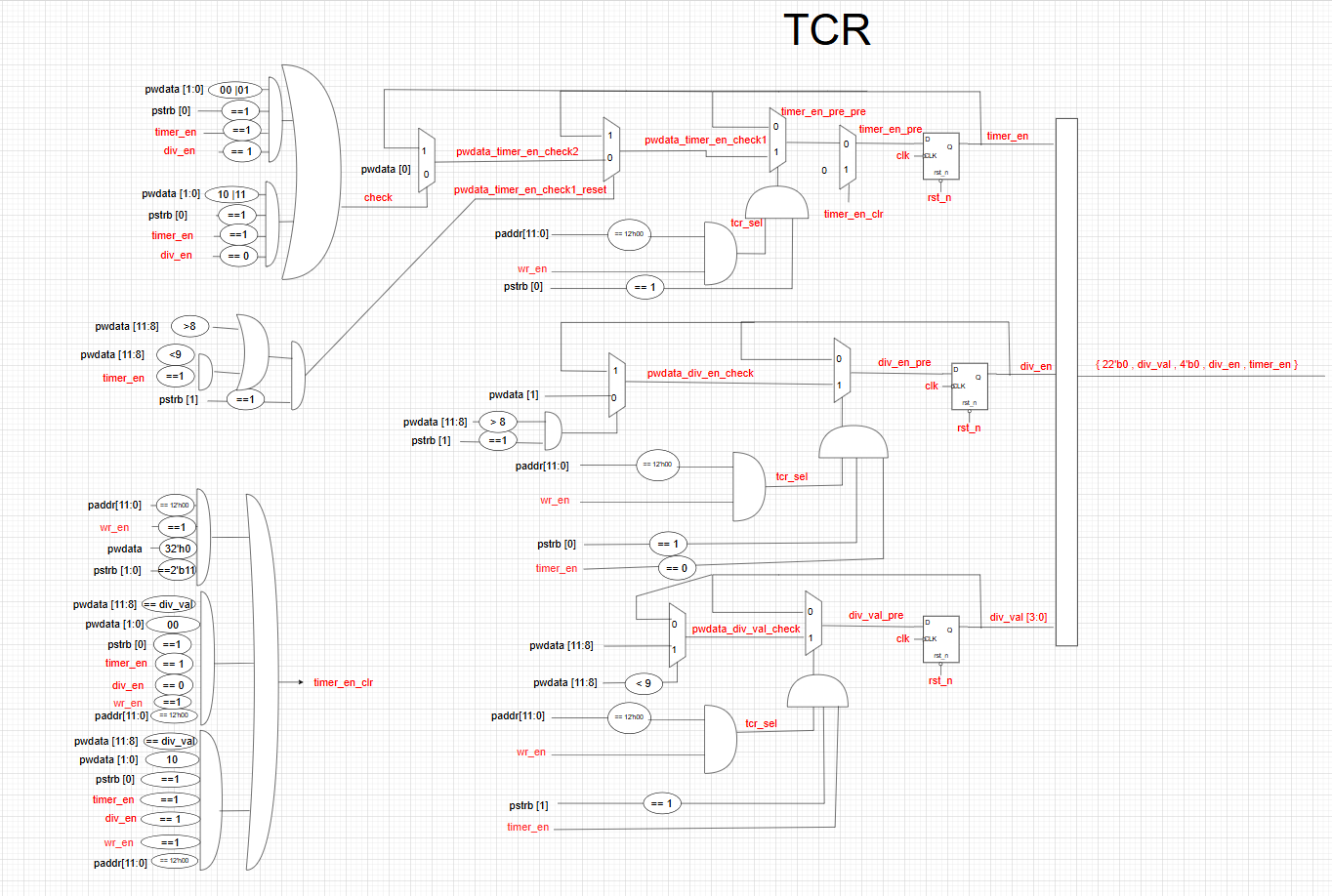


Fig 12. – Timer Control Register (TCR) Logic

*• Handles timer\_en, div\_en, div\_val  
• Protects divider values when running  
• Invalid writes flagged by PSLVERR*

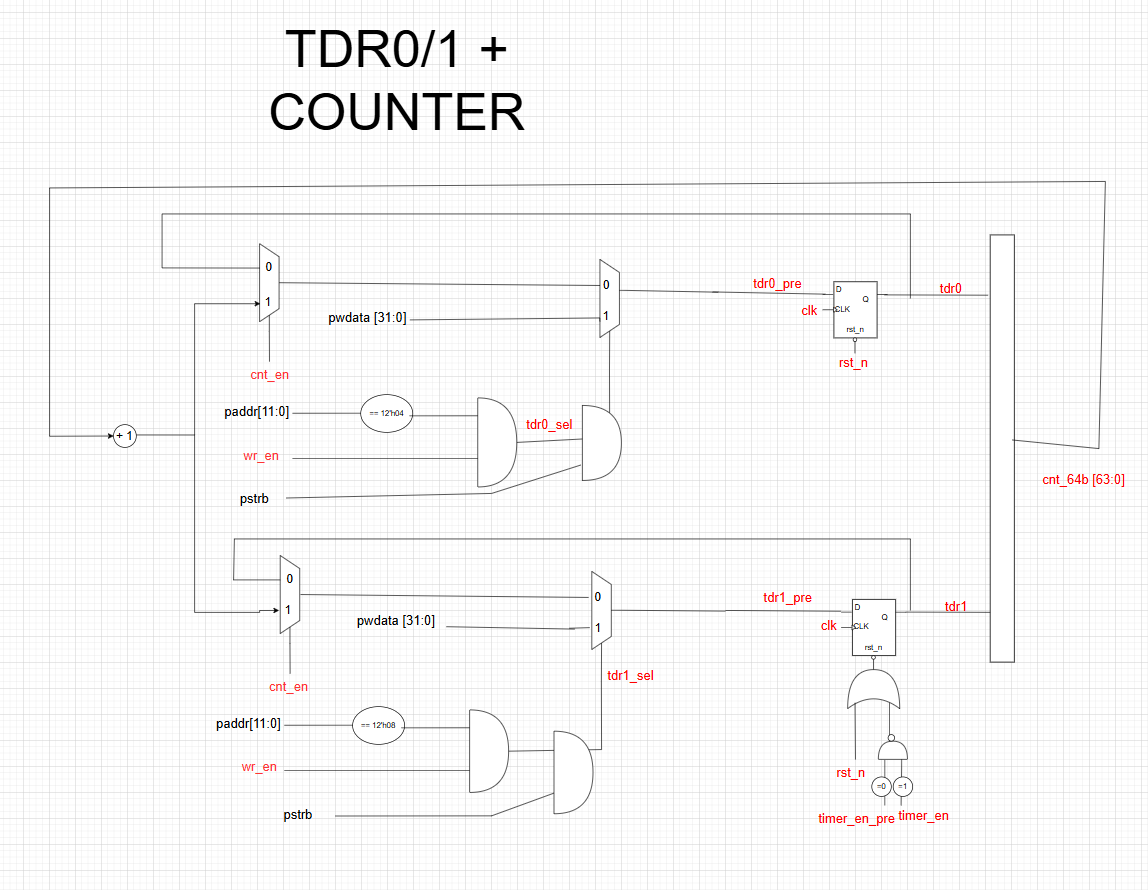


Fig 13. TDR0/TDR1 and 64-bit Counter

*• TDRs mirror CNT64 for readback  
• Counter reload on timer\_en   
• Counter reset on timer\_en high to low*

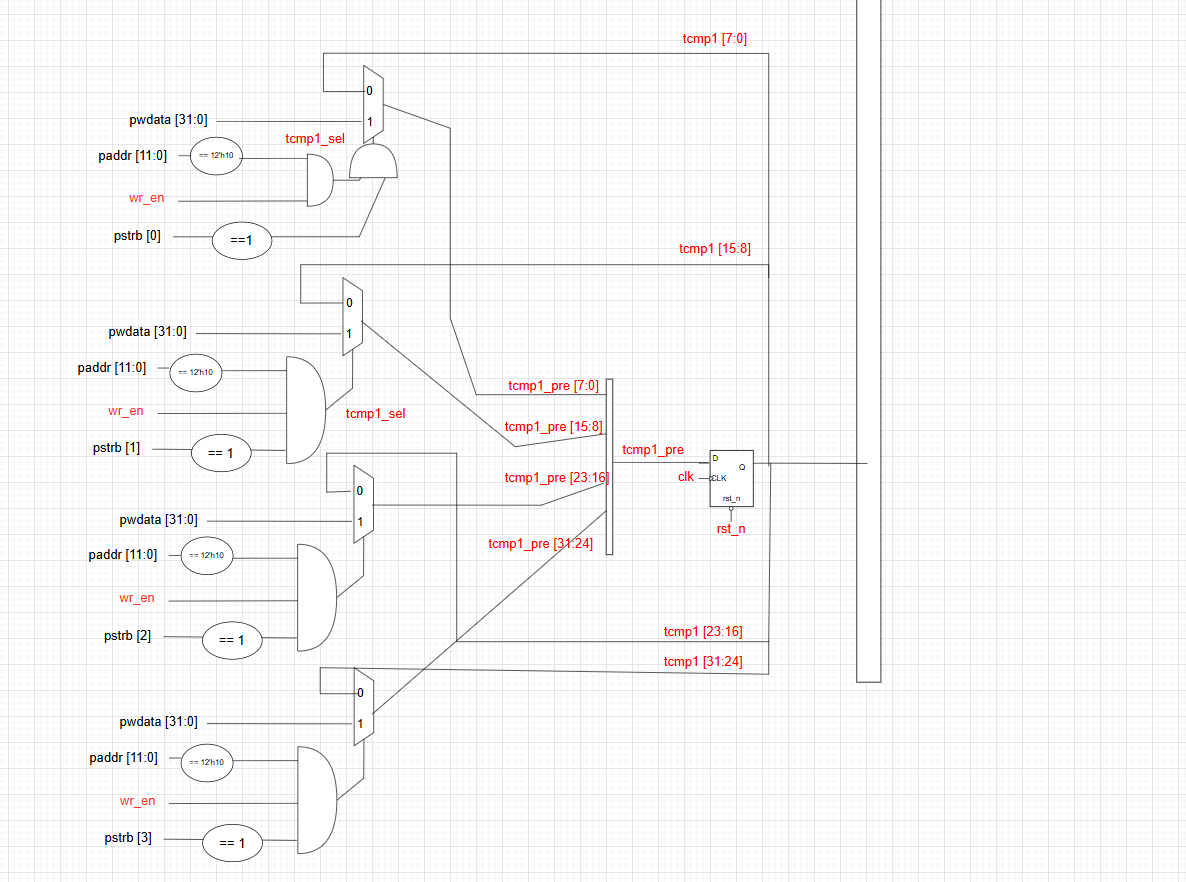
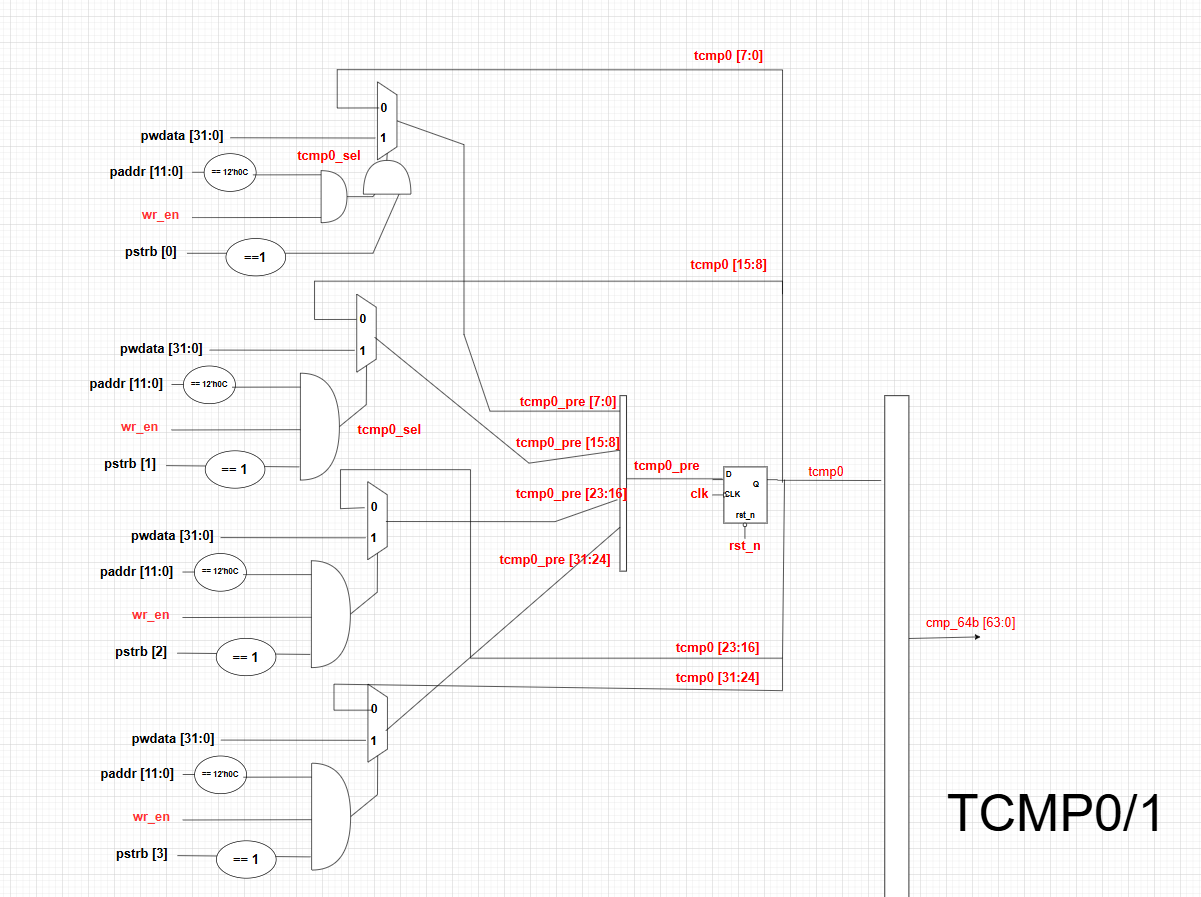


Fig 14. Compare Registers (TCMP0/TCMP1)

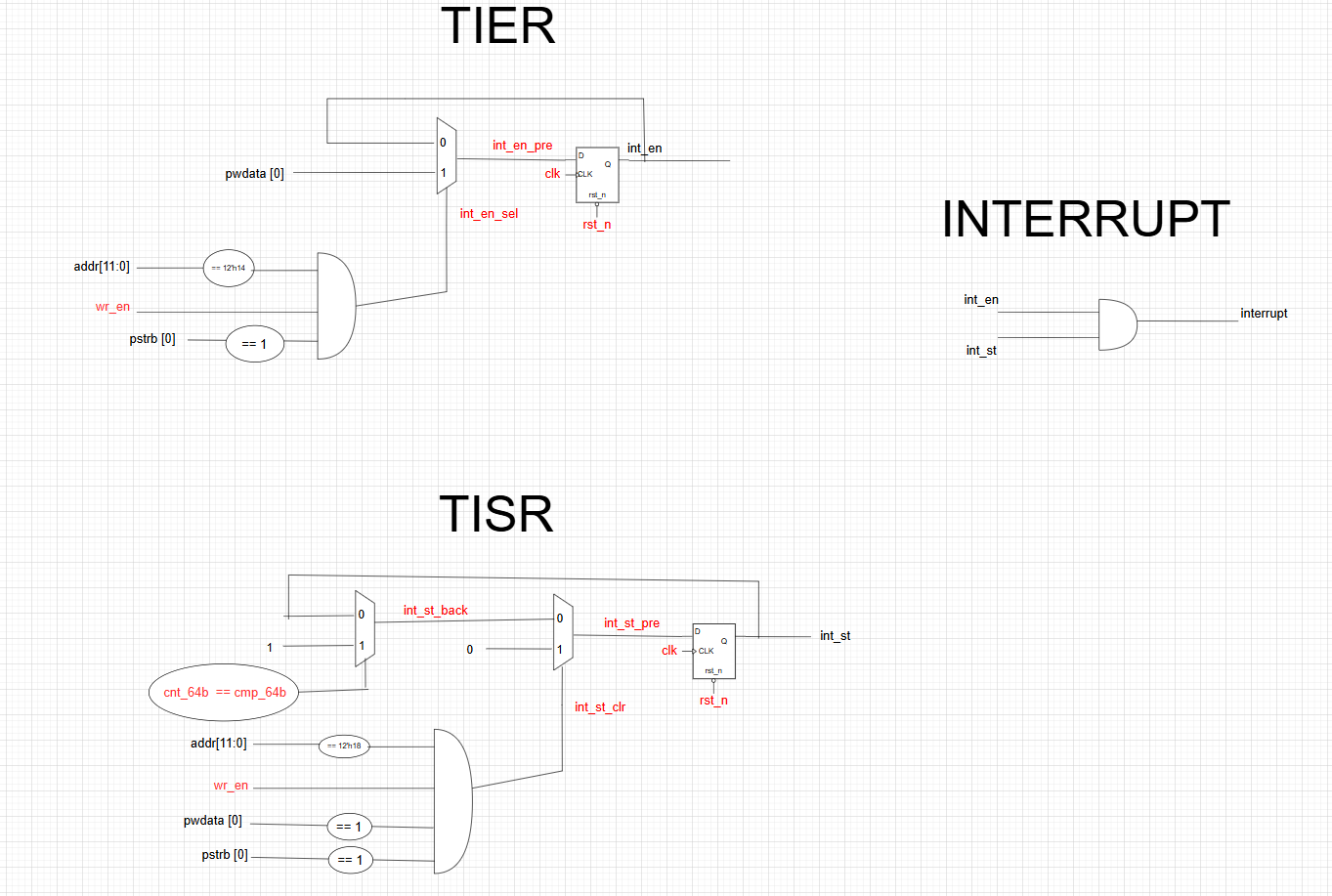


Fig 15. Interrupt Logic (TIER, TISR in Register) & Interrupt Block

*• 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*

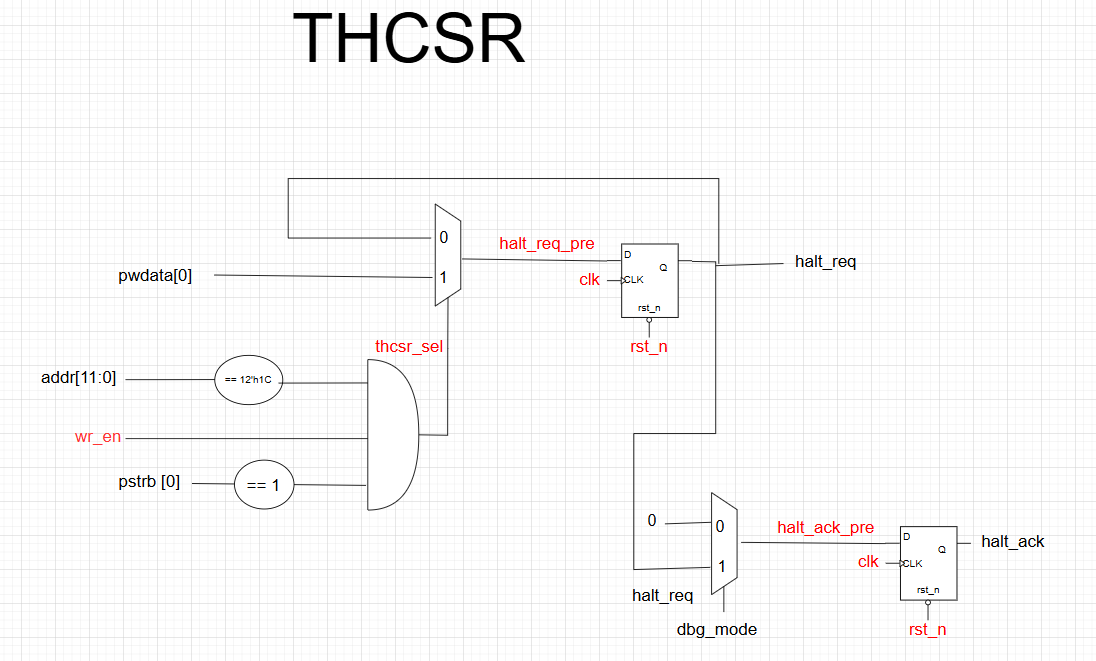


Fig 16. Halt Control Register (THCSR)

*• Works only when dbg\_mode = 1  
• Freeze counter + assert halt\_ack*

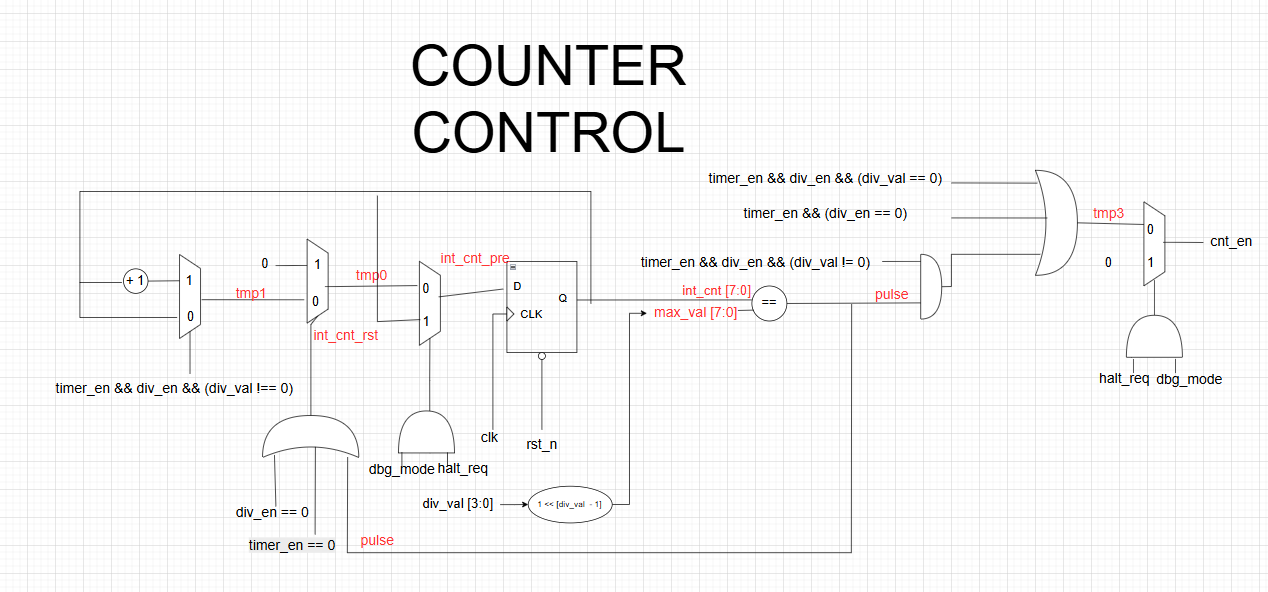


Fig 17. Counter Control Module

*• Generates cnt\_en  
• Normal mode or divided mode  
• Prescaler resets when timer disabled*

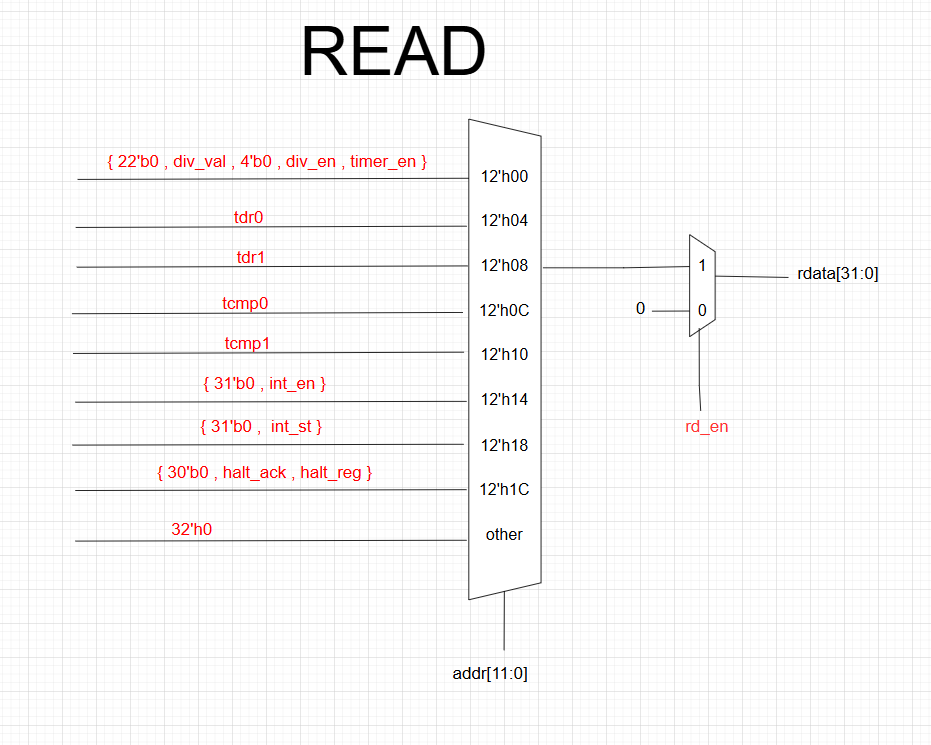


Fig 18. Register Read Path

*• 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. |
| 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**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | DIV\_VAL[3:0] | | | | Reserved | | | | | | DIV\_  EN | TIM\_  EN |
|  |  |  |  | rw | | | |  |  |  |  |  |  | rw | rw |

Table 7. Timer Control Register (TCR)

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| cnt\_64b[31:16] | | | | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
| cnt\_64b[15:0] | | | | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

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**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| cnt\_64b[63:48] | | | | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
| cnt\_64b[47:32] | | | | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

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**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| cmp\_64b[31:16] | | | | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
| cmp\_64b[15:0] | | | | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

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**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| cmp\_64b[63:48] | | | | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
| cmp\_64b[47:32] | | | | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | | | | | | | | Int\_en |
|  |  |  |  | rw | | | |  |  |  |  |  |  | rw | rw |

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**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | | | | | | | | Int\_st |
|  |  |  |  | rw | | | |  |  |  |  |  |  | rw | rw |

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**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | | | | | | | halt\_ack | halt\_req |

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 2⁶⁴).

**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

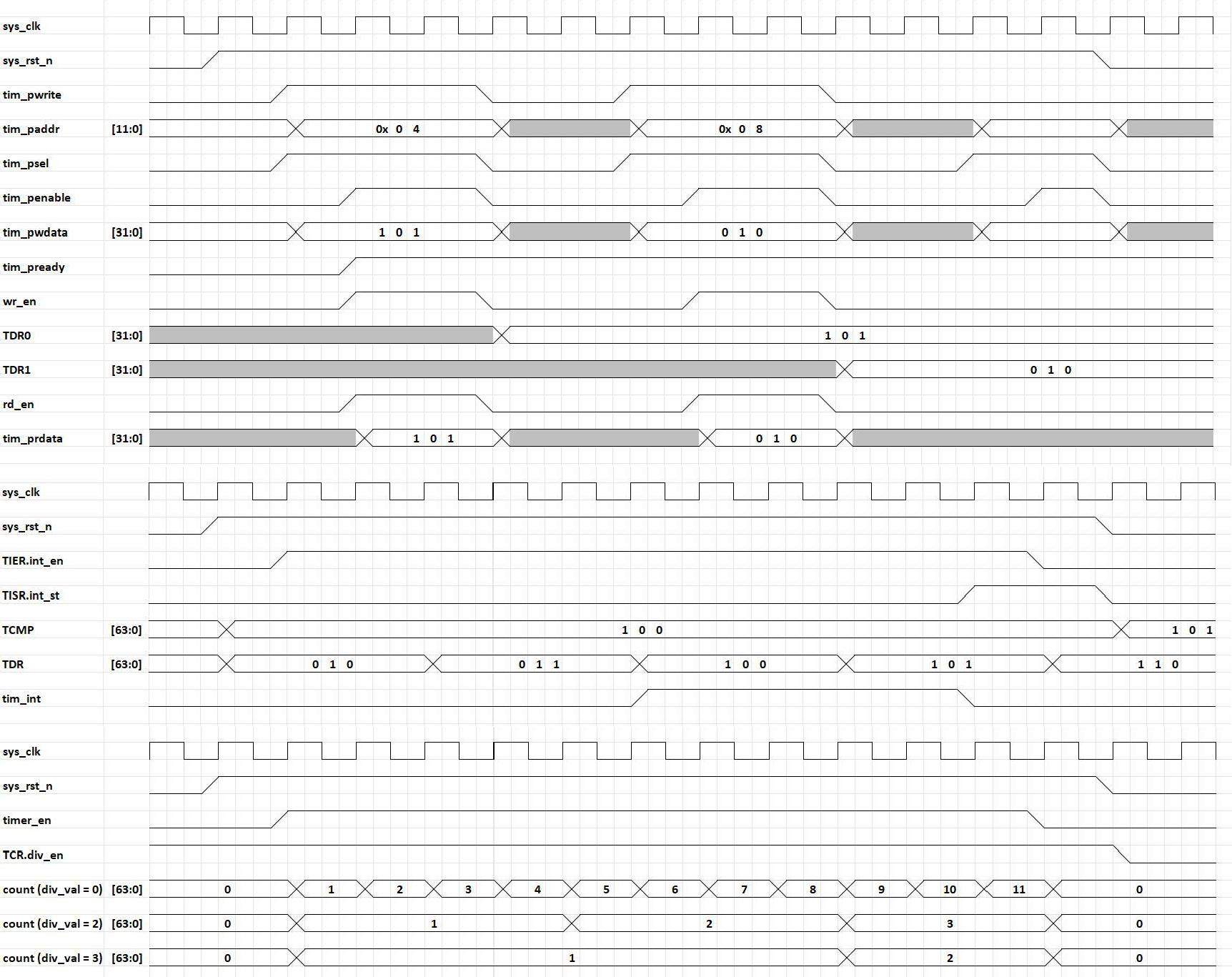
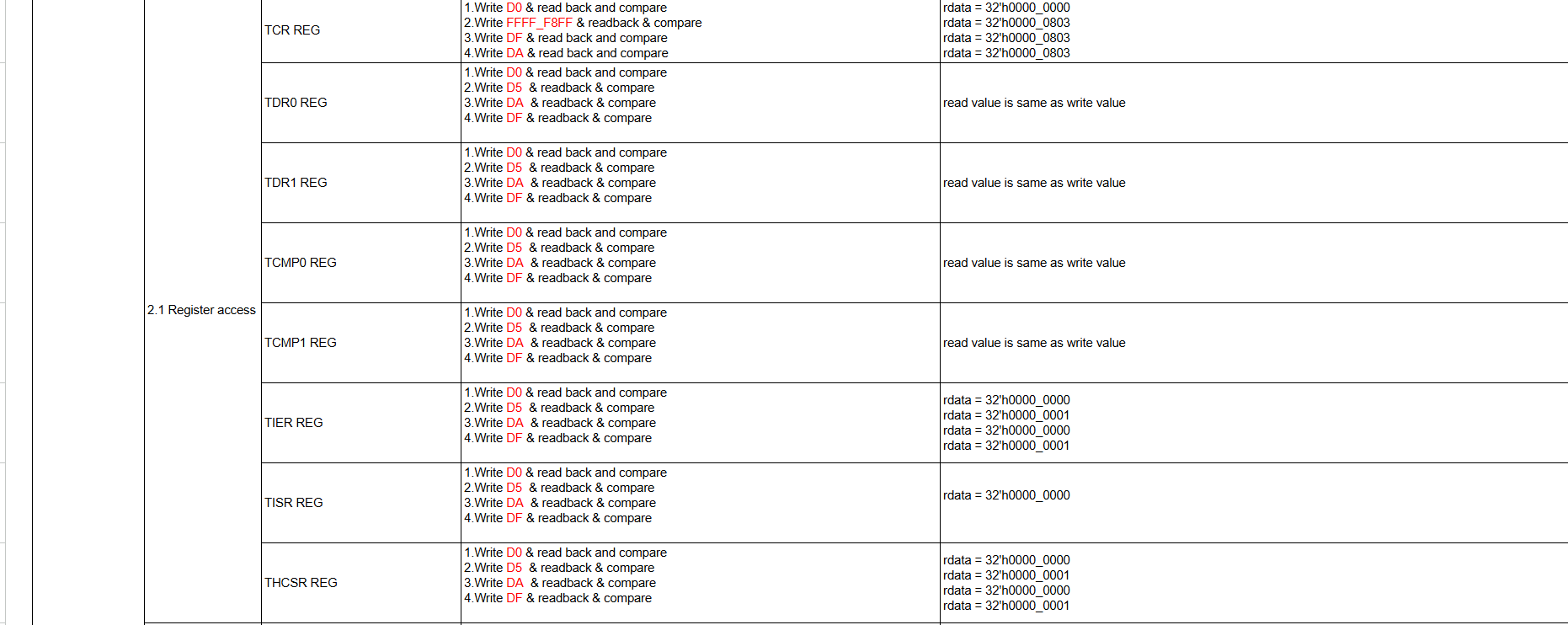
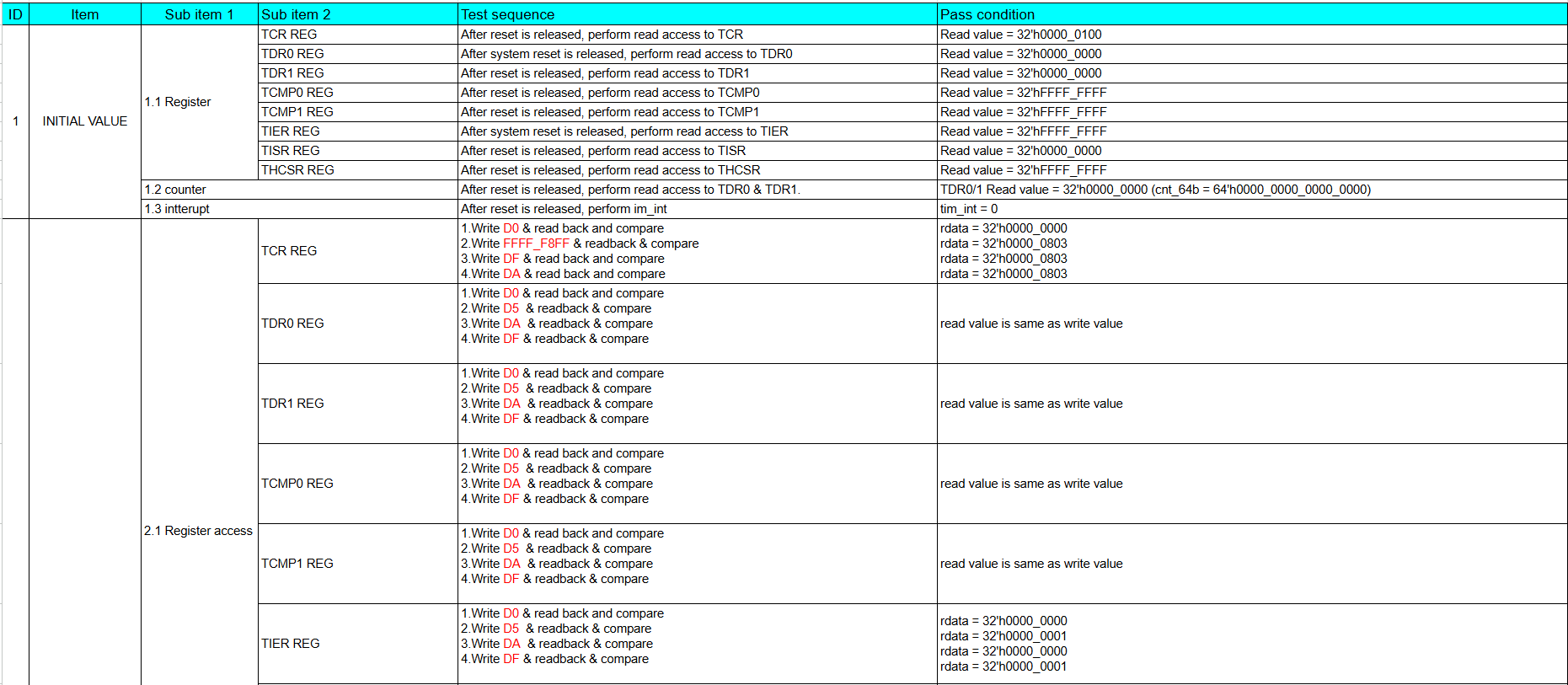
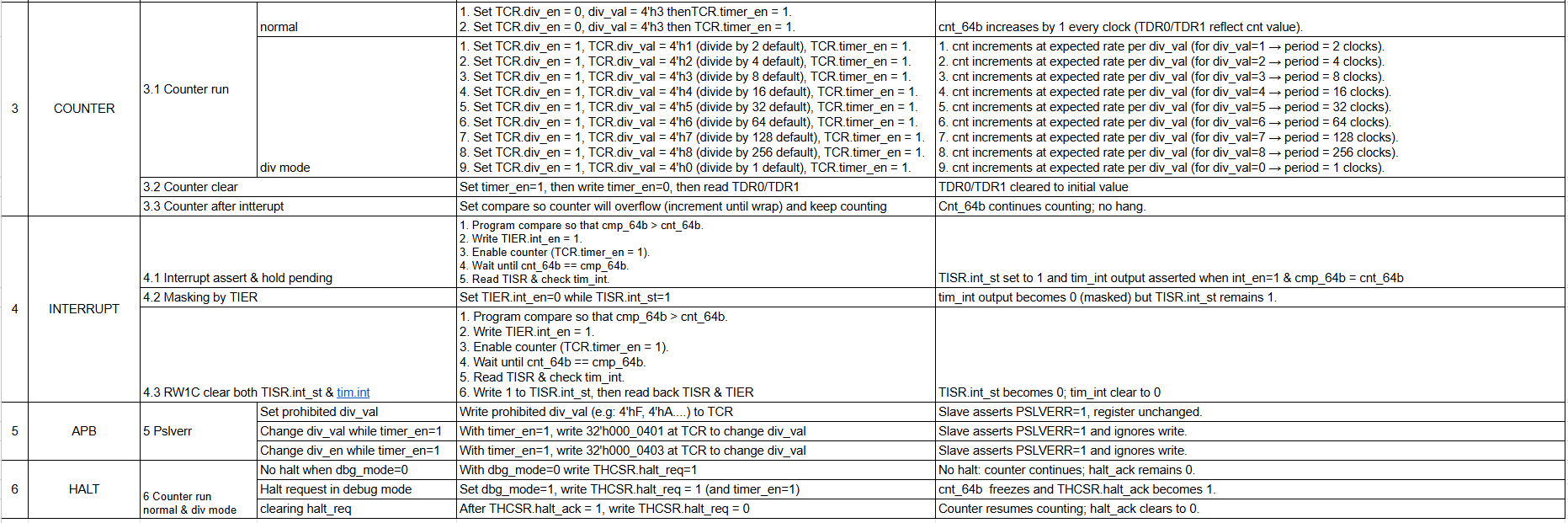
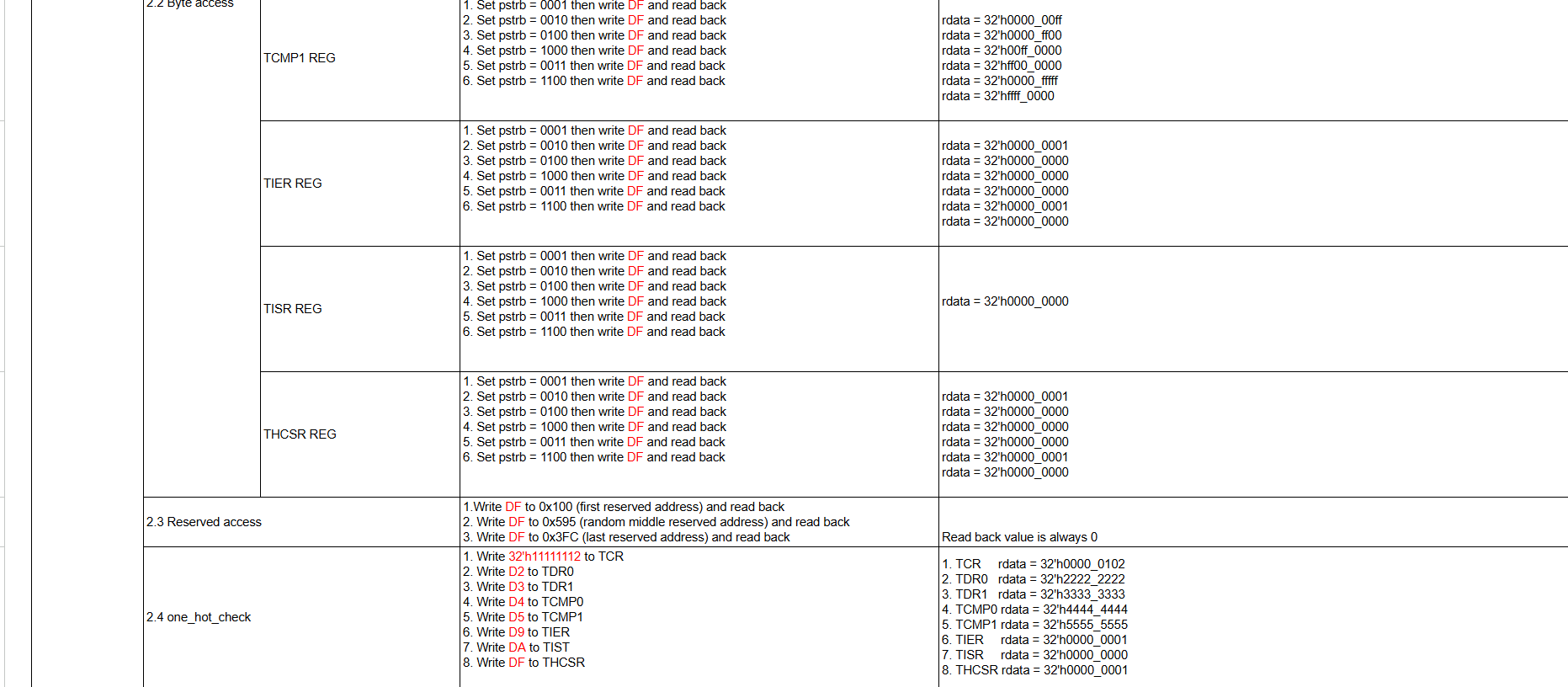


Fig 20. Sample Waveform

# 5. VPLAN

Table 15. Vplan





# 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

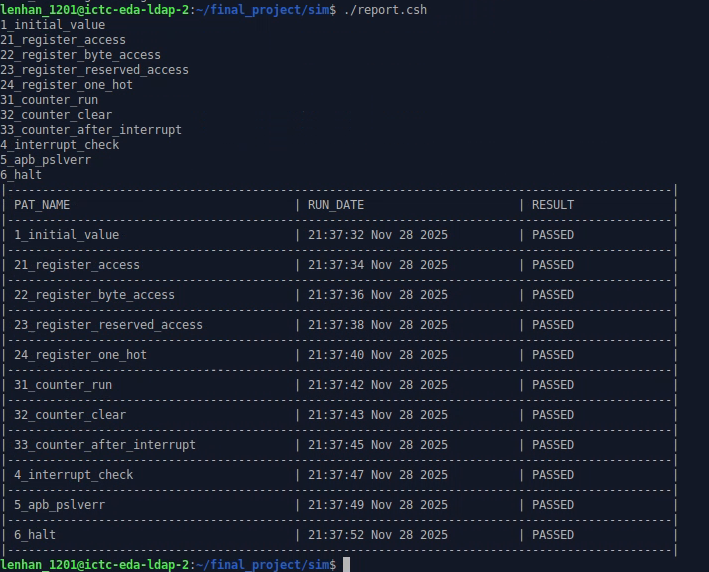
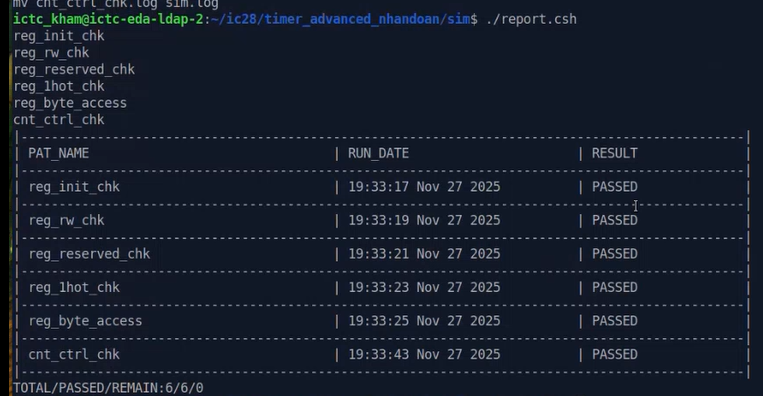


Fig 21. Testcase Report

## RESULT WITH GOLDEN MODEL

****

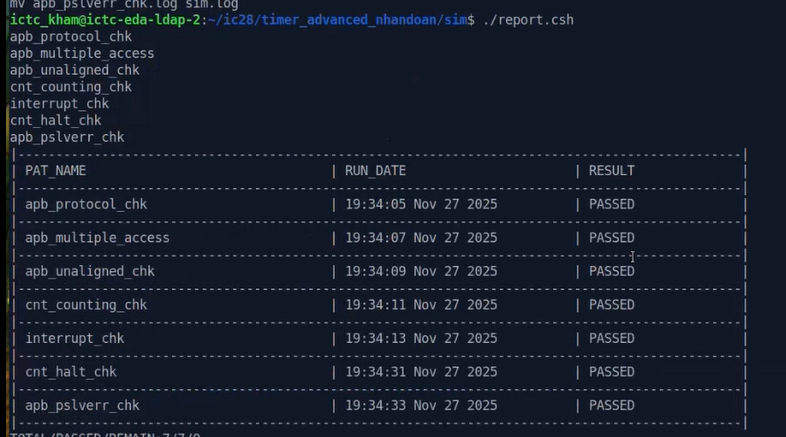
****

Fig 22. Golden Model Report

## COVERAGE

BEFORE EXCLUDE:

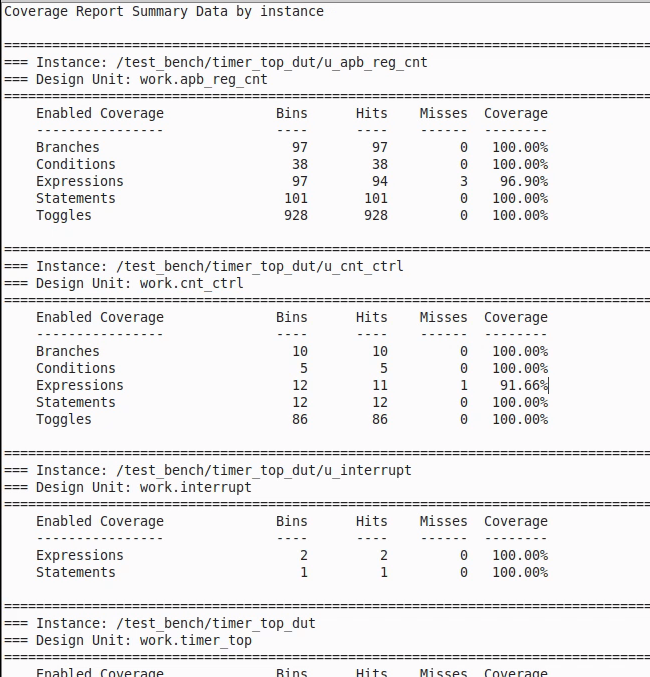
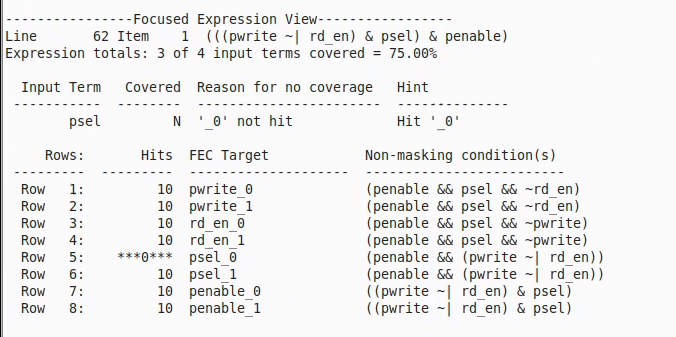
****

Fig.23 Sumary coverage report

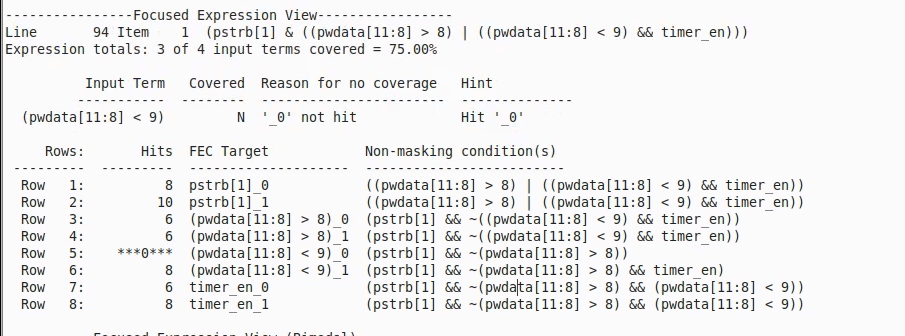
Testbench never have 100% coverage because :

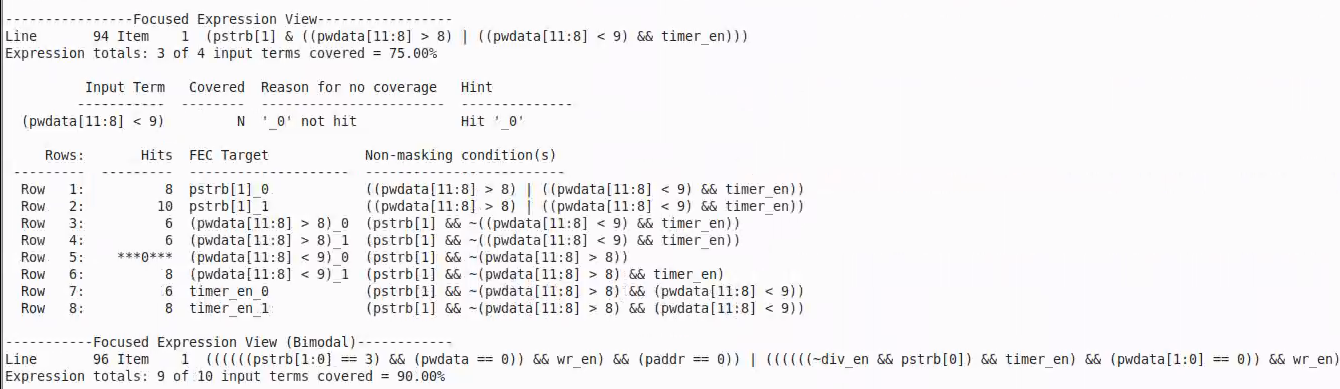
* + 1. wr\_en & rd\_en never asserts when PSEL=0 
    2. *This pwdata[11:8] is not div\_val: this is just an condition of TCR register to prevent write data*

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.



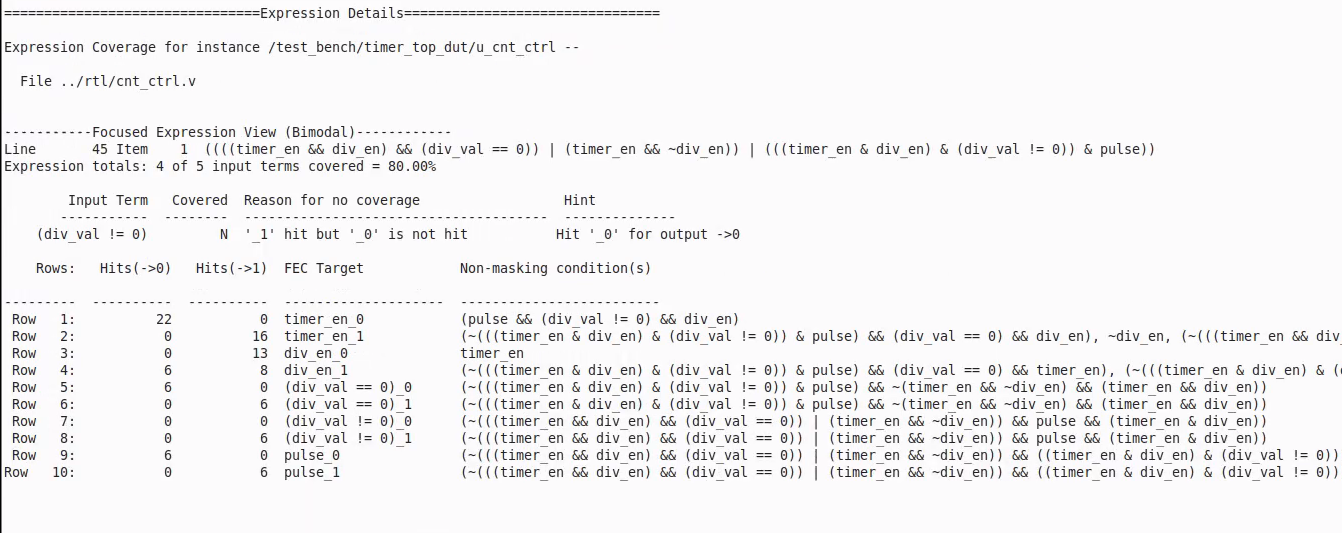


* + 1. The FALSE case of the condition (div\_val != 0) is unreachable in this expression.  
       The term belongs to the third OR-branch: ((timer\_en && div\_en) && (div\_val != 0) && pulse).

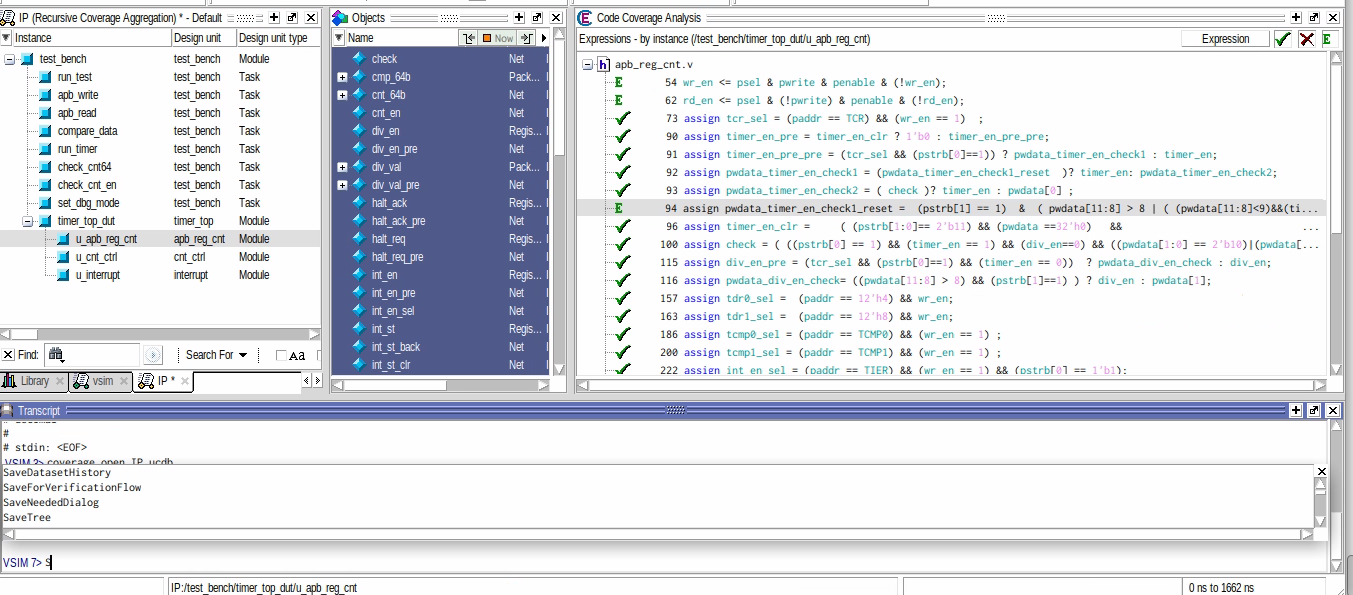
For the expression to evaluate this term in the FALSE case, all of the following must hold simultaneously:

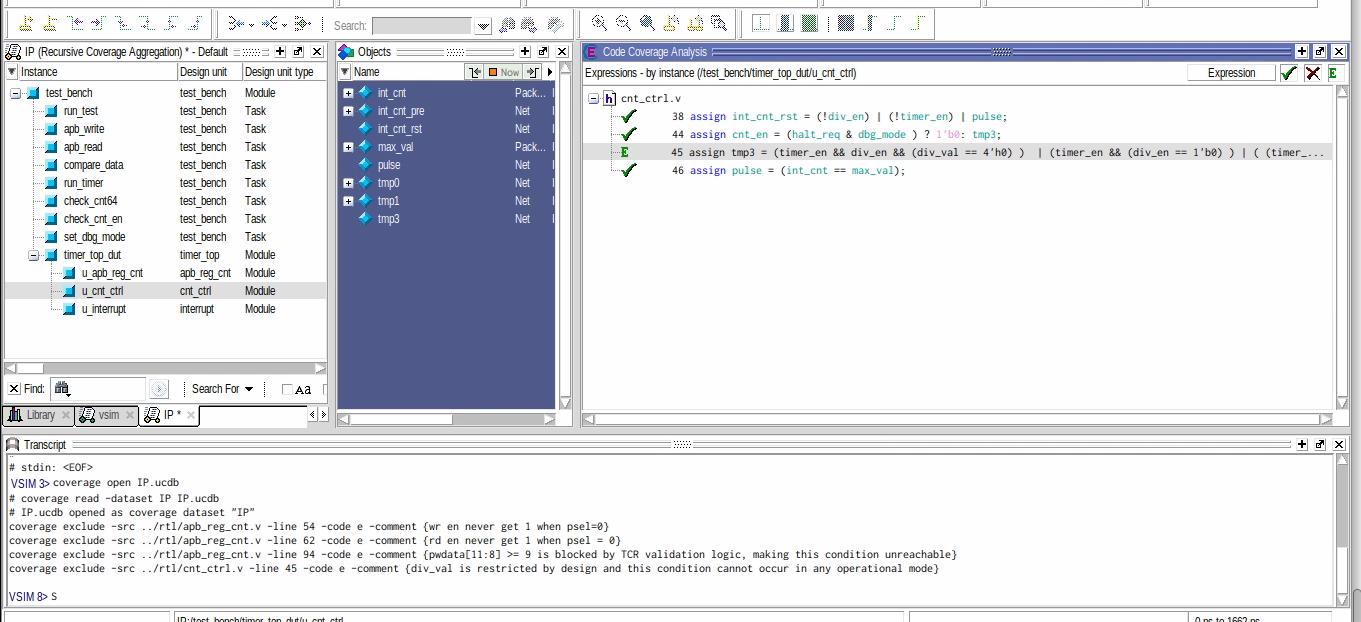
• timer\_en = 1  
• div\_en = 1  
• pulse = 1  
• div\_val == 0

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.



AFTER EXCLUDE:





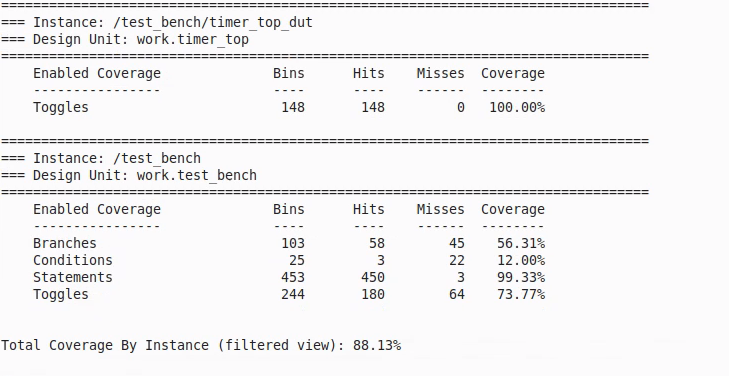
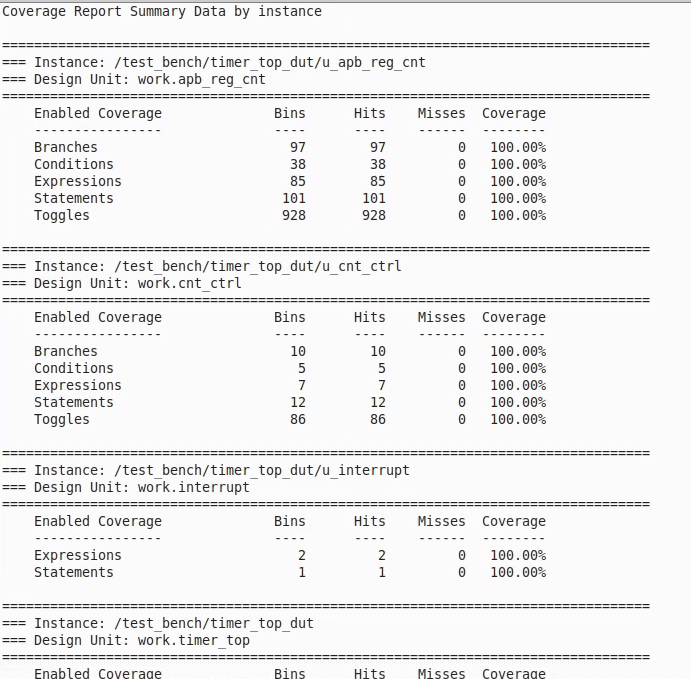


Fig 24. Summary report - Coverage after exclude

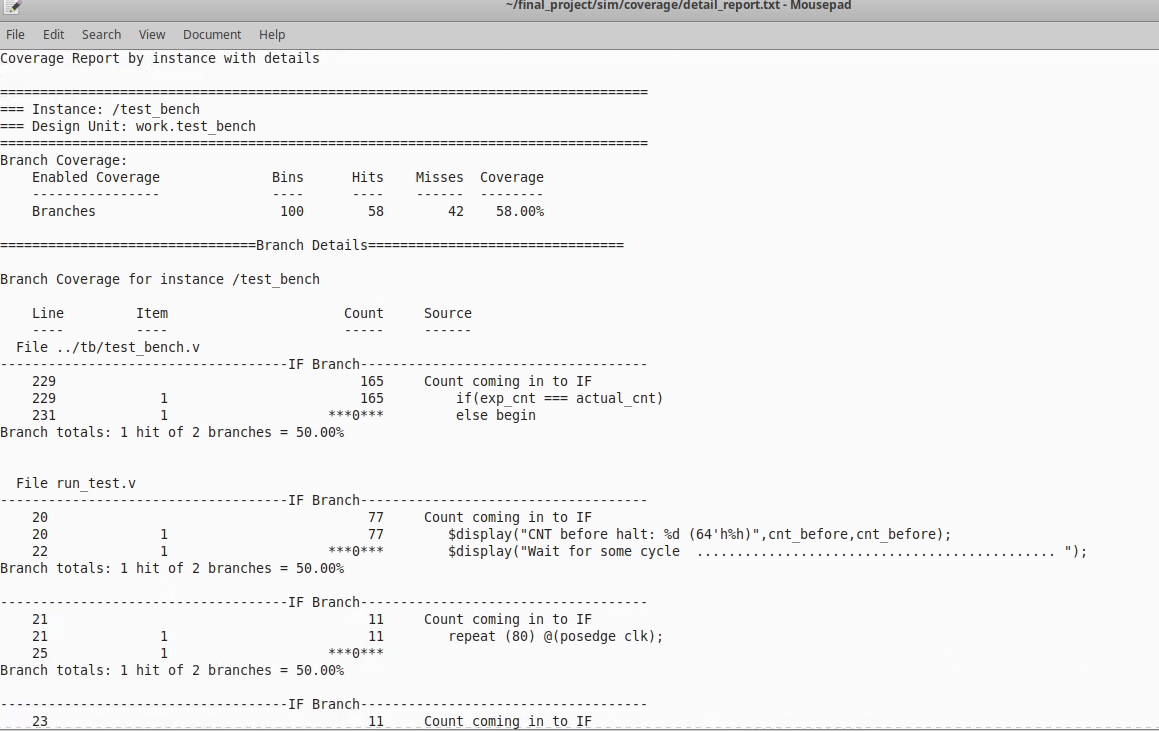


Fig 25. Detail report - Coverage after exclude