IC TRAINING CENTER VIET NAM

FUNDAMENTAL IC DESIGN AND VERIFICATION COURSE

FINAL PROJECT

TIMER IP DESIGN SPECIFICATION

Student: Phan Lê Minh

Class: IC24

Instructor: Mr.Long

Contents

LIST OF TABLES

LIST OF FIGURES

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 system clock or divided up to 256.
- Support timer interrupt (can be enabled or disabled).
- Timer is an essential module for every chip.
- In this project, a timer module is customized from CLINT module of industrial RISC-V architecture. It is used to generate interrupt based on user settings.

1.2. Main features

1. **Counter**

- 64 – bit count-up
- Counting mode:
    - Default mode: counter's speed is same as system clock.
    - Control mode: when enabled by writing 1 to TCR.div_en bit, the counter's speed is determined by the divisor value set in TCR.div_val.
- Counter continues counting when interrupts occurs.
- Counter continues counting when overflow occurs.
- Support halted mode describe in next page.

- When timer_en changes from High to Low, the counter is cleared to its initial value. When timer_en is Low to High, timer can normally.
- Note: the div_en and div_val is not related to frequency divisor (clock divider). Those settings only control the counter then to count.

2. Halted

- Counter can be halted (stopped) in debug mode when both below conditions occur:
  - Input debug_mode signal is High.
  - THCSR.halt_req is 1.
- THCSR.halt_ack is 1 after a halt request indicates that the request is accepted.
- After halted, 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 halt and not halt as described in the below waveform.

3. Timer Interrupt

- Timer interrupt (tim_int) is asserted(set) when interrupt is enabled and counter's value matched (equal) 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.

4. Counting mode

- In default mode, counting speed depend on system clock (same as div_val = 0 case as below waveform).
- The counting's 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 allow to change div_en and div_val while timer_en is High by an error reponse when user mistakenly accesses.

5. APB slave/ Register

- Read.write to reserved area is RAZ/WI (read ad zero, write ignored).
- Support byte access: bus can access to individual bytes in the register.
- Support wait state (1 cycle) 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 is operating.
- When error occurs, data is not written into register bit/fields.

1.3. Block diagram

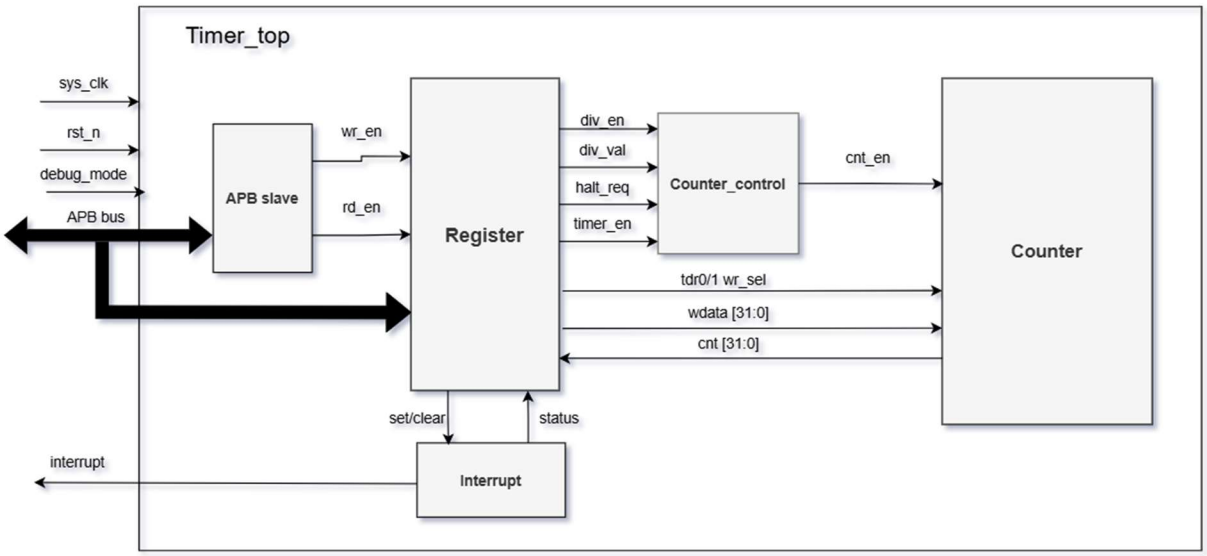


Figure 1: block diagram of Timer IP

1. APB slave
   The APB Slave block provides the interface betwrrn 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.
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.
3. Counting mode
   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.
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

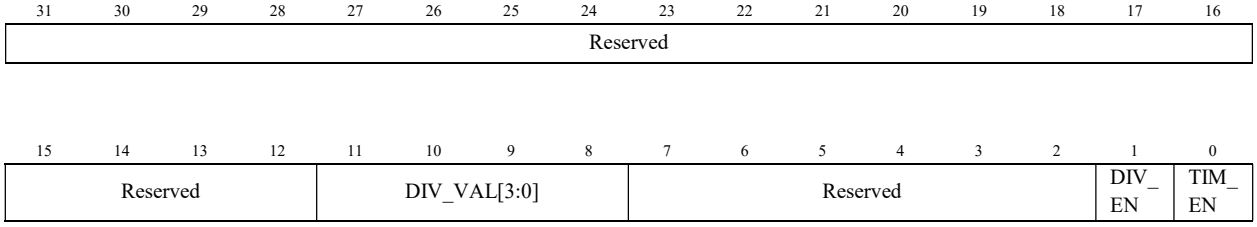


Table 1. Timer Control Register (TCR)

Bit	Name	Type	Default value	Description
31:12	Reserved	-	20'b0	Reserved
11:8	DIV_VAL[3:0]	RW	4'b0001	• 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: prohibit settings. When setting the prohibit value, div_val is not changed. Note: user must not change div_val while timer_en is High (*): add hardware logic to ensure div_val is prohibited to change when timer_en is High. Access is error response in this case. (*)access is "error response" when setting prohibit value to div_val
7:2	Reserved	RO	6'b0	Reserved
1	DIV_EN	RW	1/b0	Counter control mode enable. • 0: Disabled. Counter counts with normal speed based on system clock • 1: Enabled. The couting speed of counter is

				controlled based on div_val Note: user must not change div_en while timer_en is High (*): add hardware logic to ensure div_en is prohibited to change when timer_en is High. Access is error response in this case
0	TIM_EN	RW	1'b0	Timer enable • 0: Disabled. Counter does not count. • 1: Enabled. 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: 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	Lower 32-bit of 64-bit counter. 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: 0x08

Reset value: 32'h0000_0000

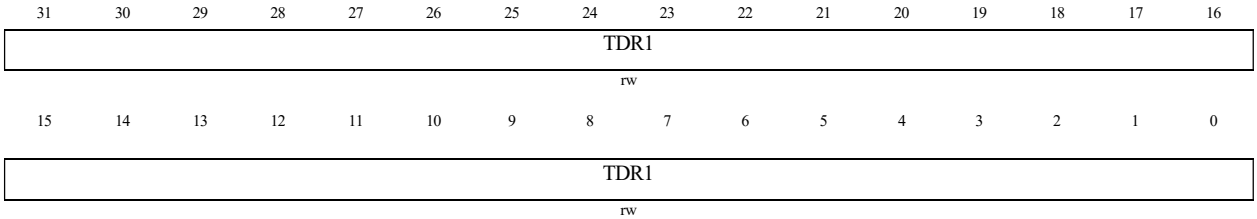


Table 5. Timer Data Register 1 (TDR1)

Bit	Name	Type	Default value	Description
31:0	TDR0	RW	32'h0000_0000	Lower 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

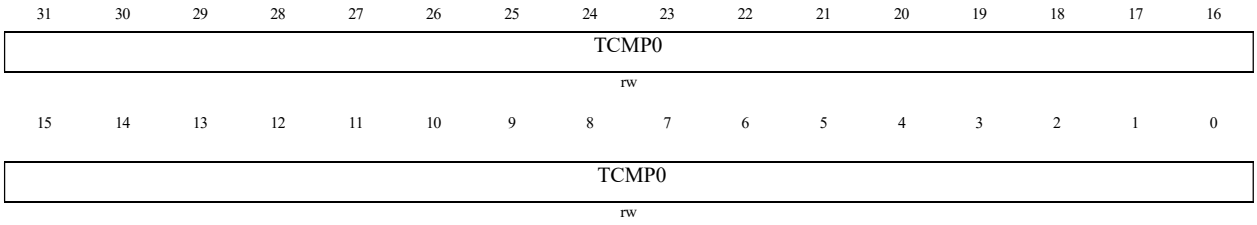


Table 6. Timer Compare Register 0 (TCMP0)

Bit	Name	Type	Default value	Description
31:0	TCMP0	RW	32'hFFFF_FFFF	Upper 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

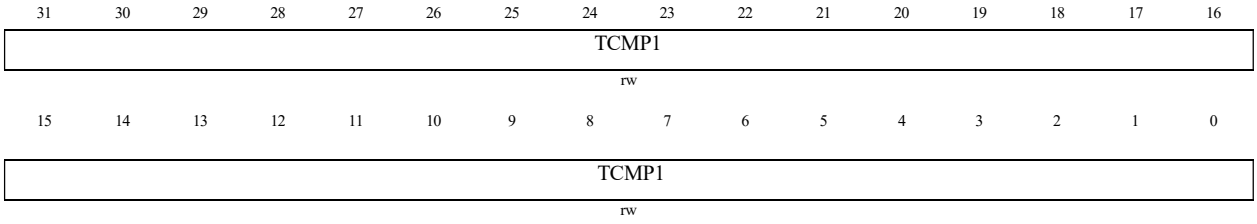


Table 7. Timer Compare Register 1 (TCMP1)

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

2.7. Timer Interrupt Enable Register (TIER)

Offset address: 0x14

Reset value: 32'h0000_0000

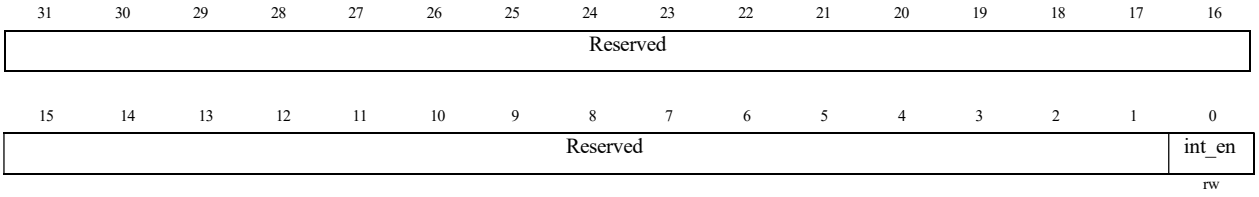


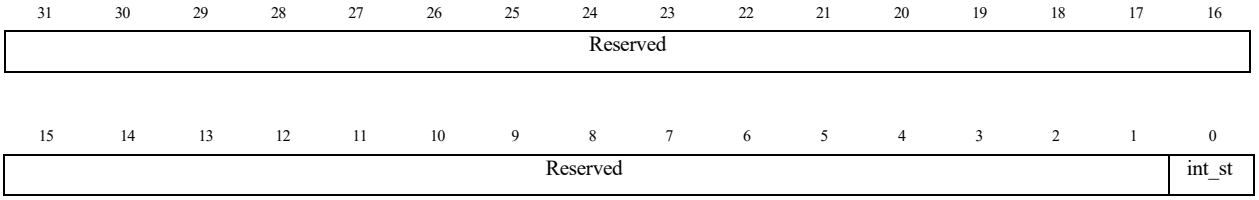
Table 8. Timer Interrupt Enable Register (TIER)

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



rw

Table 9. Timer Interrupt Status Register (TISR)

Bit	Name	Type	Default value	Description
31:1	Reserved	RO	31'h0	Reserved
0	int_st	R/W	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 Write to this bit when it is 0 has no effect. Note: When interrupt trigger condition occurred (counter reached compare value), counter continues to count normally.

2.9. Timer Halt Control Status Register (THCSR)

Offset address: 0x1C

Reset value: 32'h0000_0000

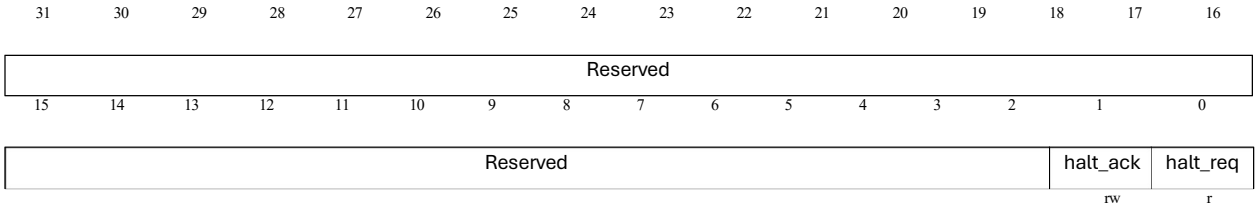


Table 10. Timer Halt Control Status Register (THCSR)

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.

```
{signal: [
                {name: 'pclk',          wave: '0p.............'},
                {name: 'prst_n',        wave: '0.1............'},
                {name: 'pwrite',                wave: '0.1..0.........'},
                {name: 'psel',                          wave: '0.1..0...1..0....'},
                {name: 'penable',                       wave: '0..1.0....1.0....'},
                {name: 'pready',                wave: '0...10.....10....'},
                {name: 'wr_en',                         wave: '0..1.0....1.0....'},
                {name: 'rd_en',                         wave: '0.............'},
]}
```

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.

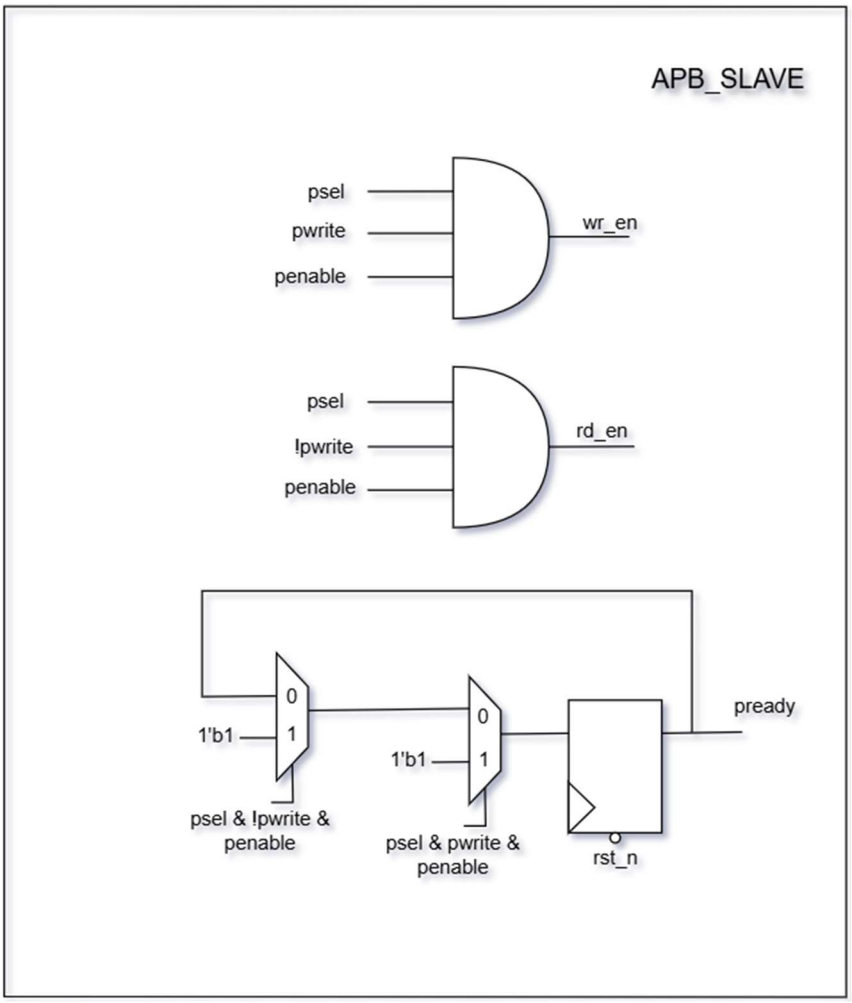


Figure 3. APB slave logic diagram

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.

```
{signal: [
        {name: 'sys_clk',          wave: '0p.....'},
        {name: 'sys_rst_n',        wave: '0.1.....'},
        {name: 'wr_en',                    wave: '0..1.0.1.0.1.0.1.0..'},
        {name: 'rd_en',                    wave: '0.....'},
  {name: 'addr[11:0]',                    wave: '=.....', data: ["12'h00"]},
  {name: 'pwdata[31:0]',                  wave: 'x..=.x.=.x.=.x.=.x..', data: ["32'h100",
"32'h101", "32'h103", "32'h303"]},
  {name: 'pstrb[3:0]',                    wave: 'x..=.x.=.x.=.x.=.x..', data: ["4'b0010",
"4'b0011", "4'b0011", "4'b0011"]},
  {name: 'pslverr',                       wave: '0.....1.0.1.0..'},
 ]}
```

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.

```
{signal: [
        {name: 'sys_clk',          wave: '0p.....'},
        {name: 'sys_rst_n',        wave: '0.1.....'},
        {name: 'wr_en',                    wave: '0......1.0.1.0.1.0..'},
        {name: 'rd_en',                    wave: '0..1.0.....'},
  {name: 'addr[11:0]',                    wave: 'x..=.x.=.x.=.x.=.x..', data: ["12'h18", "12'h14",
"12'h18", "12'h18"]},
  {name: 'pwdata[31:0]',                  wave: 'x.....=.x.=.x.=.x..', data: ["32'h01", "32'h00",
"32'h01"]},
  {name: 'rdata[31:0]',                   wave: 'x..=.x.....', data: ["32'h01"]},
  {name: 'pstrb[3:0]',                    wave: 'x..=.x.=.x.=.x.=.x..', data: ["4'b0001", "4'b0001",
"4'b0001", "4'b0001"]},
  {name: 'tim_int',                       wave: '0........1.......0..'},
 ]}
```

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.

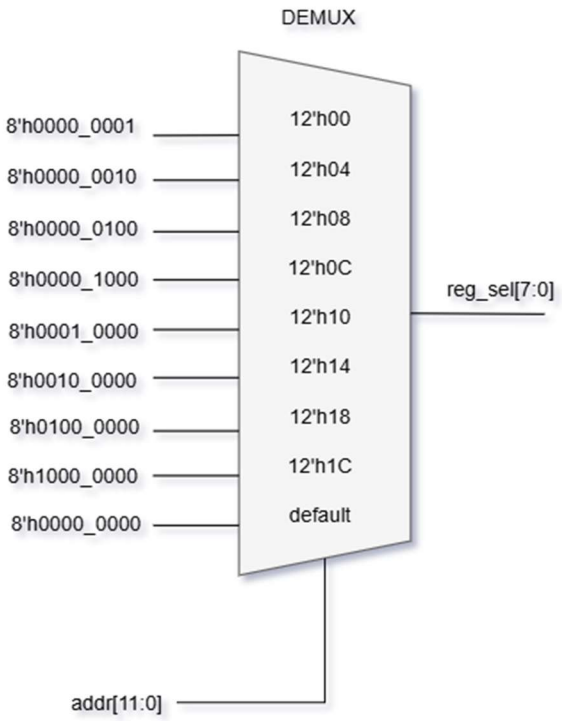


Figure 6: Logic diagram of the address decoder

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

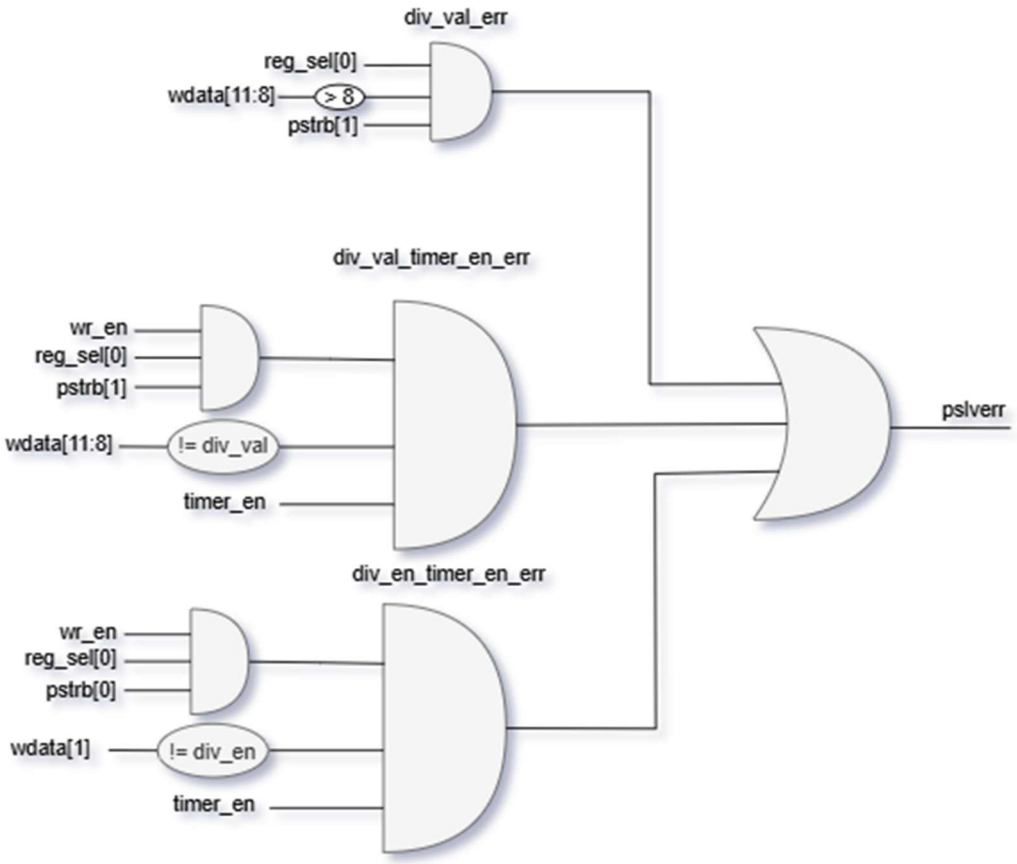


Figure 7: Logic diagram of the pslverr signal

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.

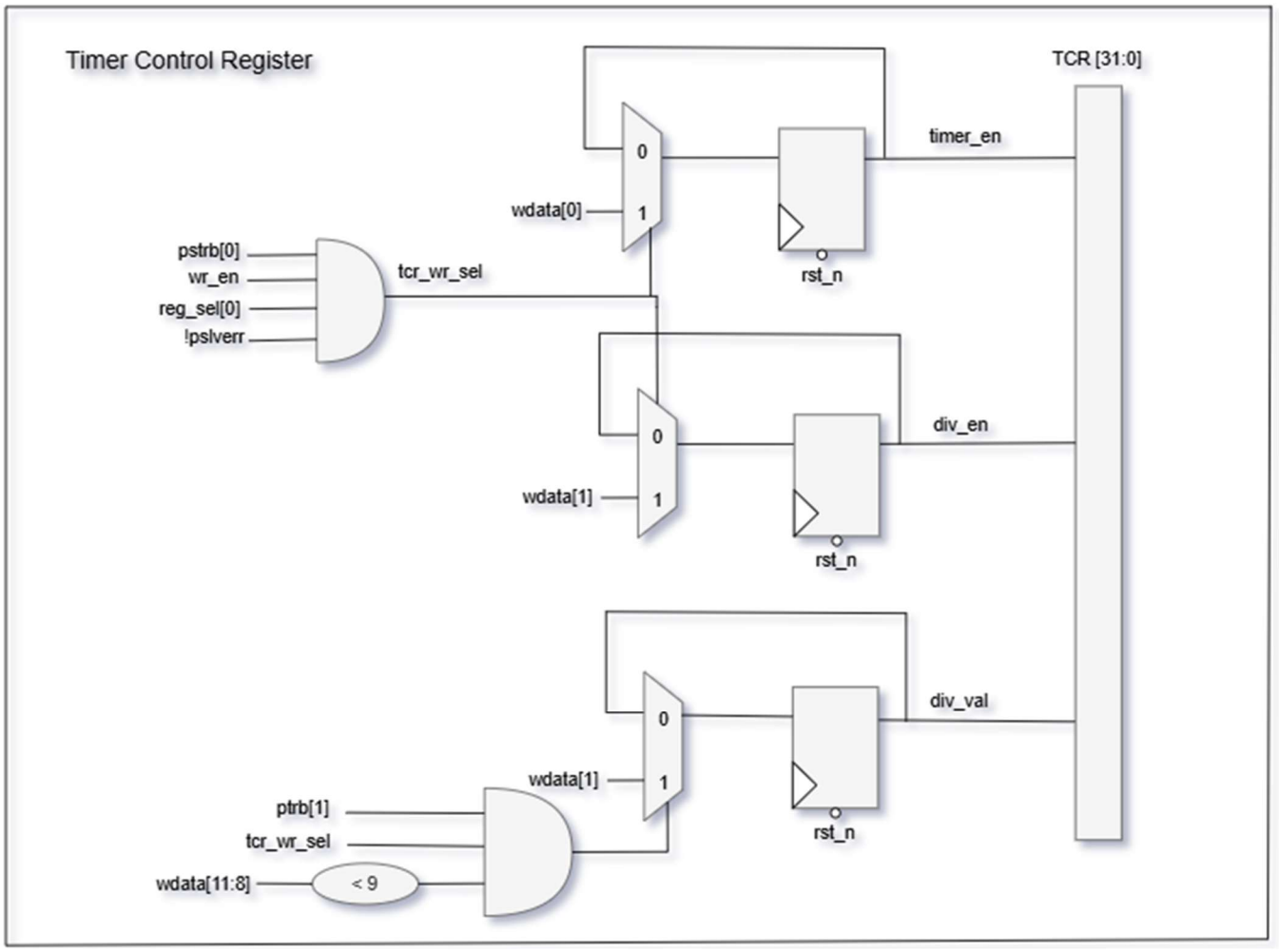


Figure 8: Logic diagram of TCR Register

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.

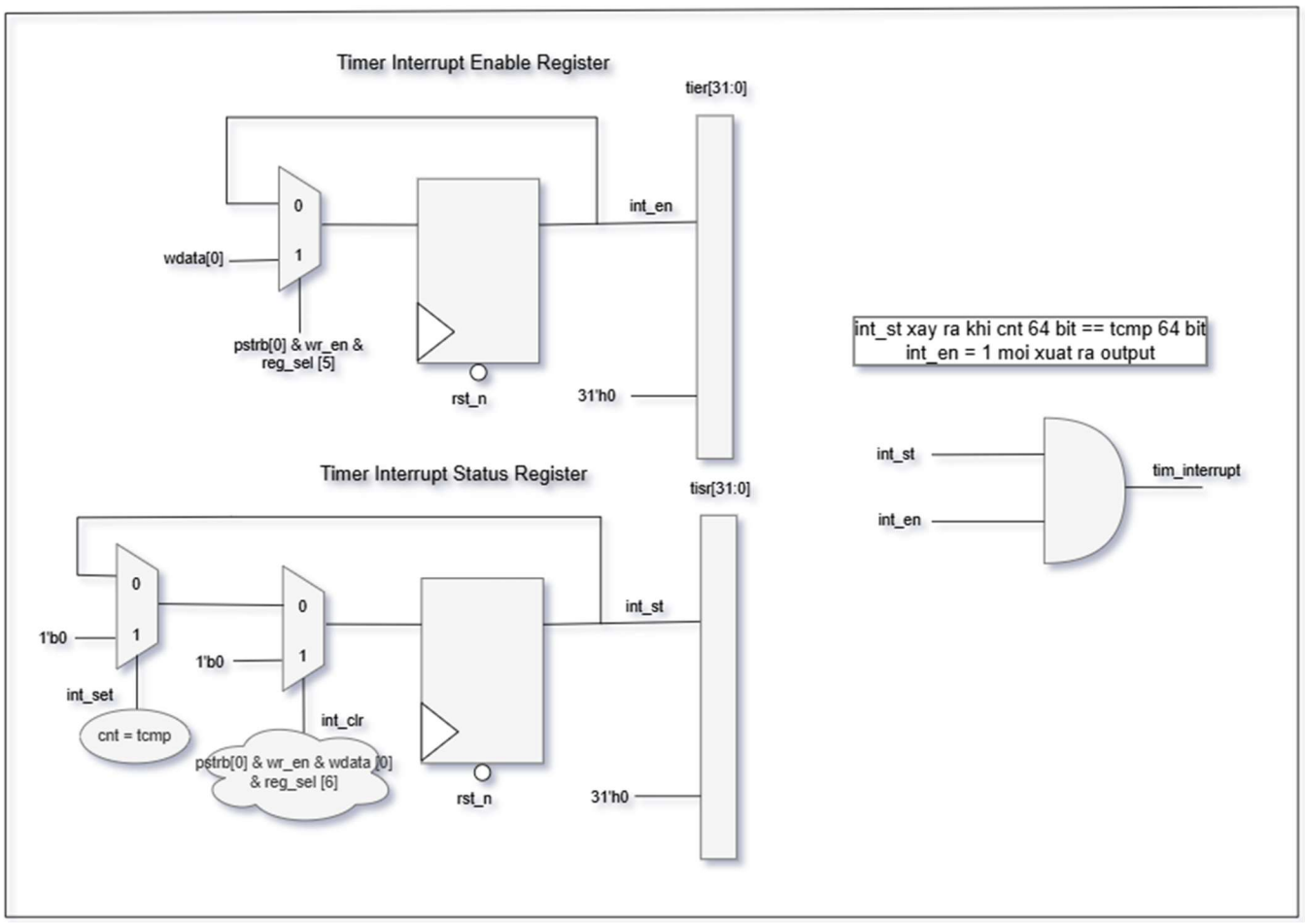


Figure 9: Logic diagram of TIER Register, TISR Register, tim_int

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.

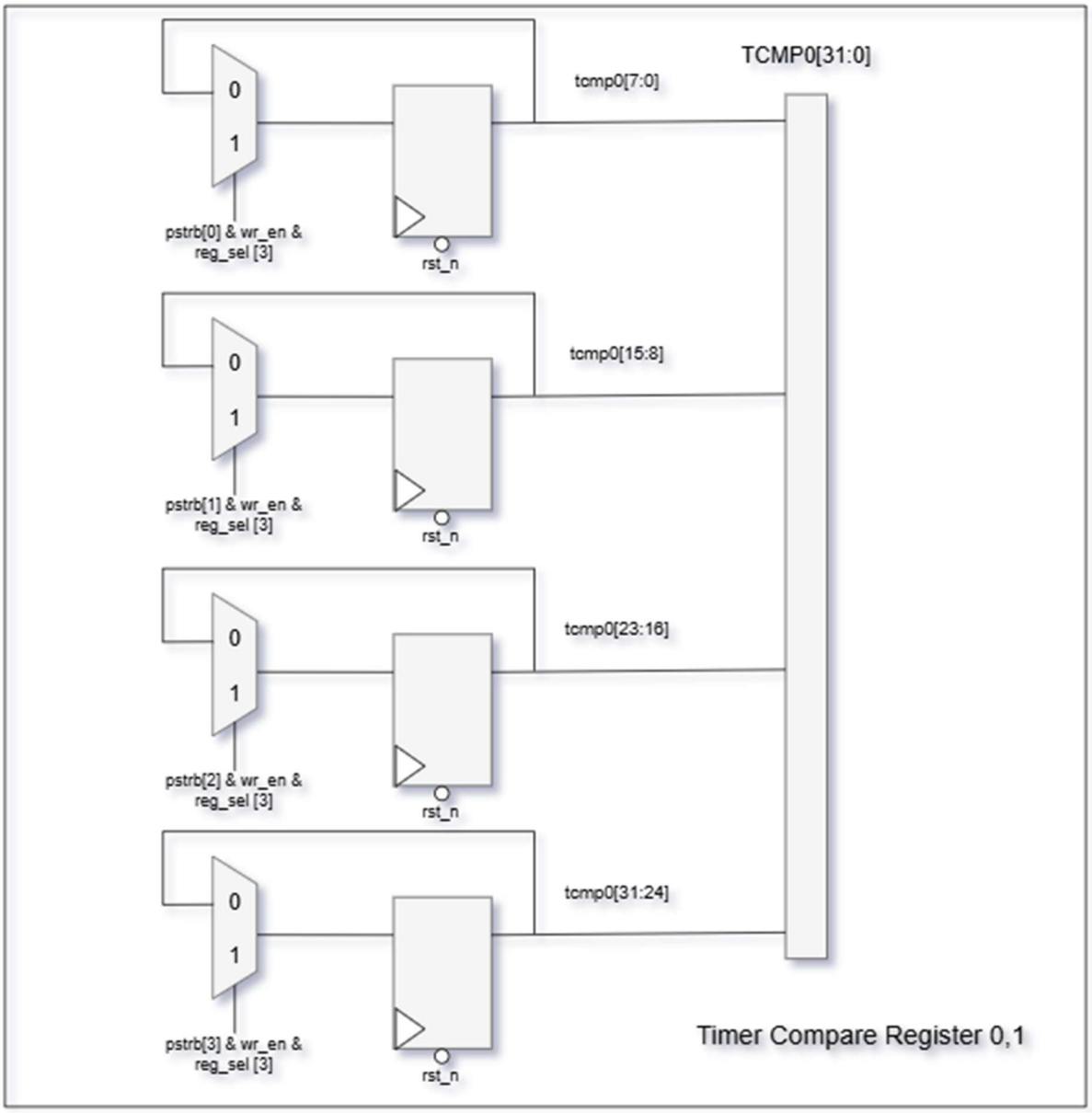


Figure 10: Logic diagram of TCMP0,1 Register

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.

```
{signal: [
        {name: 'sys_clk',          wave: '0p.....'},
        {name: 'sys_rst_n',        wave: '0.1.....'},
        {name: 'div_en',                wave: '0...1.....'},
  {name: 'div_val[3:0]',                wave: 'x...=.....', data: ["4'b0010
(divided by 4)"]},
  {name: 'timer_en',                          wave: '0...1.....',},
  {name: 'cnt_en',                             wave: '0.....10...10...10...10....',},
  {name: 'halt_req',                          wave: '0.....1..'},
  {name: 'cnt[63:0]',                         wave: '=.....=....=....=....=....', data:
["0", "1", "2", "3", "4"]},
 ]}
```

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.

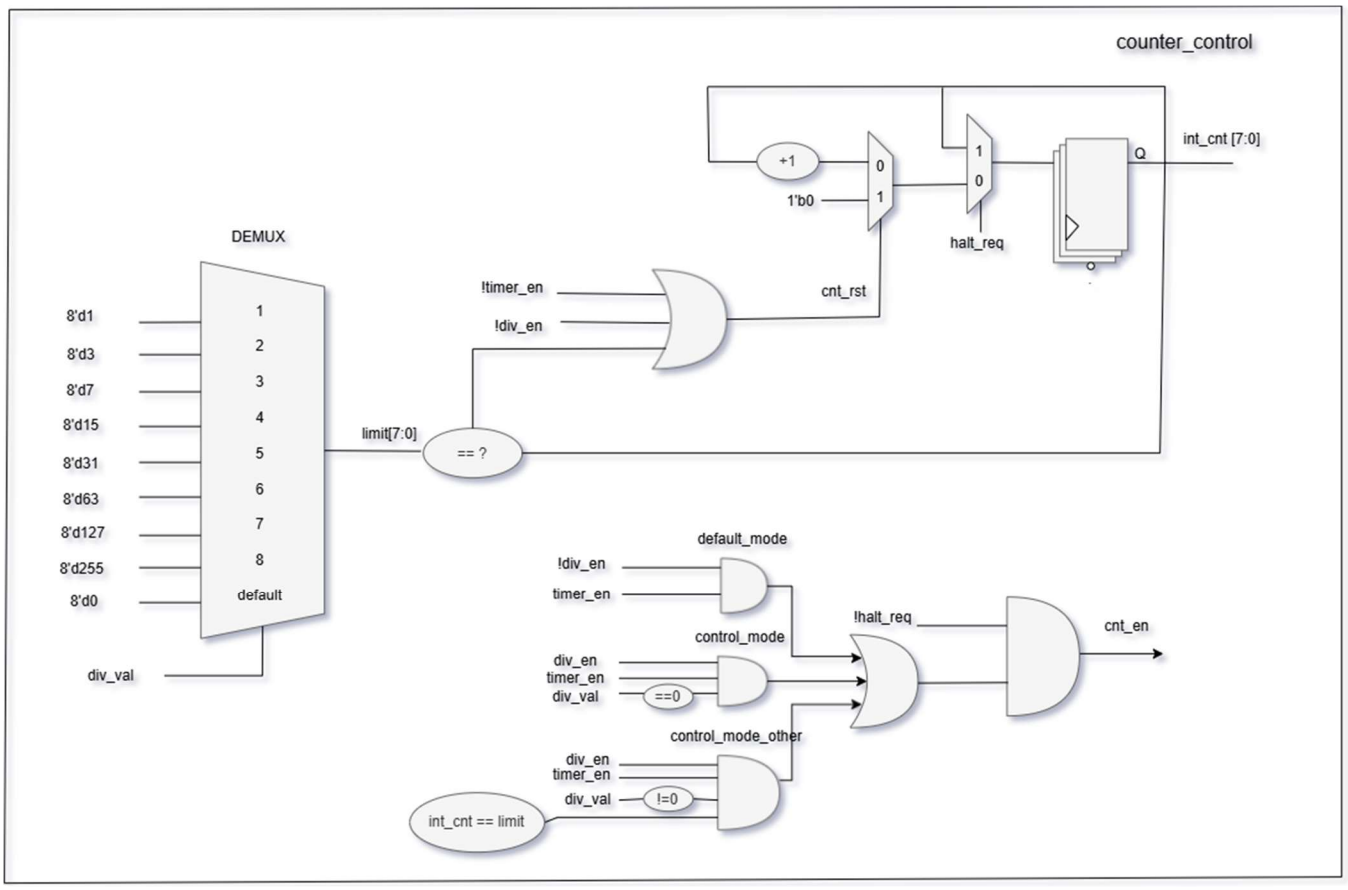


Figure 12: Logic diagram of Counter control

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.

```
{signal: [
        {name: 'sys_clk',          wave: '0p.....'},
        {name: 'sys_rst_n',        wave: '0.1.....'},
  {name: 'wr_sel[1:0]',                wave: 'x...=.x.....', data: ["2'b01"]},
  {name: 'pstrb[3:0]',                    wave: 'x...=.x.....', data:
["4'b0001"]},
  {name: 'pwdata[31:0]',                  wave: 'x...=.x.....', data:
["32'h01"]},
  {name: 'timer_en',                      wave: '0.....1.....'},
  {name: 'cnt_en',                          wave: '0.....10.10.10.10.....',},
  {name: 'cnt[63:0]',                      wave: '=.....=..=..=..=....', data: ["0",
"1", "2", "3", "4"]},
 ]}
```

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.

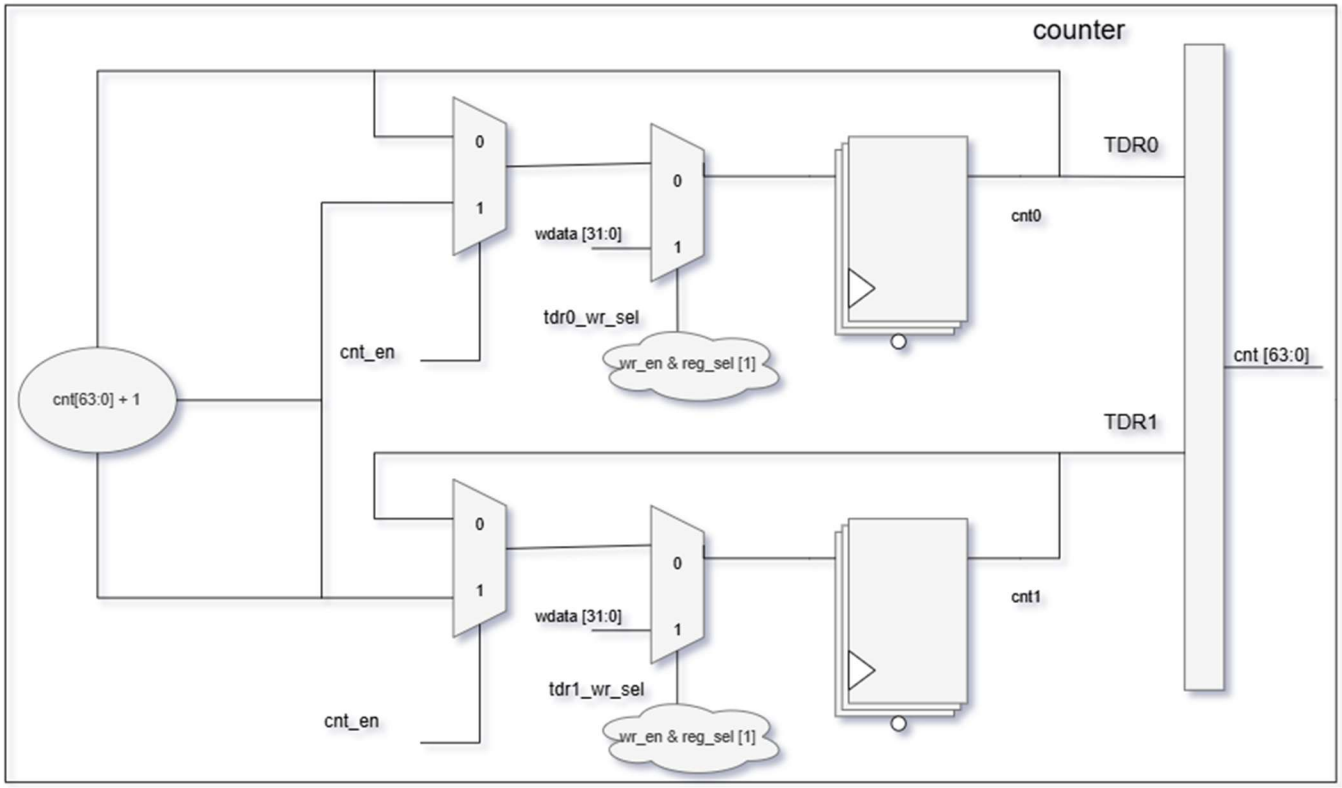


Figure 14: Logic diagram of Counter

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. History

Date	Version	Description	Author
	1.0	Newly created	Minh Phan