

**IC TRAINING CENTER VIET NAM**  
**FUNDAMENTAL IC DESIGN AND VERIFICATION COURSE**



**FINAL PROJECT**  
**TIMER IP DESIGN SPECIFICATION**

**Student: Nguyen Dinh Khanh Vy**

**Class: IC24**

**Instructor: Mr. Long**

# Contents

LIST OF TABLES.....	4
LIST OF FIGURES.....	5
1. Overview .....	6
1.1. Introduction .....	6
1.2. Main features.....	6
1.2.1. Counter .....	6
1.2.2. Halted .....	7
1.2.3. Timer Interrupt .....	7
1.2.4. Counting Mode .....	7
1.2.5. APB slave/ Register .....	7
1.3. Block diagram.....	8
1.3.1. APB slave.....	8
1.3.2. Register & Interrupt.....	8
1.3.3. Counter Control .....	8
1.3.4. Counter .....	9
1.4. Interface signals .....	9
2. Register Specification .....	10
2.1. Register Summary .....	10
2.2. Timer Control Register (TCR) .....	10
2.3. Timer Data Register 0 (TDR0) .....	12
2.4. Timer Data Register 1 (TDR1) .....	12
2.5. Timer Compare Register 0 (TCMP0) .....	13
2.6. Timer Compare Register 1 (TCMP1) .....	13
2.7. Timer Interrupt Enable Register (TIER).....	14
2.8. Timer Interrupt Status Register (TISR) .....	15
2.9. Timer Halt Control Status Register (THCSR).....	15
3. Functional Description.....	16

3.1. APB slave .....	16
3.2. Register & Interrupt .....	17
3.3. Counter control .....	23
3.4. Counter .....	25
4. Verify.....	27
4.1. Vplan.....	27
4.2. Coverage .....	27
5. History.....	27

## LIST OF TABLES

Table 1: Interface signals of timer IP .....	9
Table 2: Register Summary .....	10
Table 3. Timer Control Register (TCR) .....	10
Table 4. Timer Data Register 0 (TDR0).....	12
Table 5. Timer Data Register 1 (TDR1).....	13
Table 6. Timer Compare Register 0 (TCMP0).....	13
Table 7. Timer Compare Register 1 (TCMP1).....	14
Table 8. Timer Interrupt Enable Register (TIER).....	14
Table 9. Timer Interrupt Status Register (TISR).....	15
Table 10. Timer Halt Control Status Register (THCSR).....	16

## LIST OF FIGURES

Figure 1: Block diagram of Timer IP.....	8
Figure 2: Waveform of APB Slave.....	16
Figure 3: Logic diagram of APB Slave.....	17
Figure 4: Waveform of TCR register .....	18
Figure 5: Waveform of TIER register and TISR register .....	18
Figure 6: Logic diagram of the address decoder .....	19
Figure 7: Logic diagram of the pslverr signal .....	19
Figure 8: Logic diagram of TCR Register .....	20
Figure 9: Logic diagram of TIER Register, TISR Register, tim_int .....	21
Figure 10: Logic diagram of TCMP0 Register .....	22
Figure 11: Waveform of Counter control .....	23
Figure 12: Logic diagram of Counter control .....	24
Figure 13: Waveform of Counter .....	25
Figure 14: Logic diagram of Counter .....	26
Figure 15: The coverage achieved is 100% .....	27

# 1. Overview

## 1.1. Introduction

- 64-bit count-up
- 12-bit address
- Register set is configured via APB bus (IP is APB slave)
- Support wait state (1 cycle is enough) and error handling
- Support byte access
- Support halt (stop) in debug mode
- Timer uses active low async reset
- Counter can be counted based on the system clock or divided up to 256
- Support timer interrupt (can be enabled or disabled)

## 1.2. Main features

### 1.2.1. Counter

- 64-bit count-up
- Counting mode:
  - + Default mode: counter's speed is the same as the system clock
  - + Control mode: when enabled by writing 1 to the TCR.div\_en bit, the counter's speed is determined by the divisor value set in TCR.div\_val
- Counter continues counting when interrupts occur.
- Counter continues counting when overflow occurs.
- Support halted mode
- When timer\_en changes from High to Low, the counter is cleared to its initial value. When timer\_en is L->H again, the timer can work normally.
- Note: the div\_en and div\_val are not related to the frequency divisor (clock divider). Those settings only control when the counter counts.

### 1.2.2. Halted

- Counter can be halted (stopped) in debug mode when both below conditions occur: Input debug\_mode signal is High, THCR.halt\_req is 1.
- THCSR.halt\_ack is 1 after a halt request indicates that the request is accepted.
- After halted, the counter can be resumed to count normally when clearing the halt request to 0.
- The period of each counting number needs to be same when halts and not halt as described in the below waveform.

### 1.2.3. Timer Interrupt

- Timer interrupt (tim\_int) is asserted (set) when the interrupt is enabled and the counter's value matches (equals) the compare value.
- Once asserted, the timer interrupt (tim\_int) remains unchanged until it is cleared by writing 1 to TISR.int\_st bit or the interrupt is disabled.

### 1.2.4. Counting Mode

- In default mode, counting speed depends on the system clock (description below the waveform)
- The counting speed can be controlled by register settings by setting into TCR.div\_en and TCR.div\_val[3:0] as the register specification
- div\_en and div\_val can not be changed during timer\_en is High.
- Not allowed to change div\_en and div\_val while timer\_en is High by an error response when the user mistakenly accesses.

### 1.2.5. APB slave/ Register

- Read/write to reserved area is RAZ/WI (read as zero, write ignored)
- Support byte access: bus can access individual bytes in the register.
- Support wait state (1cycle) to improve the timing.
- Support error handling for some prohibited access:
  - + write prohibited value to TCR.div\_val
  - + div\_en or div\_val changes during timer operation
- When an error occurs, data is not written into the register bit/fields

## 1.3. Block diagram

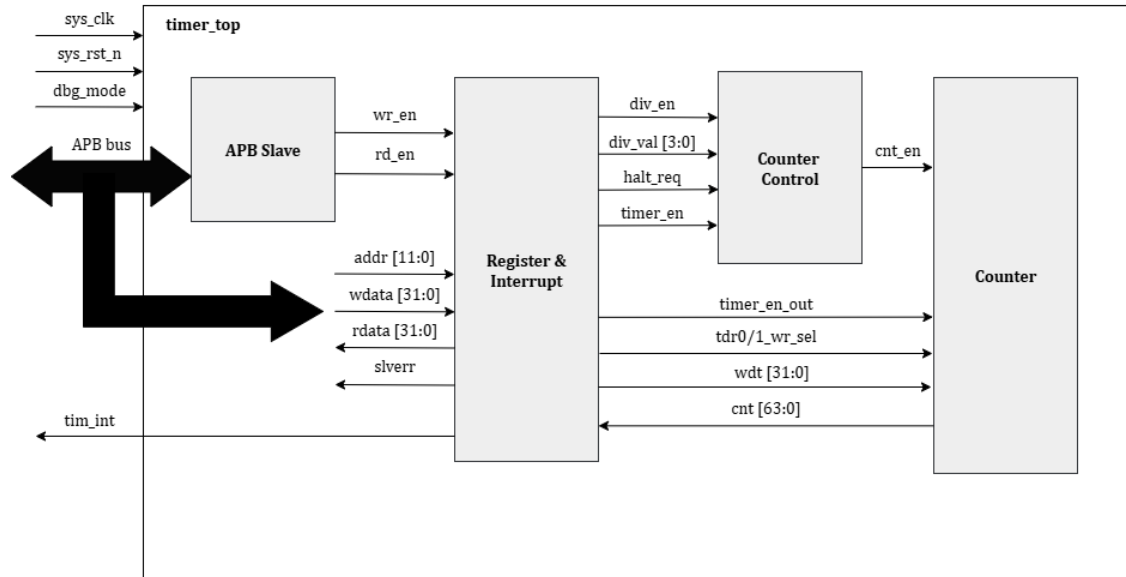


Figure 1: Block diagram of Timer IP

### 1.3.1. APB slave

The APB Slave block provides the interface between the APB bus and the timer module. It decodes read and write transactions from the bus and generates control signals (`wr_en`, `rd_en`) to enable register access. Its purpose is to ensure correct handshaking and communication with the APB bus.

### 1.3.2. Register & Interrupt

This block stores configuration and control registers, such as divider value, halt request, and timer enable. It manages data flow between the APB interface and internal logic and is also responsible for generating timer interrupts (`tim_int`) when programmed conditions are met.

### 1.3.3. Counter Control

The Counter Control block determines when the counter should increment. It generates the `cnt_en` signal based on the selected divider, timer enable, and halt request. Its purpose is to provide flexible timing control and support different counting modes depending on the configuration.



### 1.3.4. Counter

The Counter block is a 64-bit up-counter that increments on every active cnt\_en pulse. It can be accessed and updated through the APB interface, allowing both software read/write operations and automatic counting. Its purpose is to serve as the core timing element of the module.

## 1.4. Interface signals

Table 1: Interface signals of timer IP

Signal name	Width	Direction	Description
sys_clk	1	Input	A clock signal
sys_rst_n	1	Input	The reset signal is active-LOW
tim_psel	1	Input	Indicates that the Slave is selected and a data transfer is required.
tim_pwrite	1	Input	An APB write access when is HIGH and an APB read access when is LOW
tim_penable	1	Input	Indicates the second and subsequent cycles of an APB transfer.
tim_paddr [31:0]	32	Input	Is the APB address bus
tim_pwdata [31:0]	32	Input	Write data bus is driven during write cycles when tim_pwrite is HIGH
tim_prdata [31:0]	32	Output	Read data bus is driven during read cycles when tim_pwrite is LOW
tim_pstrb [3:0]	4	Input	Byte lanes to update during a write transfer, not be active during a read transfer
tim_pready	1	Output	Slave signals ready; can extend the transfer
tim_pslverr	1	Output	An optional signal that can be asserted HIGH by the Master to indicate an error condition
tim_int	1	Output	Is asserted when the interrupt is enabled and counter's value matches the compare value
dbg_mode	1	Input	For simple, dbg_mode does not change after timer_en is High.

## 2. Register Specification

### 2.1. Register Summary

Table 2: Register Summary

Offset	Abbreviation	Register name
0x00	TCR	Timer Control Register
0x04	TDR0	Timer Data Register 0
0x08	TDR1	Timer Data Register 1
0x0C	TCMP0	Timer Compare Register 0
0x10	TCMP1	Timer Compare Register 1
0x14	TIER	Timer Interrupt Enable Register
0x18	TISR	Timer Interrupt Status Register
0x1C	THCSR	Timer Halt Control Status Register
Others	Reserved	

### 2.2. Timer Control Register (TCR)

Offset address: 0x0

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				div_val [3:0]				Reserved				div_en		tim_en	
rw												rw		rw	

Table 3. Timer Control Register (TCR)

Bit	Name	Type	Default value	Description
31:12	Reserved	-	20'h0	Reserved
11:8	div_val[3:0]	RW	4'b0001	Counter control mode setting:

				<ul style="list-style-type: none"> <li>• 4'b0000: Counting speed is not divided</li> <li>• 4'b0001: Counting speed is divided by 2 (default)</li> <li>• 4'b0010: Counting speed is divided by 4</li> <li>• 4'b0011: Counting speed is divided by 8</li> <li>• 4'b0100: Counting speed is divided by 16</li> <li>• 4'b0101: Counting speed is divided by 32</li> <li>• 4'b0110: Counting speed is divided by 64</li> <li>• 4'b0111: Counting speed is divided by 128</li> <li>• 4'b1000: Counting speed is divided by 256</li> <li>• Others: prohibit settings.</li> </ul> <p>When setting the prohibit value, div_val is not changed.</p> <p>Note: user must not change div_val while timer_en is High</p> <p>(*): add hardware logic to ensure div_val is prohibited to change when timer_en is High. Access is error response in this case.</p> <p>(*)access is "error response" when setting prohibit value to div_val</p>
7:2	Reserved	RO	6'b0	Reserved
1	div_en	RW	RW	<p>Counter control mode enable.</p> <ul style="list-style-type: none"> <li>• 0: Disabled. Counter counts with normal speed based on system clock</li> <li>• 1: Enabled. The counting speed of counter is controlled based on div_val</li> </ul> <p>Note: user must not change div_en while timer_en is High</p>

				(*): add hardware logic to ensure div_en is prohibited to change when timer_en is High. Access is error response in this case.
0	timer_en	RW	1'b0	<p>Timer enable</p> <ul style="list-style-type: none"> <li>• 0: Disabled. Counter does not count.</li> <li>• 1: Enabled. Counter starts counting.</li> </ul> <p>(*) timer_en changes from H-&gt;L will initialize the TDR0/1 to their initial value</p>

## 2.3. Timer Data Register 0 (TDR0)

Offset address: 0x04

Reset value: 32'h0000\_0000

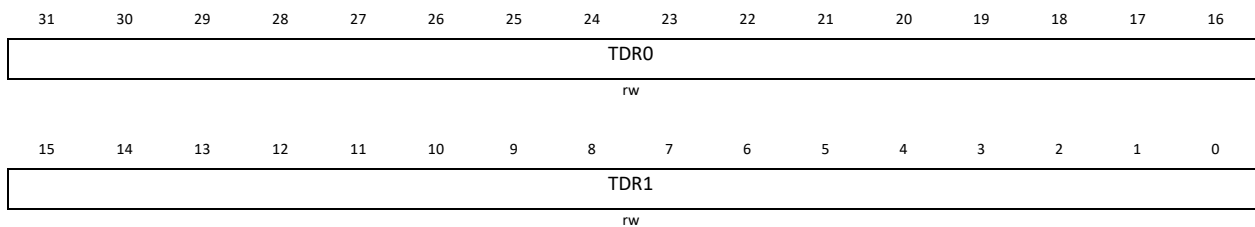


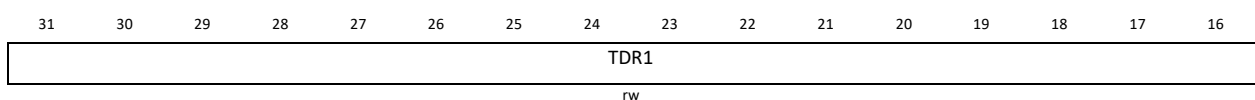
Table 4. Timer Data Register 0 (TDR0)

Bit	Name	Type	Default value	Description
31:0	TDR0	RW	32'h0000_0000	<p>Lower 32-bit of 64-bit counter.</p> <p>value of this register is cleared to initial value when timer_en changes from H-&gt;L.</p>

## 2.4. Timer Data Register 1 (TDR1)

Offset address: 0x08

Reset value: 32'h0000\_0000



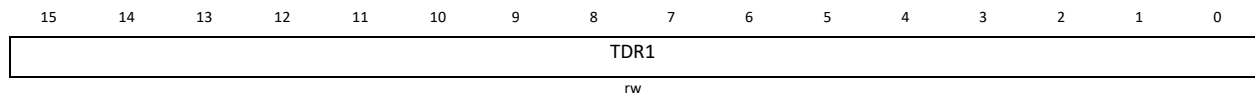


Table 5. Timer Data Register 1 (TDR1)

Bit	Name	Type	Default value	Description
31:0	TDR1	RW	32'h0000_0000	Upper 32-bit of 64-bit counter. value of this register is cleared to initial value when timer_en changes from H->L.

## 2.5. Timer Compare Register 0 (TCMP0)

Offset address: 0x0C

Reset value: 32'hFFFF\_FFFF

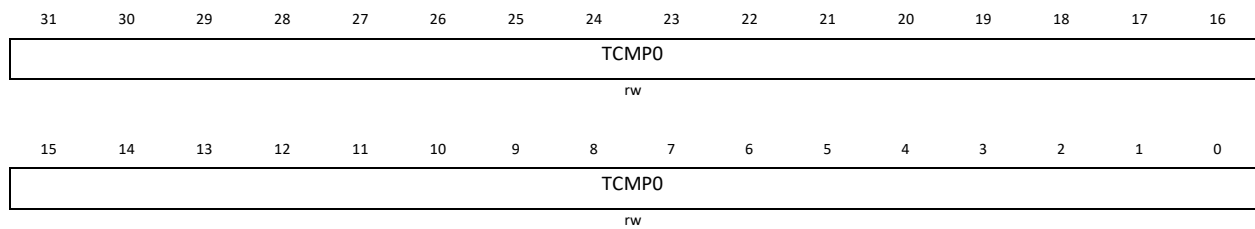


Table 6. Timer Compare Register 0 (TCMP0)

Bit	Name	Type	Default value	Description
31:0	TCMP0	RW	32'hFFFF_FFFF	Lower 32-bit of 64-bit compare value. Interrupt is asserted when counter value is qual to compare value.

## 2.6. Timer Compare Register 1 (TCMP1)

Offset address: 0x10

Reset value: 32'hFFFF\_FFFF

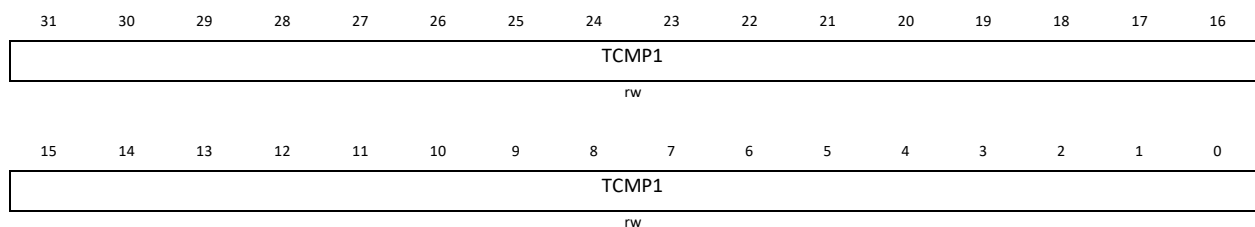


Table 7. Timer Compare Register 1 (TCMP1)

Bit	Name	Type	Default value	Description
31:0	TCMP1	RW	32'hFFFFFF_FFFF	Upper 32-bit of 64-bit compare value. Interrupt is asserted when counter value is equal to compare value.

## 2.7. Timer Interrupt Enable Register (TIER)

Offset address: 0x14

Reset value: 32'h0000\_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

Table 8. Timer Interrupt Enable Register (TIER)

Bit	Name	Type	Default value	Description
31:1	Reserved	RO	31'h0	Reserved
0	int_en	R/W	1'b0	Timer interrupt enable 0: Timer interrupt is disabled. 1: Timer interrupt is enabled. When this bit is 0, no timer interrupt is output. When this bit is 1, timer interrupt can be output when reaching trigger condition. Clearing this bit to 0 while interrupt is asserting will mask the interrupt to 0 but does not affect the interrupt pending bit TISR.int_st bit

## 2.8. Timer Interrupt Status Register (TISR)

Offset address: 0x18

Reset value: 32'h0000\_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_st	
															rw

Table 9. Timer Interrupt Status Register (TISR)

Bit	Name	Type	Default value	Description
31:1	Reserved	RO	31'h0	Reserved
0	int_en	R/W	1'b0	<p>Timer interrupt trigger condition status bit (interrupt pending bit)</p> <p>0: the interrupt trigger condition does not occur.</p> <p>1: the interrupt trigger condition occurred.</p> <p>Write 1 when this bit is 1 to clear it</p> <p>Write 0 when this bit is 1 has no effect</p> <p>Write to this bit when it is 0 has no effect.</p> <p>Note: When interrupt trigger condition occurred (counter reached compare value), counter continues to count normally.</p>

## 2.9. Timer Halt Control Status Register (THCSR)

Offset address: 0x1C

Reset value: 32'h0000\_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														halt_ack	halt_req
														rw	rw

Table 10. Timer Halt Control Status Register (THCSR)

Bit	Name	Type	Default value	Description
31:2	Reserved	RO	30'h0	Reserved
1	halt_ack	RO	1'b0	Timer halt acknowledge 0: timer is NOT halted 1: timer is halted Timer accepts the halt request only in debug mode, indicates by debug_mode input signal
0	halt_req	RW	RW	Timer halt request 0: no halt req. 1: timer is requested to halt.

## 3. Functional Description

### 3.1. APB slave

The APB slave module implements the control logic for an AMBA APB (Advanced Peripheral Bus) slave interface. Its main responsibilities include handling read/write enable signals and generating the pready signal according to the APB protocol timing. This APB slave has below features:

- Supports APB protocol (setup & access phases).
- Generates wr\_en for write operations.
- Generates rd\_en for read operations.
- Provides pready signal to indicate transfer completion.
- Active-low reset and synchronous operation with pclk.

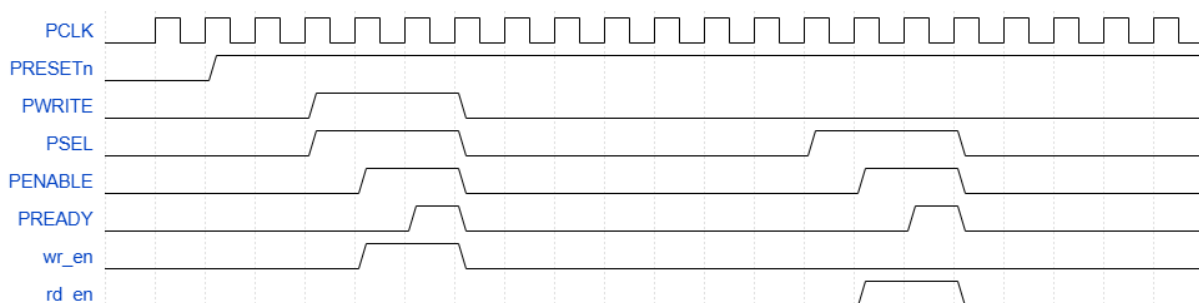


Figure 2: Waveform of APB Slave



The `wr_en` signal is a pulse and is asserted when `psel`, `penable` and `pwrite` is High. It indicates that there's a write command. This signal is sent to register module to store data.

The `rd_en` signal is a pulse and is asserted when `psel`, `penable` is High. It indicates that there's a read command. This signal is sent to the register module to load data into the registers.

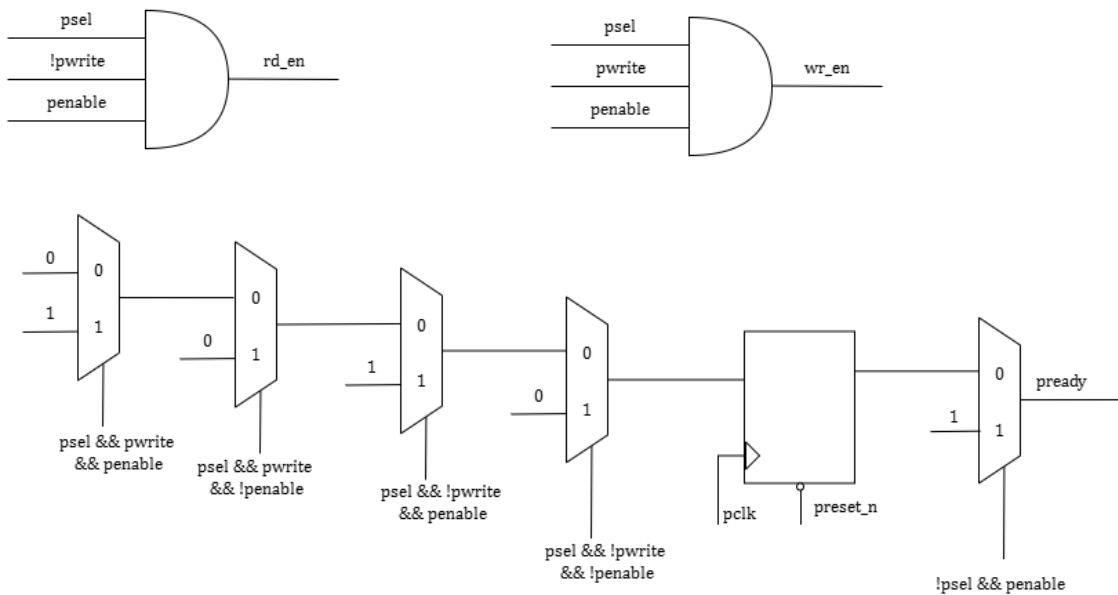


Figure 3: Logic diagram of APB Slave

According to the APB protocol, the read process occurs when both `psel` and `penable` are active, while the write process occurs when `psel`, `penable`, and `pwrite` are active. Therefore, AND gates are used to generate two signals that satisfy these conditions.

For the `pready` signal, multiple multiplexers are used to evaluate different cases of `psel`, `pwrite`, and `penable`, corresponding to the read and write operations of the protocol. The result is then stored in a D flip-flop and finally selected through a multiplexer to generate the `pready` signal.

## 3.2. Register & Interrupt

The register module implements the control and status registers of a 64-bit timer/counter. It interfaces with the APB bus through read/write signals and a 12-bit address decoder. The registers allow software to configure timer operation, enable interrupts, read the current counter value, and control debug-related functions. This register and interrupt has below features:

- Supports a 64-bit counter (accessible through TDR0 and TDR1).

- Includes 64-bit compare registers (TCMP0/TCMP1) for match detection.
- Generates an interrupt on counter/compare match when enabled.
- Provides debug halt control for stopping the timer in debug mode.
- Implements error checking to prevent invalid register updates.

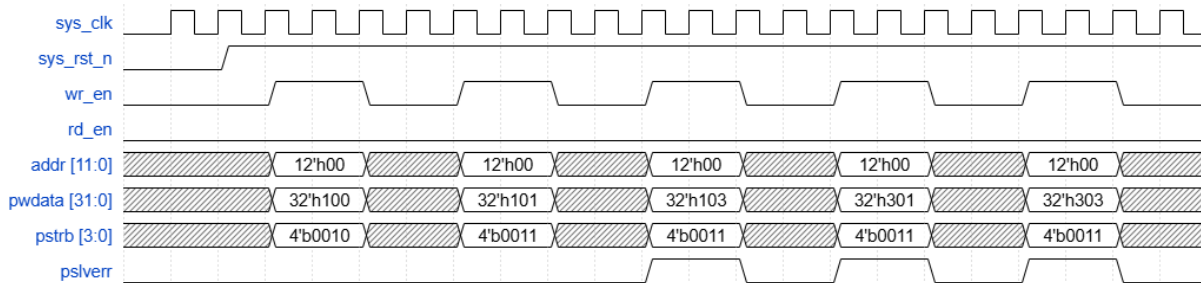


Figure 4: Waveform of TCR register

The waveform illustrates how the pslverr signal behaves when changing the values of div\_val and div\_en while timer\_en = 1.

After the sys\_rst\_n signal is asserted, a write operation takes place. The wr\_en signal is asserted for two cycles to write div\_val = 4'b0001 into the TCR register. Next, after wr\_en is asserted again, we write timer\_en = 1 into the TCR register. On the following assertion of wr\_en, we write div\_en = 1 into the register. At this moment, the pslverr signal is asserted to indicate an error has occurred.

On the next wr\_en assertion, we write div\_val = 4'b0011 into the register, and the pslverr signal remains asserted. Finally, on the last wr\_en assertion, we write both div\_val = 4'b0011 and div\_en = 1, and the pslverr signal is still asserted.

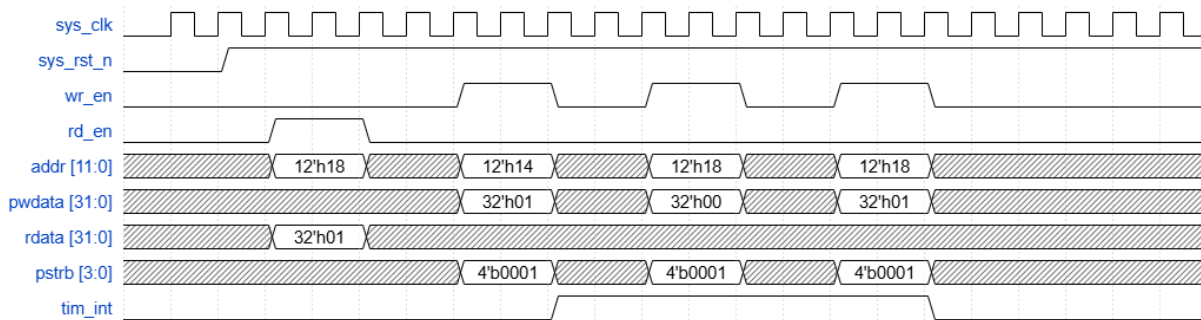


Figure 5: Waveform of TIER register and TISR register

The waveform illustrates the interrupt behavior involving the TIER and TISR registers. Suppose an interrupt occurs, and the corresponding value is stored in the TISR register. When reading the TISR, we

obtain the value 32'h01. Next, we write `int_en = 1` into the TIER register to enable the interrupt output, and after one cycle the `tim_int` signal becomes active.

Then, we write the value 32'h00 into the TIER register to test the RW1C (Read-Write 1 to Clear) mechanism. Since this does not clear the interrupt, `tim_int` remains asserted. Finally, by writing 32'h01 into the TIER register, the interrupt is cleared correctly according to the RW1C mechanism.

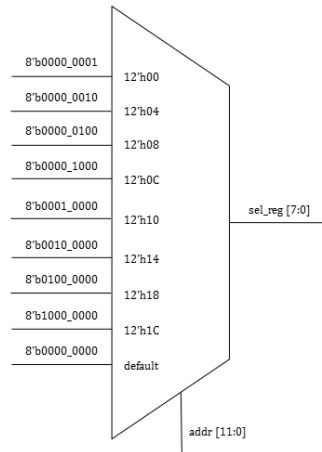


Figure 6: Logic diagram of the address decoder

The circuit is a multiplexer designed for register address selection, where the stored value is controlled using a one-hot encoding mechanism.

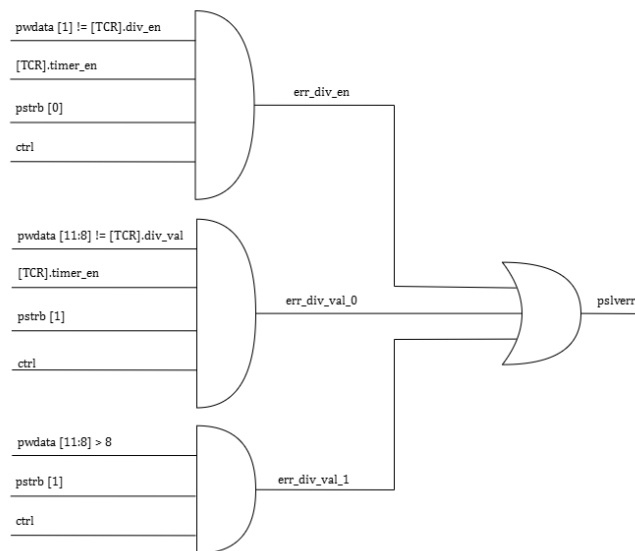


Figure 7: Logic diagram of the pslverr signal

This is the logic diagram for generating the pslverr signal. First, for the AND gate with output err\_div\_en, the condition is that when wr\_en is asserted, the TCR register is selected, timer\_en = 1, the first 8 bits are enabled for data modification, and the written div\_en value differs from the initial value when timer\_en = 0, an error will occur.

For the AND gate with output err\_div\_val\_0, the condition is similar to that of div\_en, except that the checked data is div\_val, and the second 8 bits are enabled for writing. Finally, an error is also generated if the written div\_val exceeds 8. All these outputs are combined through an OR gate to produce the pslverr signal.

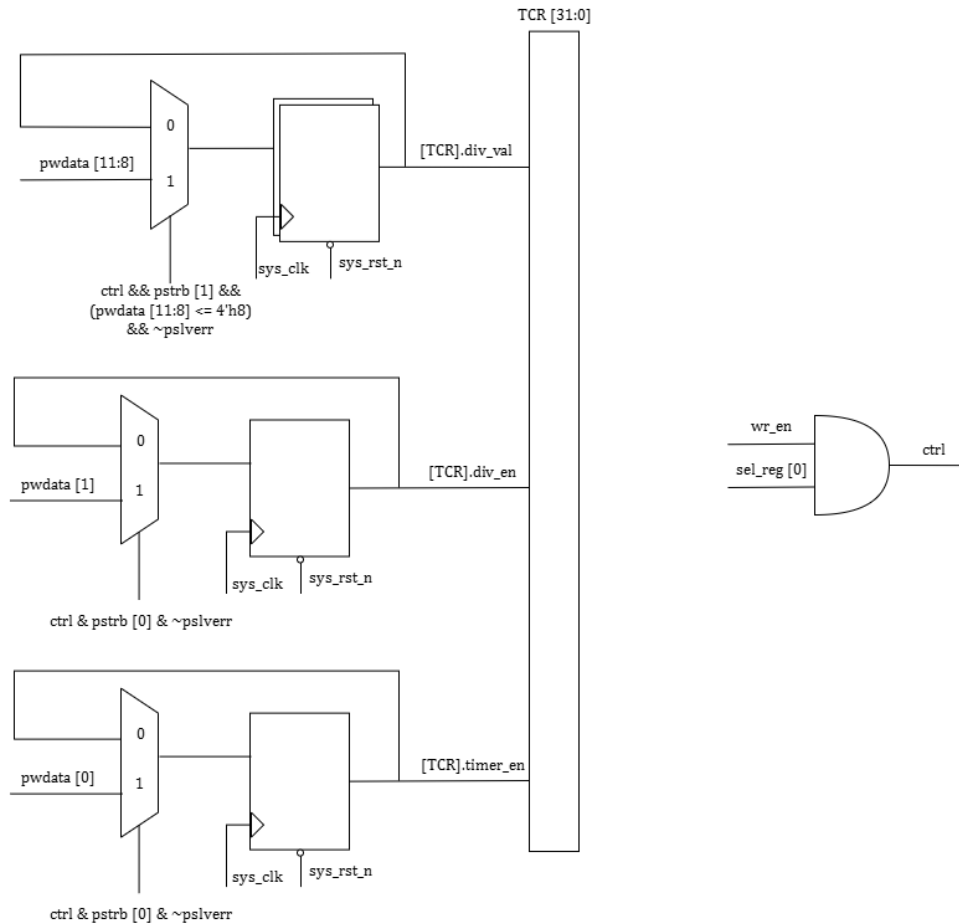


Figure 8: Logic diagram of TCR Register

This logic diagram employs D flip-flops with conditional feedback to store the bit values of the TCR register. When wr\_en is asserted, the lower 8 bits are enabled for modification, and pslverr is low, the

values of `div_en` and `timer_en` are updated from the corresponding bit positions of `pwdata`. If the condition is not satisfied, the feedback path preserves the previous values.

For `div_val`, when `wr_en` is asserted, the next 8 bits are enabled for writing, the new value is less than 9, and `pslverr` is low, the `div_val` field is updated from the corresponding bit positions of `pwdata`. Otherwise, the feedback path retains the existing value.

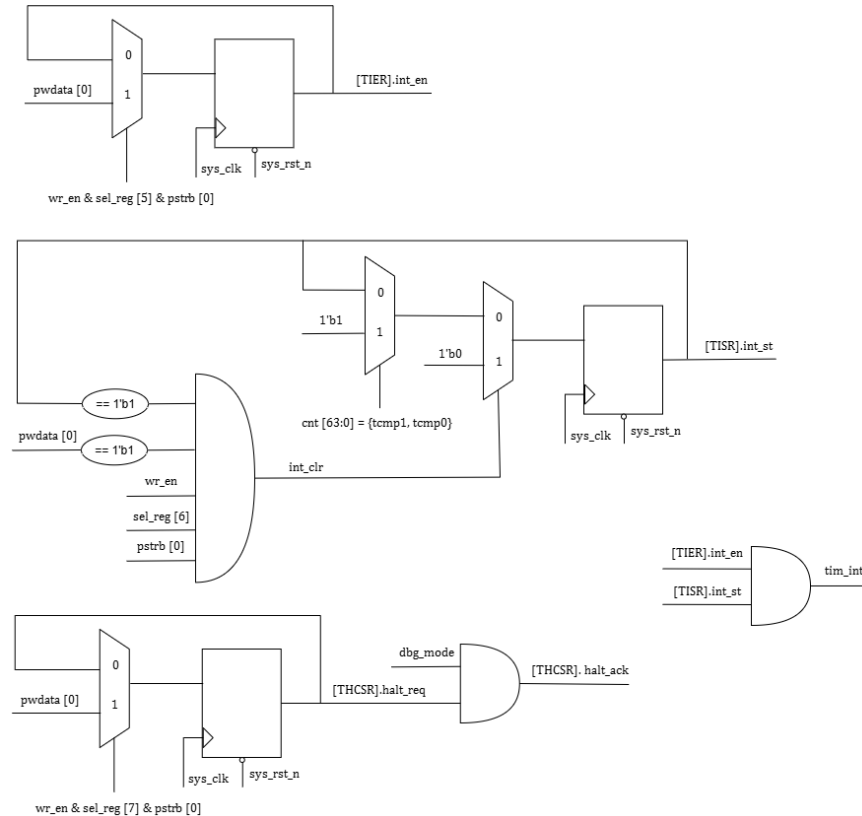


Figure 9: Logic diagram of TIER Register, TISR Register, `tim_int`

This logic diagram implements the interrupt functionality of the timer. For the `halt_req` bit in the THCSR register, a D flip-flop is used to store its value. A multiplexer selects either the feedback path (to retain the current value) or the new write data, with the select condition defined by `wr_en` being asserted, the THCSR register being selected, and the lower 8 bits enabled for modification. The output of this flip-flop is then ANDed with `dbg_mode` to generate the `halt_ack` bit.

For the `int_st` bit in the TISR register, a D flip-flop is also used for storage. The bit is set to 1 when the counter value matches the concatenated values of `TCMP1` and `TCMP0`. It is cleared to 0 when the following conditions are met: `int_st = 1`, the write data = 1, the TISR register is selected, the lower 8 bits

are enabled, and `wr_en` is asserted. If neither condition is satisfied, the feedback path retains the current value of `int_st`. Finally, the output `tim_int` is asserted only when both the interrupt is enabled and an interrupt event occurs, which is achieved through an AND gate.

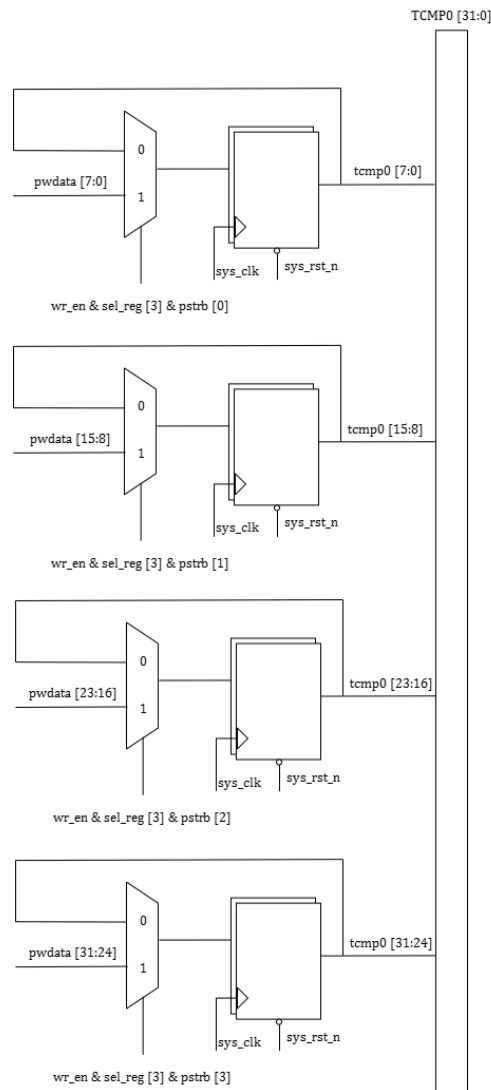


Figure 10: Logic diagram of TCMP0 Register

Multiple D flip-flops are used to store each 8-bit segment of the register, with their inputs connected to a multiplexer. New data from `pwdata` is transferred when the following conditions are met: `wr_en` is asserted, the register is selected, and the corresponding `pstrb` bit enables the update of the associated 8-bit field in the TCMP0 register.

### 3.3. Counter control

The counter control module is responsible for generating the enable signal (cnt\_en) that drives the operation of a 64-bit timer/counter. It defines the timing at which the counter is incremented and provides flexible control features to support both normal and debug operation. The design is based on an internal counter with programmable division logic, combined with control signals such as div\_en, div\_val, timer\_en, and halt\_req. This allows the timer to operate either in a straightforward free-running mode or in a more fine-grained programmable division mode, depending on system requirements. This counter control has below features:

- Supports both direct clock mode (no division) and programmable clock division mode.
- Flexible division ratios up to divide-by-256 (with div\_val=1000).
- Debug halt function to stop counter progression during debug.
- Clean integration with the timer register block to control counter stepping.

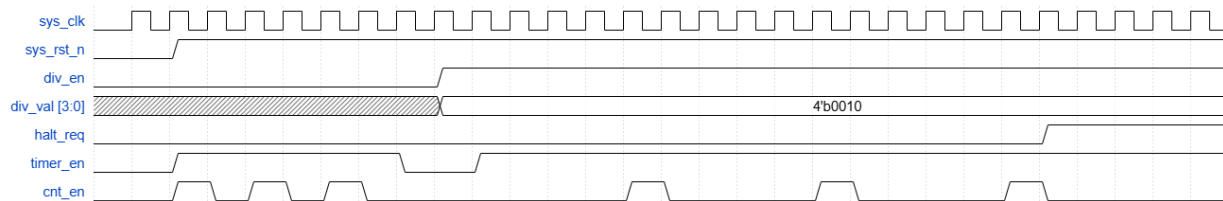


Figure 11: Waveform of Counter control

The waveform illustrates the block operation in default mode. When div\_en and halt\_req are inactive while timer\_en is active, the block operates in default mode, and cnt\_en is asserted every clock cycle. When div\_en is asserted with div\_val = 4'b0010, the block switches to control mode. In this mode, after four cycles, cnt\_en is asserted and then de-asserted immediately on the next rising edge of sys\_clk. Finally, when halt\_req is asserted, cnt\_en remains inactive regardless of the operating mode.

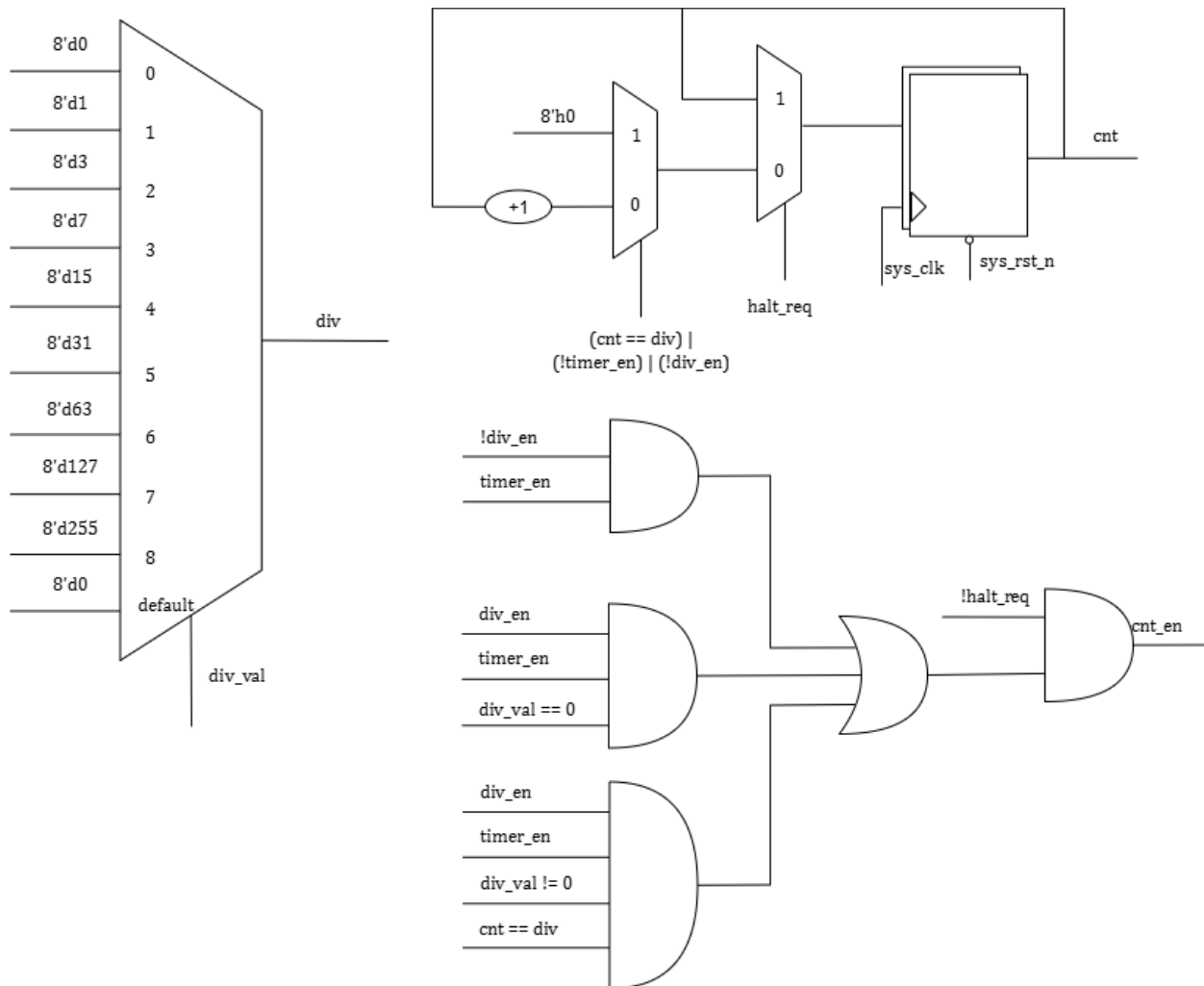


Figure 12: Logic diagram of Counter control

A multiplexer is used to select the division value for the timer based on the `div_val` signal. Then, a set of D flip-flops combined with a multiplexer forms a conditional up-counter circuit. The purpose of this circuit is to track the number of rising edges of `sys_clk`. When the number of rising edges equals the selected division value, and both `timer_en = 1` and `div_en = 1`, the counter increments by one. Additional logic gates are employed to generate the counter enable signal (`cnt_en`) under different conditions, depending on whether the block is operating in default mode or control mode, and also taking into account the `halt_req` signal.



### 3.4. Counter

The counter module implements a 64-bit up-counter with support for software writes, programmable byte strobes, and reset control. It provides the timer's core counting functionality, controlled by enable signals from the counter\_control and register modules. This counter has below features:

- Implements a 64-bit free-running counter with write access for software control.
- Supports byte-level write masking via pstrb.
- Counter value can be reset via timer\_en or asynchronous reset.
- Provides seamless incrementing while allowing software overwrite of specific bytes.

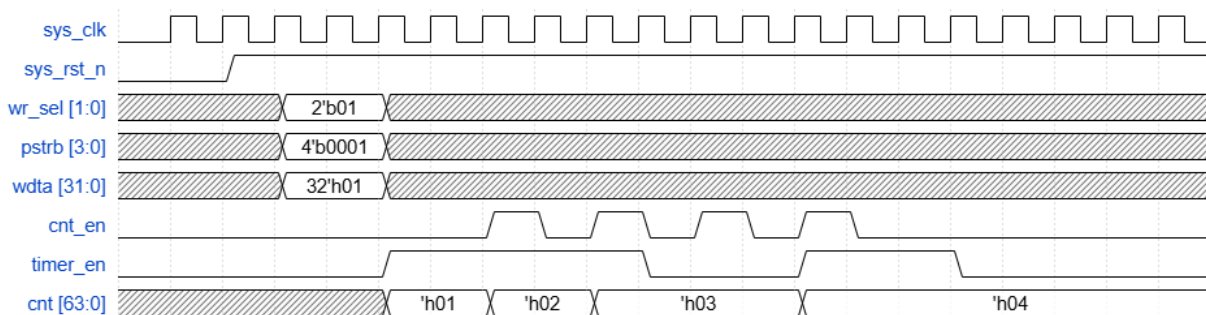


Figure 13: Waveform of Counter

When `wr_sel = 2'b01`, the TDR0 register is selected, and the value `32'h01` is written into the register. Once `timer_en` is asserted, the counter is enabled to start counting. For every rising edge of `cnt_en`, the counter increments by one. When `timer_en` is de-asserted, the counter remains unchanged even if rising edges of `cnt_en` occur.

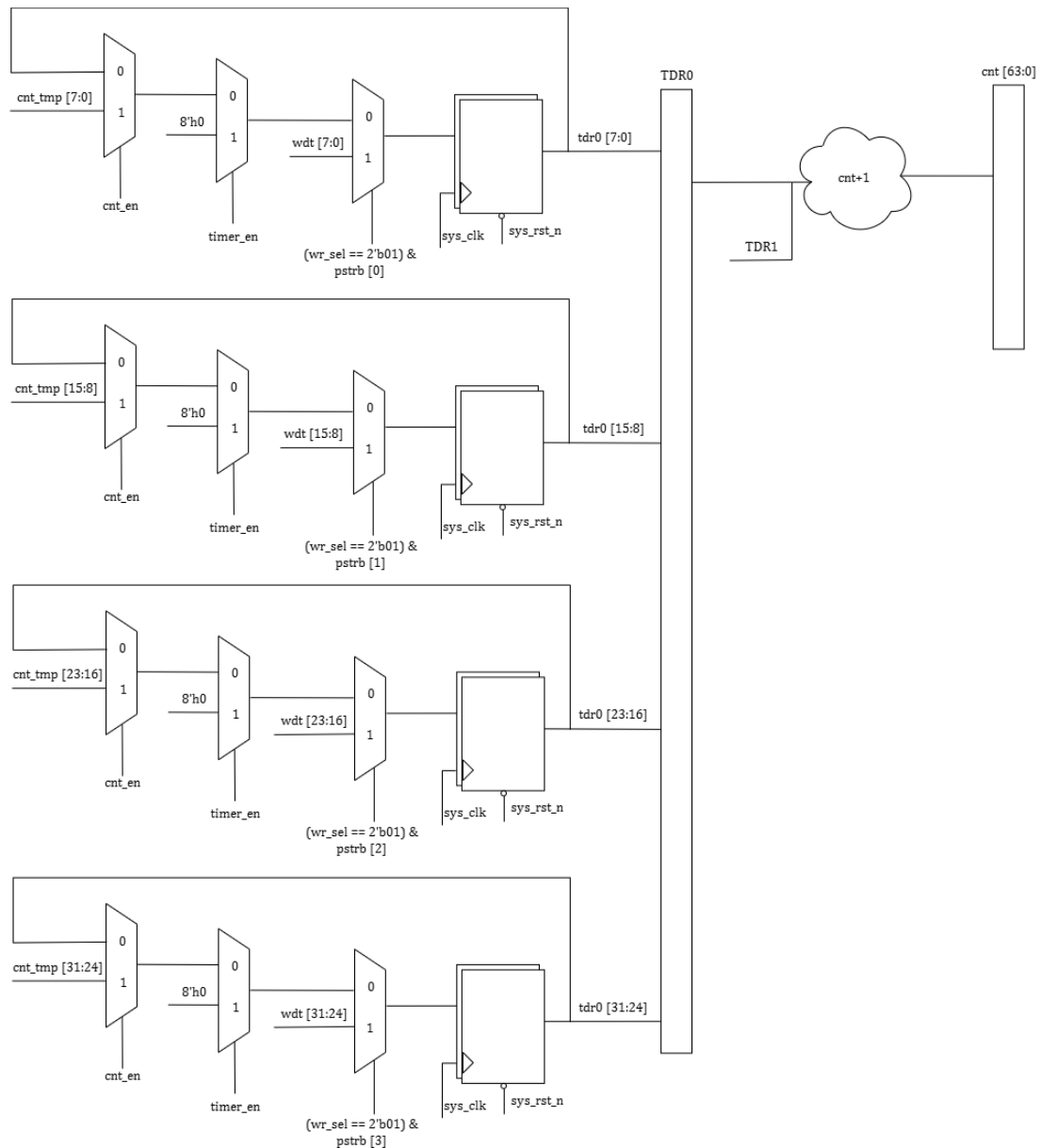


Figure 14: Logic diagram of Counter

This is the logic diagram of the TDR0 register (with TDR1 implemented in a similar way) together with the circuit that generates the cnt value. Multiple D flip-flops are used to store each 8-bit segment of the register, while multiplexers are employed to select the inputs of the flip-flops based on the control signals timer\_en, cnt\_en, wr\_sel, and pstrb.

## 4. Verify

### 4.1. Vplan

Link vplan for Timer IP verification: [Vplan](#)

### 4.2. Coverage

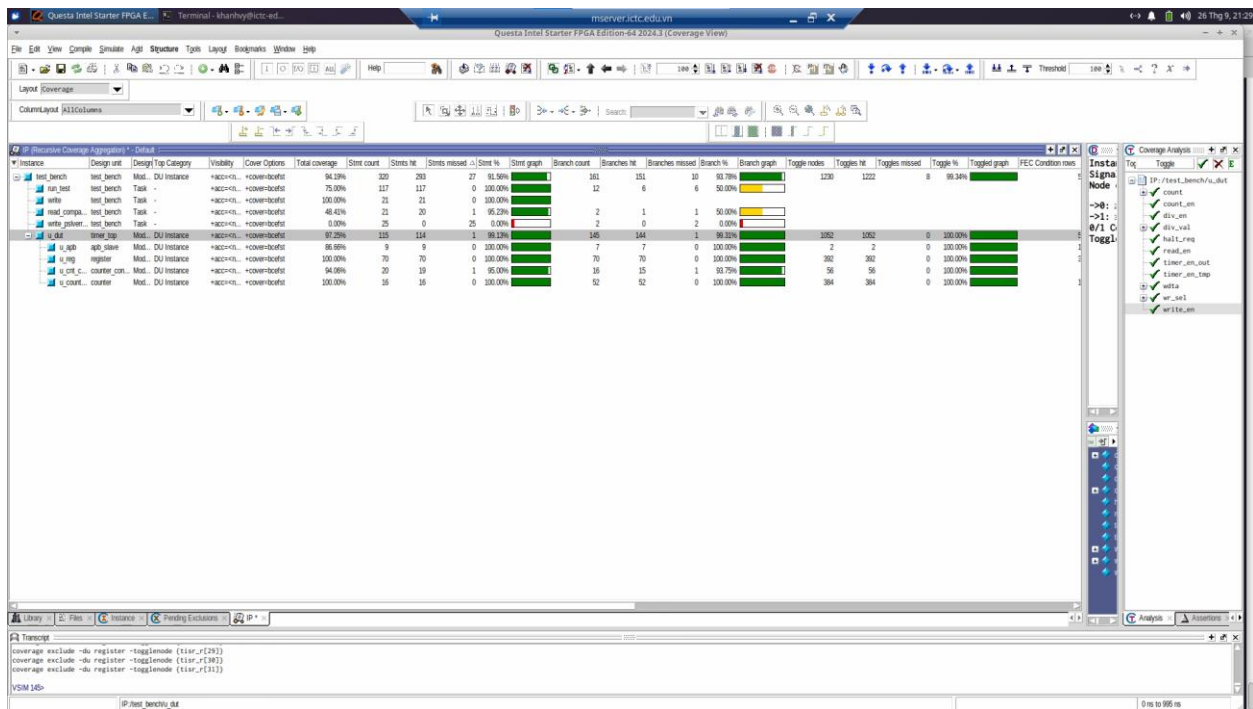


Figure 15: The coverage achieved is 100%

## 5. History

Date	Version	Description	Author
9/26/2025	1.0	Newly created	Nguyễn Đình Khánh Vy