IC TRAINING CENTER VIET NAM
FUNDAMENTAL IC DESIGN AND VERIFICATION COURSE

FINAL PROJECT
TIMER IP DESIGN SPECIFICATION

Student: Nguyễn Tân Thành
Class: IC30
Instructor: Nguyễn Thanh Trí

Contents

LIST OF TABLES and FIGURES ... **3**
LIST OF FIGURES ... **3**
1.Overview ... **5**
1.1. Introduction ... 5
1.2. Main features ... 5
1.3. Block diagram ... 6
1.4. Interface signals ... 7
2.Register Specification ... **7**
2.1. Register Summary ... 7
2.2. Timer Control Register (TCR) ... 7
2.3. Timer Data Register 0 (TDR0) ... 9
2.4. Timer Data Register 1 (TDR1) ... 9
2.5. Timer Compare Register 0 (TCMP0) ... 10
2.6. Timer Compare Register 1 (TCMP1) ... 10
2.7. Timer Interrupt Enable Register (TIER) ... 11
2.8. Timer Interrupt Status Register (TISR) ... 11
2.9. Timer Halt Control Status Register (THCSR) ... 12
3. Functional Description ... **13**
3.1. APB slave Block ... 13
3.2. Register Block ... 14
3.2.1. TCR register ... 14
3.2.2. Function strobe_merge ... 15
3.2.3. TDR register ... 16
3.2.3. TCMP register ... 17
3.2.4. TIER and TISR register ... 18
3.2.4. THCSR register ... 19
3.3. Counter block ... 20
3.3.1. Sub counter ... 20
3.3.2. Main counter ... 21
4. History ... **24**

LIST OF TABLES and FIGURES

Table 1: Interface signals of timer IP....7
Table 2: Register Summary.... 7
Table 3. Timer Control Register (TCR).... 8
Table 4: Timer Data Register 0 (TDR0).... 9
Table 5: Timer Data Register 1 (TDR1).... 9
Table 6: Timer Compare Register 0 (TCMP0).... 10
Table 7: Timer Compare Register 1 (TCMP1).... 10
Table 8: Timer Interrupt Enable Register (TIER)....11
Table 9: Timer Interrupt Status Register (TISR)....11
Table 10: Timer Halt Control Status Register (THCSR)....12

LIST OF FIGURES

Figure 1: Block diagram of Timer IP.... 6
Figure 2: Waveform of apb slave....13
Figure 3: block diagram of apb slave....14
Figure 4: Block diagram of TCR register.... 14
Figure 5: Block diagram of TDR register....16
Figure 6: Block diagram of TCMP register.... 17
Figure 7: Block diagrams of TIER and TISR register.... 18
Figure 8:Block diagram of THCSR register....19
Figure 9: wave form of sub counter.... 20
Figure 10: Block diagram of sub counter.... 21
Figure 11: wave form default mode and control mode with div_val=0.... 22
Figure 12: Control mode with div_val=1.... 22
Figure 10: Block diagram of main counter.... 23

1.Overview

1.1. Introduction

The timer is a fundamental and indispensable peripheral module integrated into modern microcontroller units (MCUs) and system-on-chips (SoCs). Its primary function is to provide a reliable and precise mechanism for generating specific timing intervals, thereby orchestrating the timing of various internal operations and external events. By operating from the system clock or an independent oscillator, it delivers a critical timing reference that allows the central processing unit (CPU) to execute tasks asynchronously or at predetermined rates, significantly enhancing system efficiency and control.
The utility of timer modules extends across a vast array of critical applications within embedded systems and digital circuits. They are primarily employed for the generation of precise digital waveforms, including pulses and Pulse Width Modulation (PWM) signals, which are essential for controlling motors, LEDs, and power converters. Furthermore, timers are crucial for creating programmed delays, scheduling periodic events, and generating interrupt requests to signal the CPU that a specific time-based condition has been met. This versatility makes them a cornerstone for implementing real-time operating system (RTOS) schedulers, communication protocol timeouts, and sensor data sampling routines.

1.2. Main features

This timer module is architected around a 64-bit count-up counter and features a 12-bit address space for register access. It is configured as an APB slave, with its register set programmable via the APB bus. The IP supports a standard compliance level, which includes 32-bit APB transfers with no wait states and no error handling, and an advanced level that adds support for single-cycle wait states, error handling, byte access, and debug-mode halt functionality. The module operates from a 200 MHz system clock and utilizes an active-low asynchronous reset. The counter can be clocked directly from the system clock or from a divided version, with a division factor of up to 256. Additionally, the timer provides a configurable interrupt output that can be enabled or disabled as required.

1.3. Block diagram

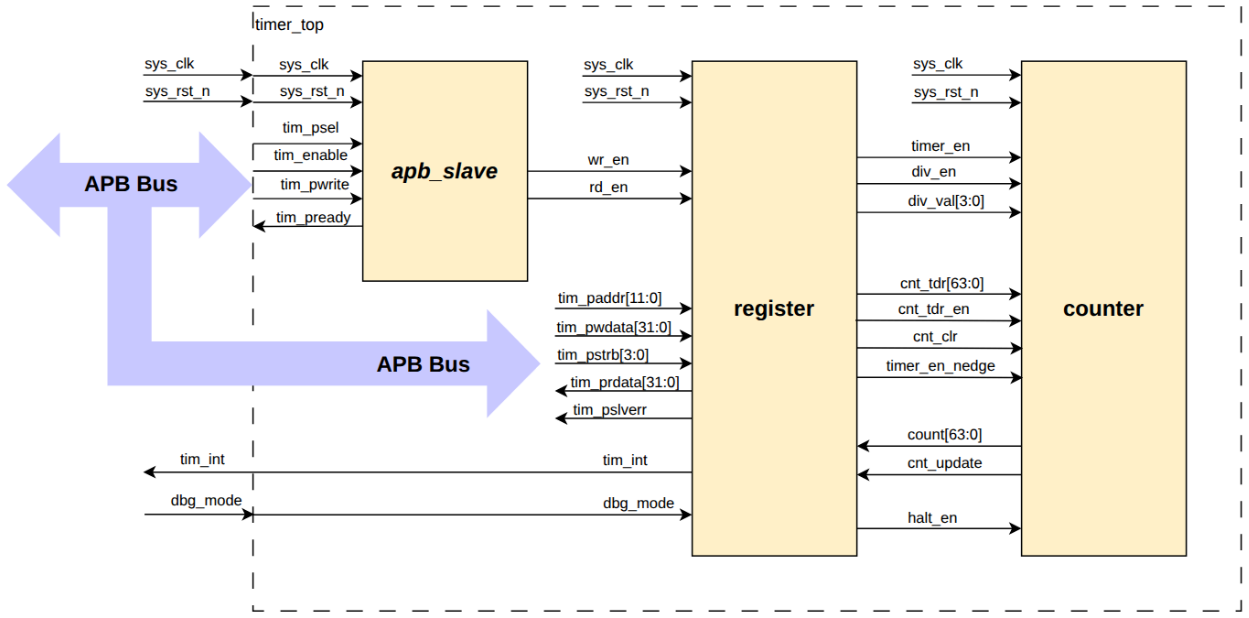


Figure 1: Block diagram of Timer IP

This Timer IP consists of an APB_slave block, a Register block, and a counter block, operating with an active-low asynchronous reset.

- **APB Slave**: Handles protocol handshake, including tim_pready and support wait states 1 cycle.
- **Register**: Stores control and status data to facilitate timer functionality.
- **Counter**: A 64-bit counter module that handles count-up logic and supports clock division; it consists of two sub-blocks: **main_counter** and **sub_counter**.

1.4. Interface signals

Table 1: Interface signals of timer IP

Signal name	Width	Direction	Description
sys_clk	1	input	System's clock, sampling at the edge of clk
sys_rst_n	1	input	Low active reset, release for normal operation
tim_psel	1	input	Select which slaves need to be in charge during the transaction
tim_pwrite	1	input	Signal indicating whether write or not
tim_penable	1	input	Signal indicating the start of Access phase
tim_paddr	12	input	Select which address in slave required for execution
tim_pwdata	32	input	Data that will be written to slave from master
tim_prdata	32	output	Data that master received from slave
tim_pstrb	4	input	Signal enables sparse data transfer
tim_pready	1	output	Signal indicating that slave is ready for the transfer
tim_pslverr	1	output	Signal indicating an error condition during APB transfer
tim_int	1	output	Update the status of interrupt reg, check the trigger condition
dbg_mode	1	input	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_0100

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'b0	Reserved
11:8	DIV_VAL[3:0]	RW	4'b0001	Counter control mode setting: • 4'b0000: Counting speed is not divided • 4'b0001: Counting speed is divided by 2 (default) • 4'b0010: Counting speed is divided by 4 • 4'b0011: Counting speed is divided by 8 • 4'b0100: Counting speed is divided by 16 • 4'b0101: Counting speed is divided by 32 • 4'b0110: Counting speed is divided by 64 • 4'b0111: Counting speed is divided by 128 • 4'b1000: Counting speed is divided by 256 • Others: reserved **- When setting the prohibited value, div_val is not changed and access is "error response".** **- div_val is prohibited to change when timer_en is High. Access is an error response in this case.**
7:2	Reserved	RO	6'b0	Reserved
1	DIV_EN	RW	1'b0	Counter control mode enabled. • 0: Disabled. Counter counts with normal speed based on the system clock. • 1: Enabled. The counting speed of the counter is controlled based on div_val. **div_en is prohibited to change when timer_en is High. Access is an error response in this case.**
0	TIM_EN	RW	1'b0	Timer enable • 0: Disabled. The counter does not count. • 1: Enabled. The counter starts counting **timer_en changes from H->L will initialize the TDR0/1 to their initial value**

2.3. Timer Data Register 0 (TDR0)

Offset address: 0x4

Reset value: 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								TDR0							
								rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								TDR0							
								rw							

Table 4: Timer Data Register 0 (TDR0)

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

2.4. Timer Data Register 1 (TDR1)

Offset address: 0x8

Reset value: 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								TDR1							
								rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								TDR1							
								rw							

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: 0xC
Reset value: 0xFFFF_FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCMP0															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDR0															
rw															

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

2.6. Timer Compare Register 1 (TCMP1)

Offset address: 0x10
Reset value: 0xFFFF_FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCMP1															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDR1															
rw															

Table 7: Timer Compare Register 1 (TCMP1)

Bit	Name	Type	Default value	Description
31:0	TCMP1	RW	32’hffff_ffff	Upper 32-bit of 64-bit compare value

2.7. Timer Interrupt Enable Register (TIER)

Offset address: 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

Table 8: Timer Interrupt Enable Register (TIER)

Bit	Name	Type	Default value	Description
31:1	Reserved	RO	31’h0	Reserved
0	int_en	RW	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 asserted 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: 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_st
-															rw1c

Table 9: Timer Interrupt Status Register (TISR)

Bit	Name	Type	Default value	Description

31:1	Reserved	RO	31’h0	Reserved
0	int_st	RW1C	1’b0	Timer interrupt trigger condition status bit (interrupt pending bit) 0: the interrupt trigger condition does not occur. 1: the interrupt trigger condition occurred. Write 1 when this bit is 1 to clear it Write 0 when this bit is 1 has no effect Writing this bit when it is 0 has no effect. Note: When the interrupt trigger condition occurs (counter reached compare value), the counter continues to count normally.

2.9. Timer Halt Control Status Register (THCSR)

Offset address: 0x18

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														halt_ack	halt_req
-														ro	rw

Table 10: Timer Halt Control Status Register (THCSR)

Bit	Name	Type	Default value	Description
31:1	Reserved	RO	31’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	1’b0	Timer halt request 0: no halt req. 1: timer is requested to halt.

3. Functional Description

3.1. APB slave block

The timer IP has an APB slave to receive settings from the user. This APB slave has below features:

- 12-bit width address.
- 32-bit width read/write data.
- Support byte access: bus can access individual bytes in the register.
- Support wait state (1cycle) to improve the timing.

a. Wave form

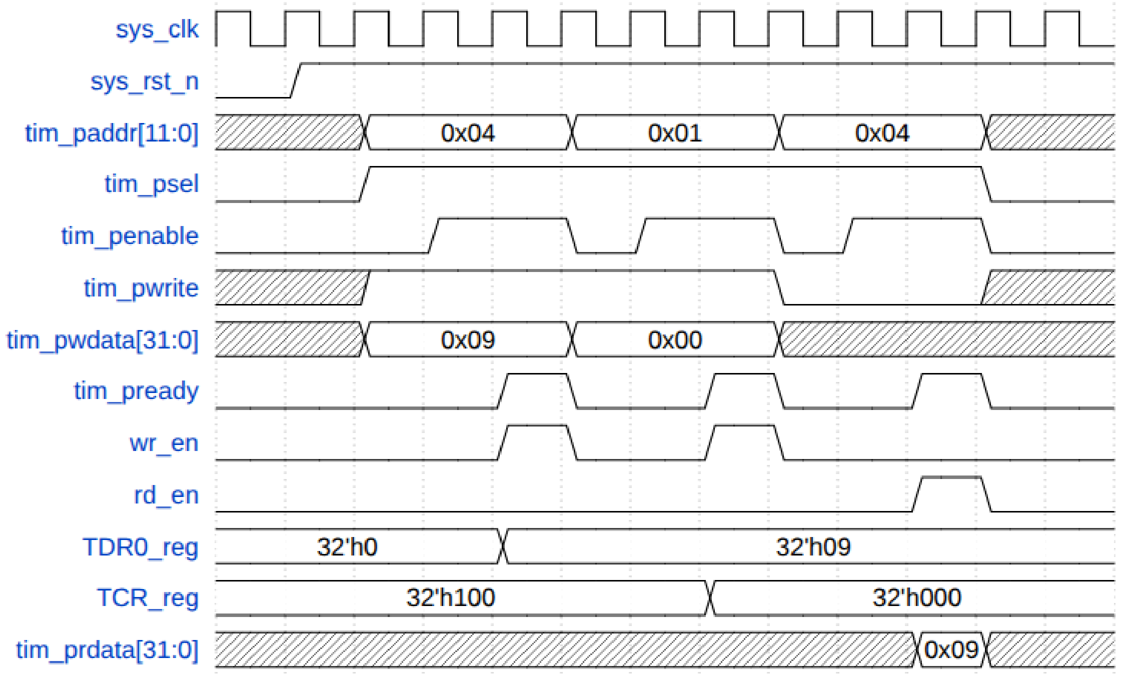


Figure 2: Waveform of apb slave

- At the 4th clock, 32’h09 is written into the TDR0 register at address 0x04.
- At the 7th clock, 32’h00 is written into the TCR register at address 0x00.
- At the 10th clock, read data from the TDR0 register.

b. Block diagram

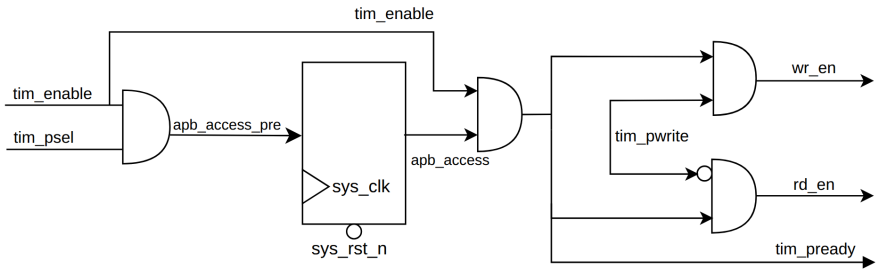


Figure 3: block diagram of apb slave

The APB Slave block decodes bus protocols by synchronizing tim_psel and tim_enable through a Flip-Flop to generate a stable access signal. This signal is gated with tim_pwrite and tim_pready to produce precise wr_en and rd_en control pulses for register operations. This design effectively manages wait-states and ensures data integrity by qualifying transfers only when the system is ready.

3.2. Register block

3.2.1. TCR register

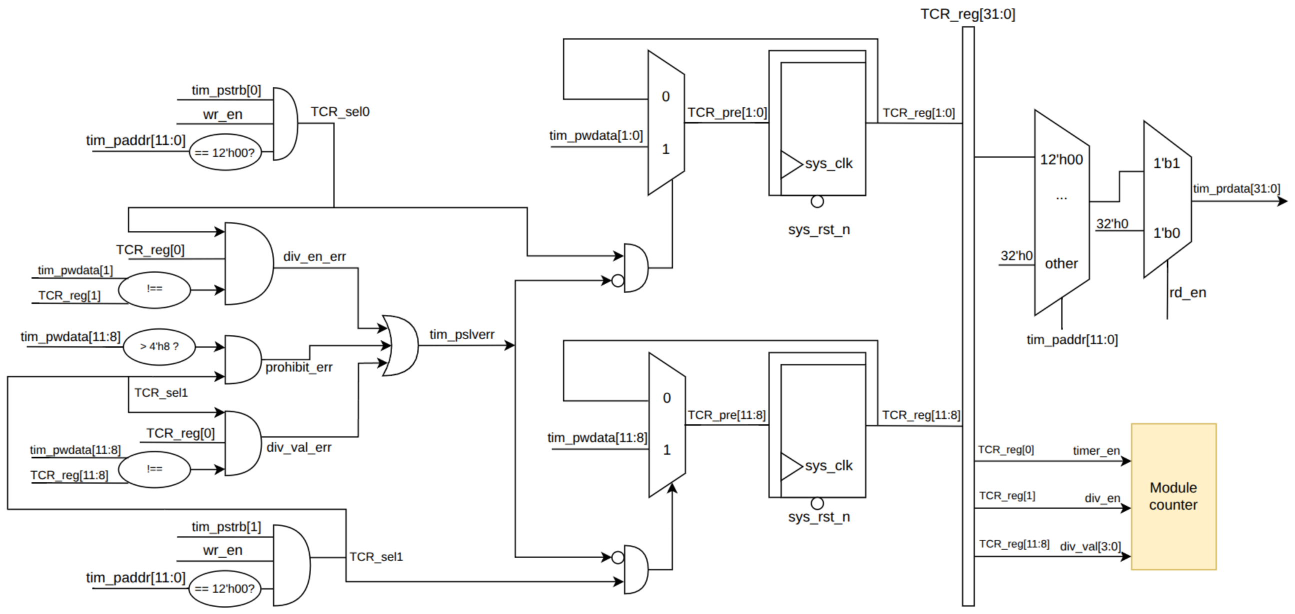


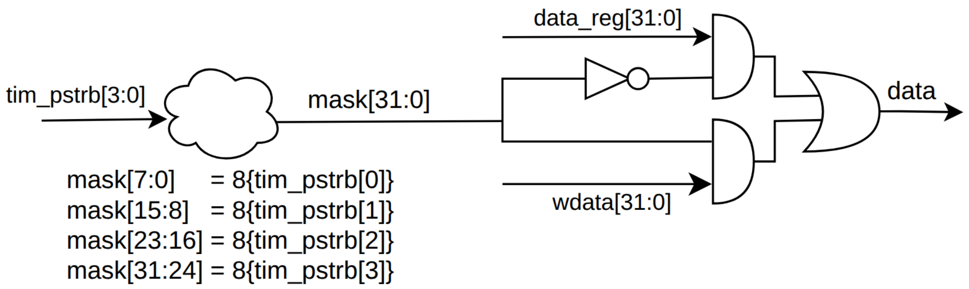
Figure 4: Block diagram of TCR register

While timer_en=1, it is not permitted to change div_en and div_val, and the div_val value must be valid. When these three conditions are met, the data will be written; otherwise, the data will not be written and an error response will be sent to the master.

The three conditions above correspond to the three signals: div_en_err, div_val_err, and prohibit_err:

- div_en_err is asserted when TCR.timer_en = 1 and TCR.div_en != tim_pwdata[1] (change value of TCR.div_en).
- div_val_err is asserted when TCR.timer_en = 1 and TCR.div_val != tim_pwdata[11:8] (change value of TCR.div_val).
- prohibit_err is asserted when the div_val value > 8.
- tim_pslverr includes all three above errors; if any error is asserted, tim_pslverr will be asserted. In this case, data will not be written into the TCR register.

3.2.2. Function strobe_merge


data_reg[31:0]
tim_pstrb[3:0]
mask[31:0]
data
wdata[31:0]
mask[7:0]   = 8{tim_pstrb[0]}
mask[15:8]   = 8{tim_pstrb[1]}
mask[23:16] = 8{tim_pstrb[2]}
mask[31:24] = 8{tim_pstrb[3]}

Purpose: merge write dât with data in register when write strobe.

- Data_reg & (reverse mask): to clear the area want to write
- w_data & mask: to clear unwanted data and retain only the data to be written.

Example:

w_data = 32'h 4444_4444.

TDR0   = 32'h 5555_5555.

Pstrb   = 4'b1010.

- mask   = 32'h FF00_FF00.
- ~mask = 32'h 00FF_00FF.
- TDR0 & ~mask = 32'h 0055_0055.
- W_data & mask = 32'h 4400_4400.
- Data = 32'h 4455_4455.

3.2.3. TDR register

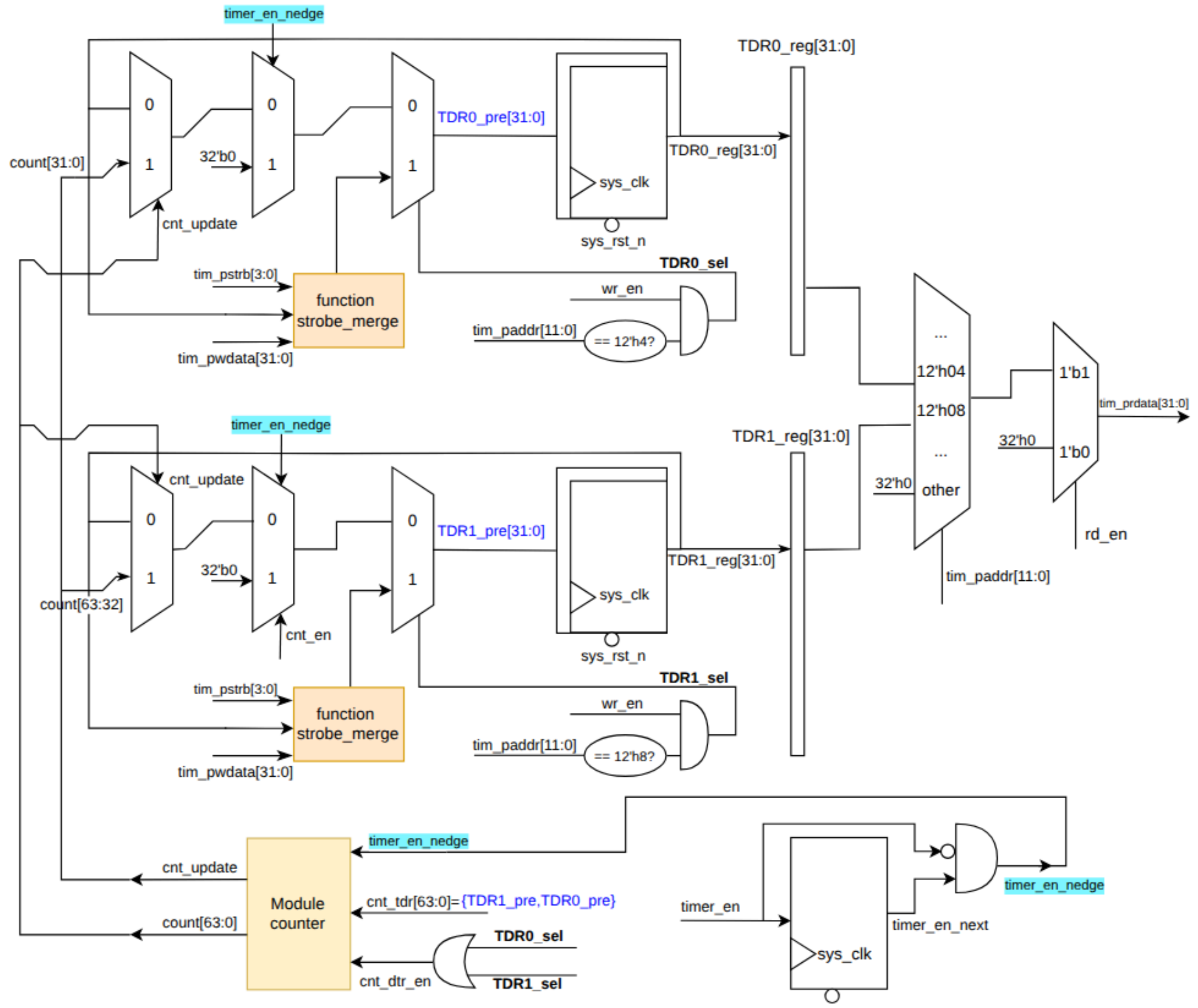


Figure 5: Block diagram of TDR register

- timer_en_edge is asserted when timer_en transitions from H -> L, at this time, the values of TDR0,1 are reset and sent to the counter module to reset the counter module.
- When the module counter sends cnt_update=1, TDR0,1 will update data from the count signal.
- cnt_dtr_en signal sent to the counter module to enable updating TDR0,1 value to counter value.
- TDR0_sel and TDR1_sel used to determine write access to the register.

3.2.3. TCMP register

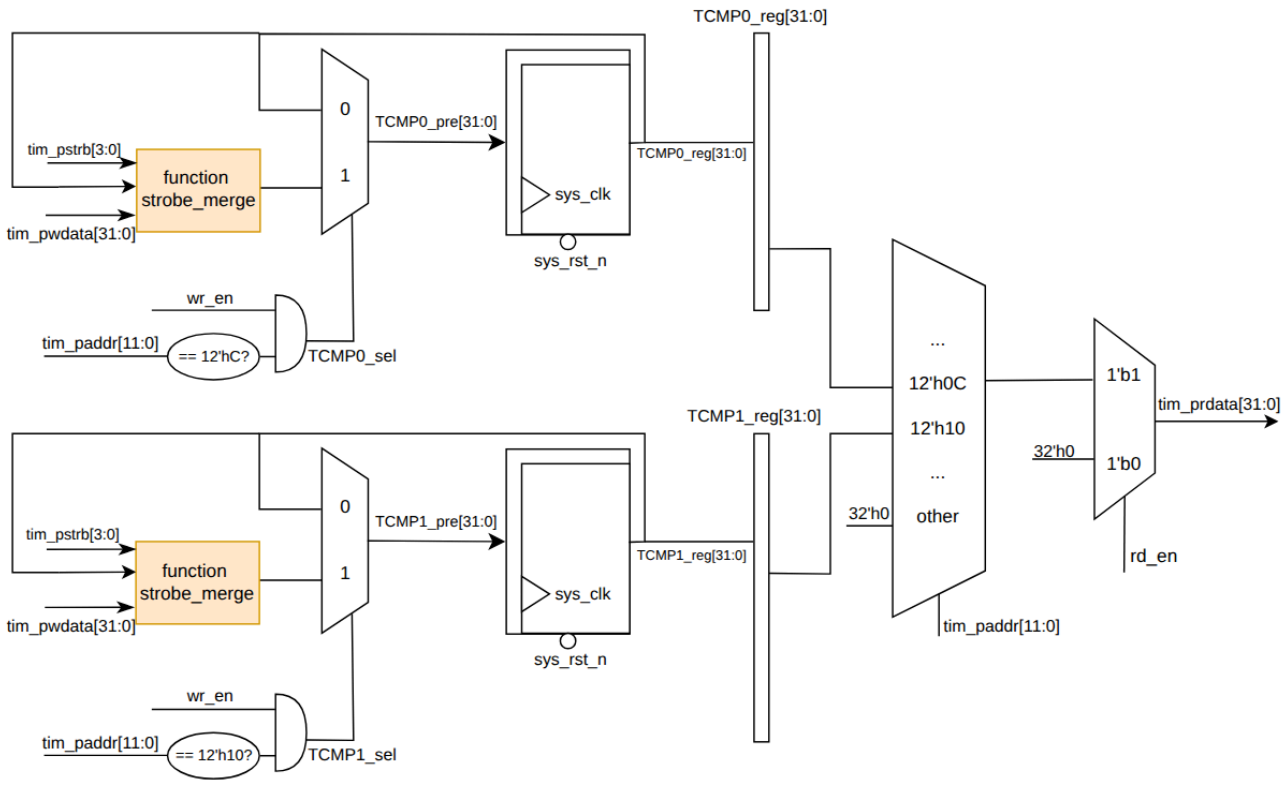


Figure 6: Block diagram of TCMP register

- This register contains data used to compare with the counter value.
- TCMP0_sel and TCMP1_sel used to determine write access to the register.

3.2.4. TIER and TISR register

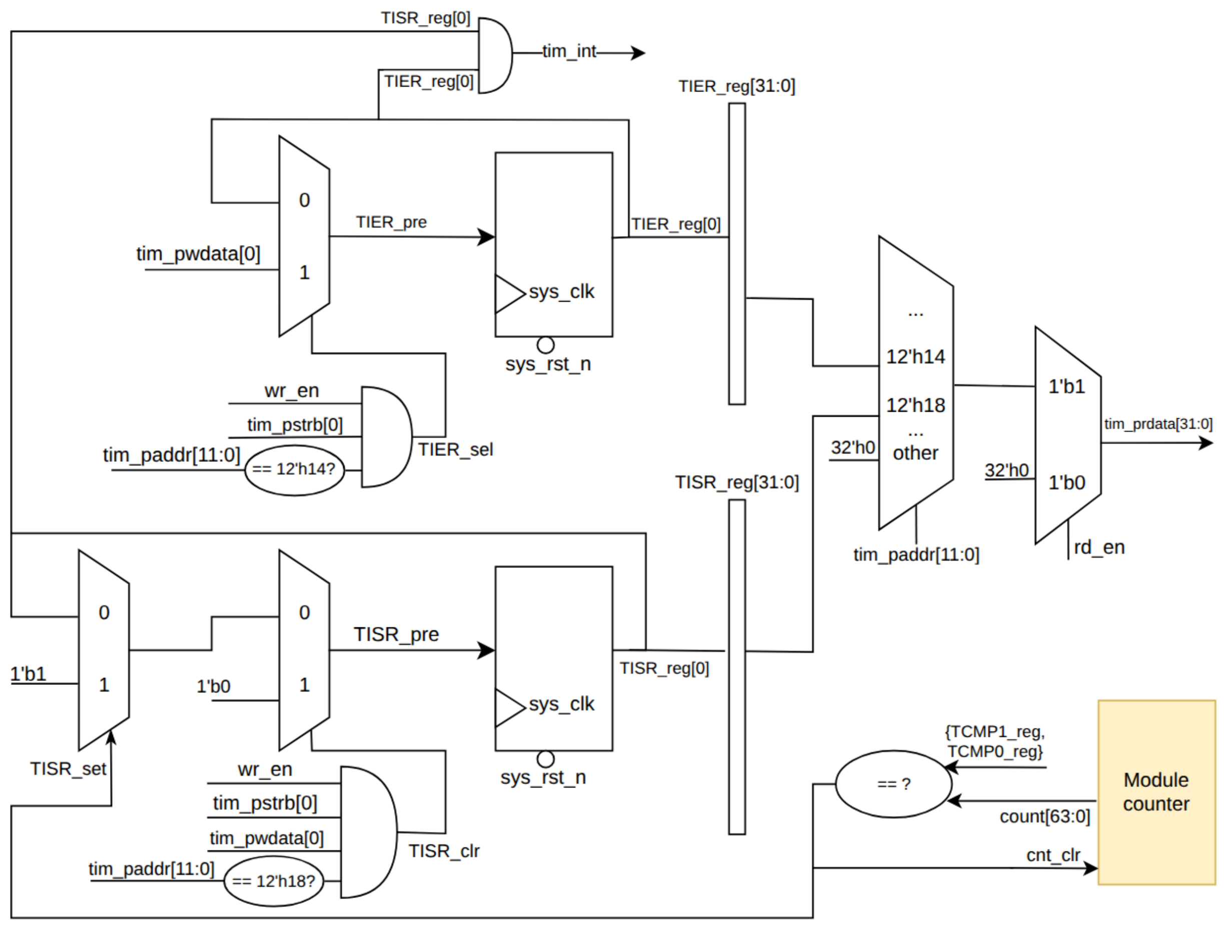


Figure 7: Block diagrams of TIER and TISR register

TIER register:

- Contains data used to control whether the interrupt is hidden or appears on the tim_int port.
- TIER_sel is asserted when wr_en=1, addr=12'h14 and pstrb[0]=1. When TIER_sel is asserted, data is written to register.

TISR register:

- When count[63:0] equal to {TCMP0,TCMP1}, the TISR signal_set is assert, write 1'b1 to the TISR register.
- To clear interrupt: wr_en=1, wdata[0]=1, addr=12'h18 and pstrb[0]=1.
- Cnt_clr signal sent to the counter module, used to clear counter when interrupt is asserted or timer_en signal change H->L.

3.2.4. THCSR register

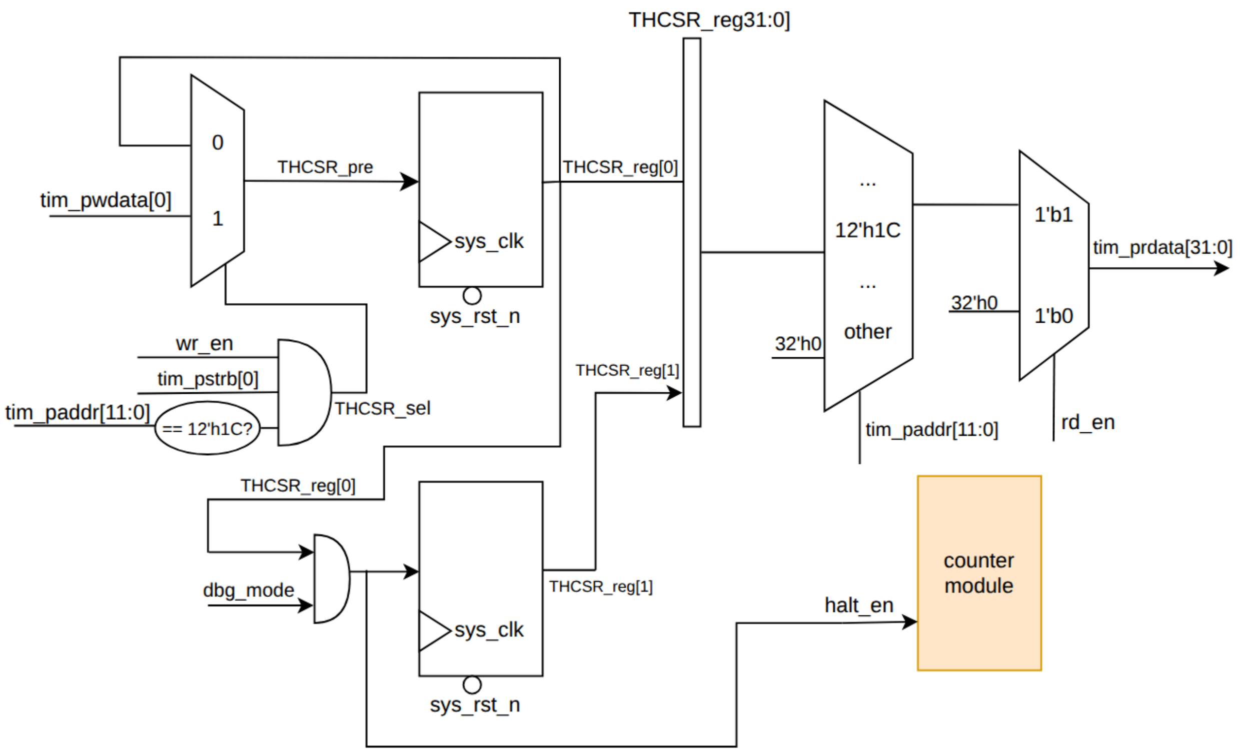


Figure 8: Block diagram of THCSR register

- THCSR_sel is asserted when wr_en=1, addr=12'h1C and pstrb[0]=1. When TIER_sel is asserted, data is written to register.
- halt_ack (THCSR[2]) is asserted only when dgb_mode=1 and halt_req(THCSR[0])=1.
- halt_ack used to send to the counter module to put the counter module into halt mode.

3.3. Counter block

3.3.1. Sub counter

The sub counter block is used to divide the counting speed in control mode. The counter speed is set by the div_en and div_val values.

The working principle of the sub counter is to provide a count variable sub_cnt, which counts from 0 -> div_val-1; when the sub_cnt value = div_val-1 value, it sends a pulse to the main counter block.

a. Wave form

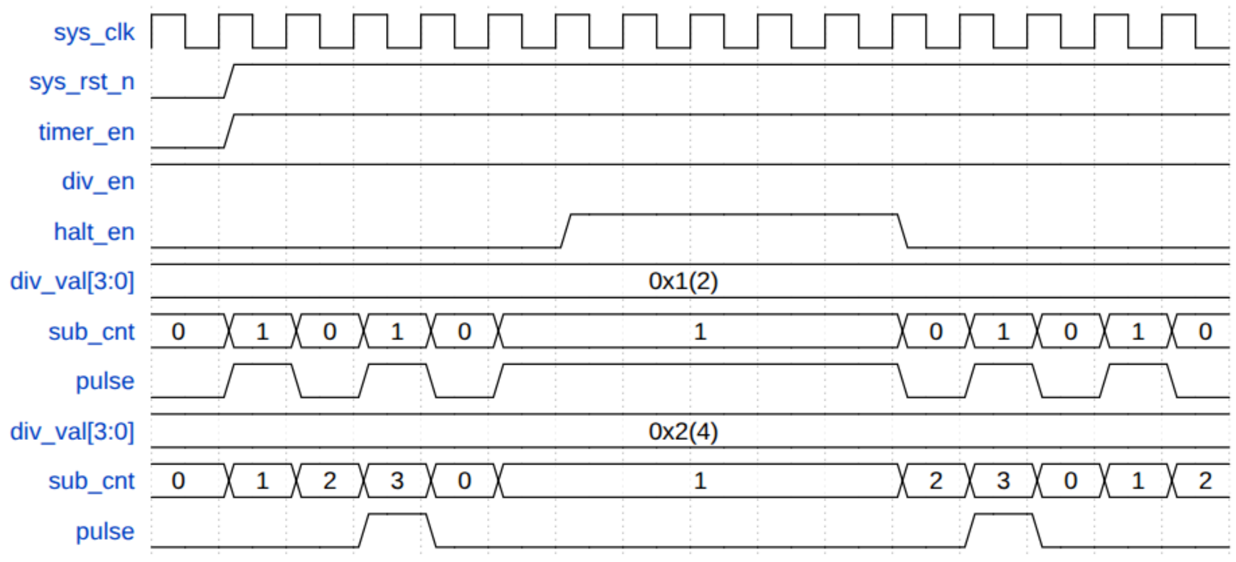


Figure 9: wave form of sub counter

- div_val =1 (Counting speed is divided by 2): when sub_cnt=1, a pulse will be sent to the main counter.
- Similarly with div_val=2 (Counting speed is divided by 4): when sub_cnt=3, a pulse will be sent to the main counter.
- When halt_en=1, sub_cnt will keep current value until halt_en=0.

b. Block diagram

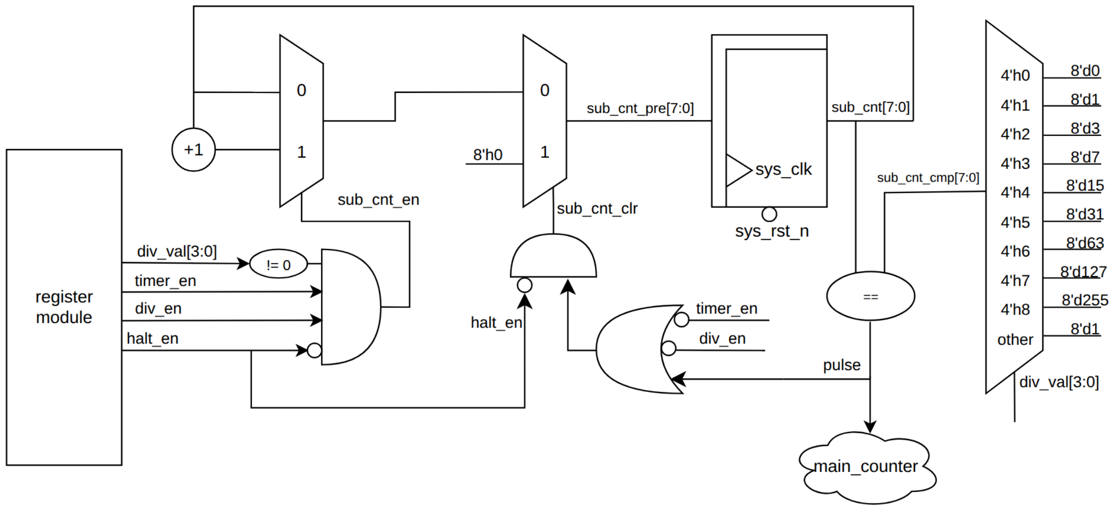


Figure 10: Block diagram of sub counter

- sub_cnt_en signal is asserted when timer_en =1, div_en =0, div_val !=0 and no halt mode (halt_en=0).
- sub_cnt_clr is asserted when timer_en=0 , div_en =0 and pulse is asserted. All the above conditions are accompanied by no halt mode (halt_en=0).
- When sub_cnt equals sub_cnt_cmp, a pulse will be sent to the main counter.

3.3.2. Main counter

The main counter block is used to count up value.

When the counter is in default mode or control mode with div_val=0, sub counter is disabled, cnt_en always is 1, the counter speed based on system clock.

When the counter is in control mode !=0, the main counter receives a pulse from the sub counter. After that, the counter will increment by 1.

When the counter receives the cnt_clr signal =1 from the register module, the counter will clear count value to 0.

When the counter receives the cnt_tdr_en signal =1 from the register module, the counter will update cnt_tdr value to count value.

When timer_en changes from H->L, the counter is cleared to its initial value, and when timer_en is L->H again, the timer can work normally.

Counter can be halted (stopped) in debug mode, after halted, counter can be resumed to count normally.

a. Wave form

Default mode and control mode with div_val=0

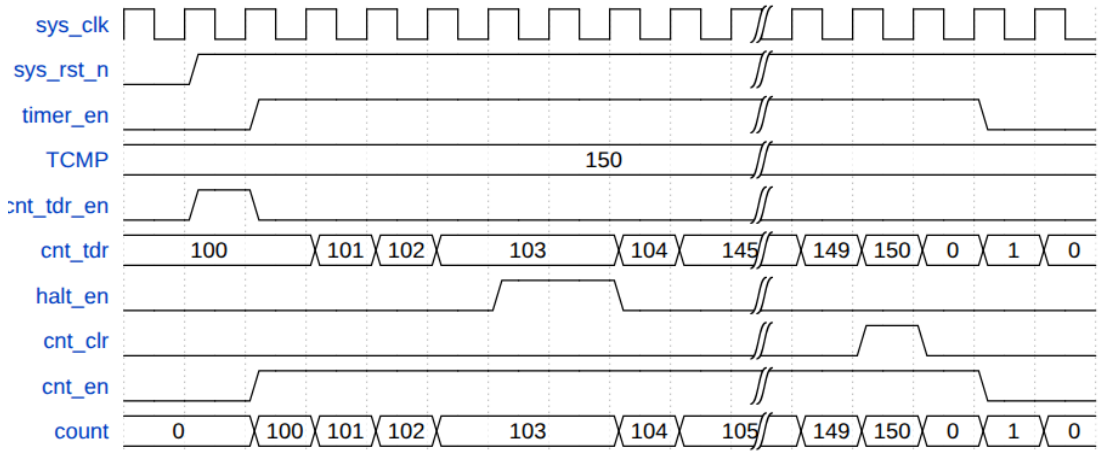


Figure 11: wave form default mode and control mode with div_val=0

- When cnt_tdr_en=1, cnt_tdr value will update to count value.
- Counter counts normal until halt_en=1, count will keep current value until halt_en=0.
- When receiving cnt_clr, the count value will reset to 0 in next cycles.
- When timer_en changes 1->0, the count value will reset to 0 in next cycles.

Control mode with div_val=1.

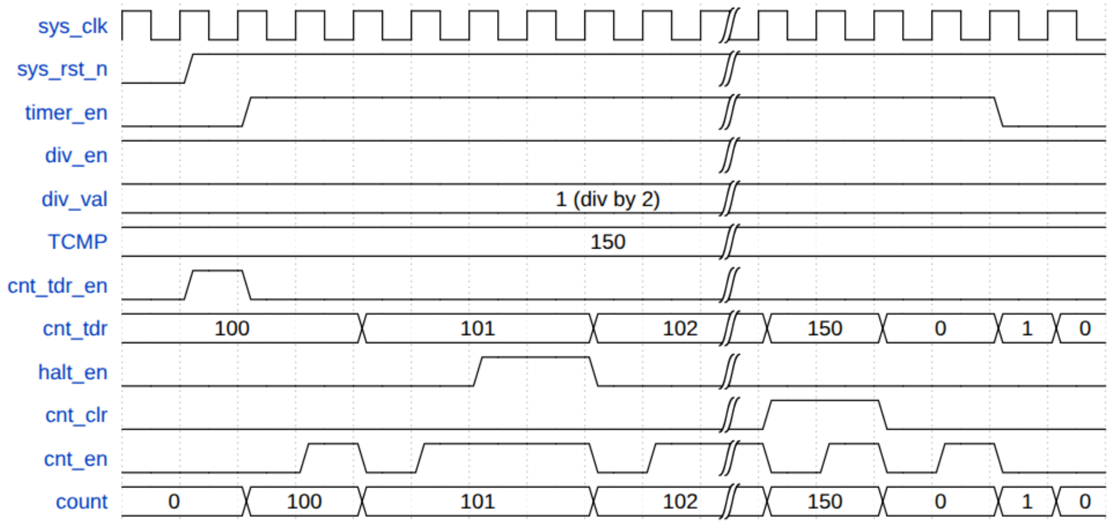


Figure 12: Control mode with div_val=1.

- When cnt_tdr_en=1, cnt_tdr value will update to count value.
- When receiving cnt_en, the counter will increment by 1 in next cycles.

- Counter counts normal until halt_en=1, count will keep current value until halt_en=0.
- When receiving cnt_clr, the count value will reset to 0 in next cycles.
- When timer_en changes 1->0, the count value will reset to 0 in next cycles.

b. Block diagram

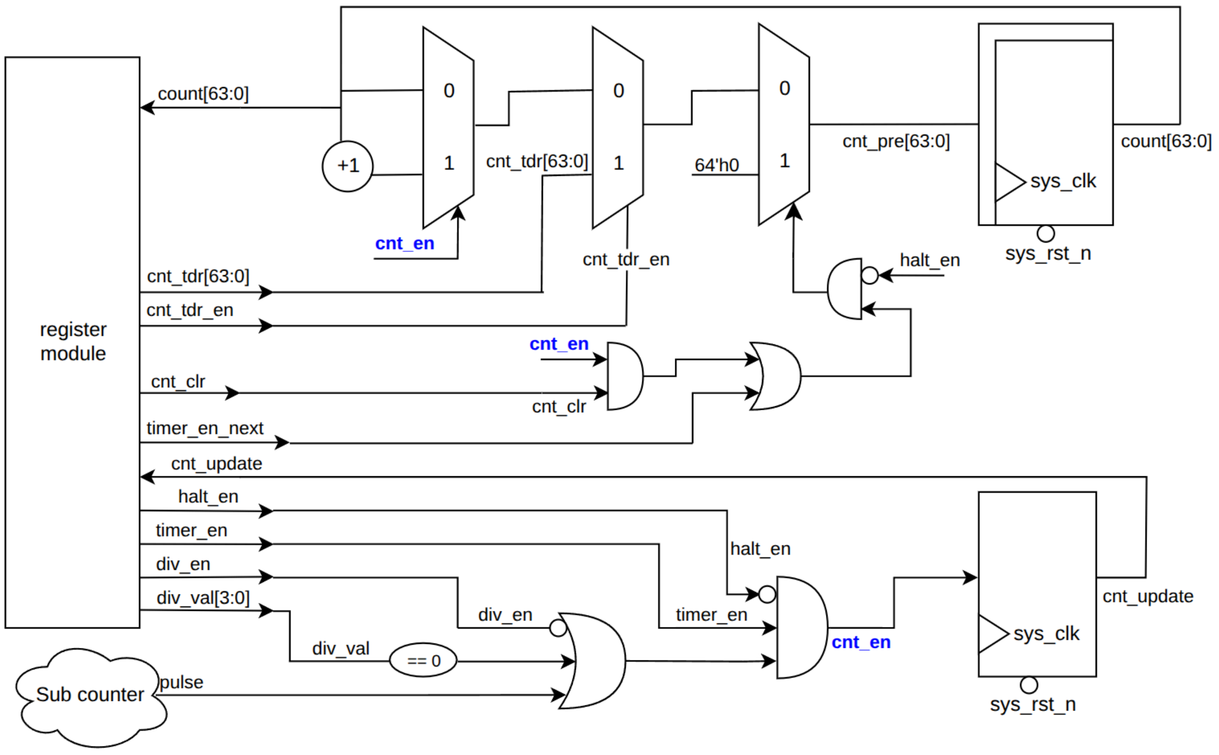


Figure 13: Block diagram of main counter

- Counter is cleared in 2 cases (all are accompanied by no halt):
  - When count is equal to TCMP(interrupt): both cnt_clr=1 and cnt_en=1(finishes the remaining cycles if it is in control mode).
  - When timer_en changes H->L: timer_en_next=1.
- cnt_en signal is asserted in 3 cases (all are accompanied by no halt):
  - Case 1, default mode: timer_en=1, div_en=0.
  - Case 2, control mode with div_val=0: timer_en=1, div_en=1, div_val==0;
  - Case 3, control mode with div_val !=0: timer_en=1, div_en=1, div_val !=0, pulse=1 (received by sub counter).
- cnt_update signal is sent to the register module to update count value to TDR register.

4. History

Date	Version	Description	Author
20/12/2025	1.0	Newly created	Nguyễn Tân Thành