Index

[I. Overview 3](#_Toc177732634)

[1. Timer 3](#_Toc177732635)

[2. Function 3](#_Toc177732636)

[3. Counter 3](#_Toc177732637)

[4. APB slave/ Register 4](#_Toc177732638)

[II. Block diagram 4](#_Toc177732639)

[III. IO list 4](#_Toc177732640)

[IV. Fuction Description 5](#_Toc177732641)

[1. APB slave 5](#_Toc177732642)

[2. Register 7](#_Toc177732643)

[2.1 Write/read transfer 7](#_Toc177732644)

[2.2 Byte access 9](#_Toc177732645)

[2.3 Error handling 12](#_Toc177732646)

[3. Counter control 14](#_Toc177732647)

[4. Counter 15](#_Toc177732648)

[5. Timer interrupt 17](#_Toc177732649)

[6. Halted mode 18](#_Toc177732650)

[V. Register Desciptioin 20](#_Toc177732651)

[1. Register summary 20](#_Toc177732652)

[2. Register specification 20](#_Toc177732653)

[2.1 Timer Control Register -TCR 20](#_Toc177732654)

[2.2 Timer Data Register 0 –TDR0 22](#_Toc177732655)

[2.3 Timer Data Register 1 –TDR1 22](#_Toc177732656)

[2.4 Timer Compare Register 0 –TCMP0 22](#_Toc177732657)

[2.5 Timer Compare Register 1 –TCMP1 22](#_Toc177732658)

[2.6 Timer Interrupt Enable Register–TIER 23](#_Toc177732659)

[2.7 Timer Interrupt Status Register–TISR 23](#_Toc177732660)

[2.8 Timer Halt Control Status Register–THCSR 23](#_Toc177732661)

**Block diagram category**

[Block diagram 1: Timer\_IP 4](#_Toc177733366)

[Block diagram 2: APB slave 6](#_Toc177733367)

[Block diagram 3: Write/read transfer 7](#_Toc177733368)

[Block diagram 4: Byte access 11](#_Toc177733369)

[Block diagram 5: Error handling 13](#_Toc177733370)

[Block diagram 6: Counter control 15](#_Toc177733371)

[Block diagram 7: Counter 16](#_Toc177733372)

[Block diagram 8: Timer interrupt 18](#_Toc177733373)

[Block diagram 9: Halt mode 20](#_Toc177733374)

**Waveform category**

[Waveform 1: APB slave 7](#_Toc179998731)

[Waveform 2: Write/read transfer 9](#_Toc179998732)

[Waveform 3: Register TCR 11](#_Toc179998733)

[Waveform 4: Error handling 15](#_Toc179998734)

[Waveform 5: Counter control 16](#_Toc179998735)

[Waveform 6: Counter enable 16](#_Toc179998736)

[Waveform 7: Counter default mode 17](#_Toc179998737)

[Waveform 8: Counter control mode 18](#_Toc179998738)

[Waveform 9: Counter 18](#_Toc179998739)

[Waveform 10: Timer interrupt 19](#_Toc179998740)

**Logic diagram category**

[Logic diagram 1: APB slave 6](#_Toc179998745)

[Logic diagram 2: Write transfer 7](#_Toc179998746)

[Logic diagram 3: Read transfer 8](#_Toc179998747)

[Logic diagram 4: Register TCR 10](#_Toc179998748)

[Logic diagram 5: Byte access (1) 12](#_Toc179998749)

[Logic diagram 6: Byte access (2) 12](#_Toc179998750)

[Logic diagram 7: Byte access 13](#_Toc179998751)

[Logic diagram 8: Error handling (1) 14](#_Toc179998752)

[Logic diagram 9: Error handling (2) 14](#_Toc179998753)

[Logic diagram 10: Counter 17](#_Toc179998754)

[Logic diagram 11: Timer interrupt 19](#_Toc179998755)

[Logic diagram 12: Halt mode 20](#_Toc179998756)

[Logic diagram 13: Halt mode 21](#_Toc179998757)

**Timer IP Specification**

1. Overview
2. Timer

* Timer is an essential module for every chip.
* This is used to generate accurate timing interval or controlling the timing of various  
  operations within the circuit. Timer can be used in various application: pulse  
  generation, delay generation, event generation, PWM generation, Interrupt  
  generation ….
* 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.
* The spec of CLINT can be referred at:  
  <https://chromitem-soc.readthedocs.io/en/latest/clint.html>

1. Function

* 64 bit timer count-up
* Address space: 4KB (0x4000\_1000 – 0x4000\_1FFF)
* Register set is configured via APB bus (IP is APB slave)
* APB 32 bit transfer:

+ Support wait state (1 cycle) and error handling

+ Support byte access

+ Support halt (stop) in debug mode

* System clock frequency is 200 Mhz. Timer uses active low async reset
* Counter can be counter based on system clock or divided up to 256
* Support timer interrupt (can be enable or disable)

1. Counter

* 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 counterk
* 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 L->H again, timer can work normally

Note: The div\_en and div\_val is not related to frequency divisor (clock divider). Those settings only control the counter when to count.

1. APB slave/ Register

* Address space: 4KB (0x4000\_1000 – 0x4000\_1FFF)
* Read/write to reserved area is RAZ/WI (read as zero, write ignored)
* System clock frequency is 200 MHz.
* Support byte access: bus can access to 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 is operating
* When error occurs, data is not written into register bit/field.

1. Block diagram

Counter  
Control

div\_en

Block diagram 1: Timer\_IP

**Timer\_top**

APB Slave

Register

sys\_clk

timer\_en

div\_val

cnt\_en

Counter

wr\_en

rd\_en

rst\_n

halt\_req

halt\_ask

addr

debug\_mode

wdata[31:0]

tdr\_0/1\_wr\_sel

rdata

wdata

APB\_bus

cnt[63:0]

int\_en

tim\_int

int\_st

cmp\_val[63:0]

tisr\_wr\_sel

Interrupt

1. IO list

|  |  |  |  |
| --- | --- | --- | --- |
| Signal name | Width | Direction | Description |
|  |  |  |  |
| sys\_clk | 1 | Input | Input clock |
| sys\_rst\_n | 1 | Input | Active low asynchronous reset  0: conter is in reset state  1: couter is in non\_reset state |
| tim\_psel | 1 | Input | Select signal  0: Timer is not selected, ABP transactions are disable  1: Timer is selected, ABP transactions are enable |
| tim\_pwrite | 1 | Input | Direction. 1: APB write access  0: APB read access |
| tim\_penable | 1 | Input | Enable signal  0: ABP transaction is not active  1: ABP transaction is active |
| tim\_paddr | 12 | Input | Address signal Address of register for read/write operations |
| tim\_pwdata | 32 | Input | Write data. Carries the data to be written to the slave during a write operation in an APB transaction. |
| dbg\_mode | 1 | Input | Debub mode  • 0: Disabled. Counter is not in debub mode.  • 1: Enabled. Counter is in debub mode |
| tim\_pstrb | 4 | Input | Write strobe. Indicates the byte lanes that are active for write operations in an APB transaction, specifying which bytes of the data are being written to the slave. |
| tim\_pready | 1 | Output | Ready signal  0: timer is not ready, transaction is still on progress  1: timer is ready, transaction can completed |
| tim\_pslverr | 1 | Output | Transfer error. 0: No error, transaction successful  1: Error occurred during the APB transaction |
| tim\_int | 1 | Output | Timer interrupt  • 0: Interrupt isn’t enabled or counter’s value is not match the compare counting  • 1: Interrupt is enabled and counter’s value matches the compare counting |
| tim\_prdata | 32 | Output | Read data. Carries the data read from the slave during a read operation in an APB transaction. |

1. Fuction Description
2. APB slave

APB slave

rd\_en

wr\_en

clk

psel

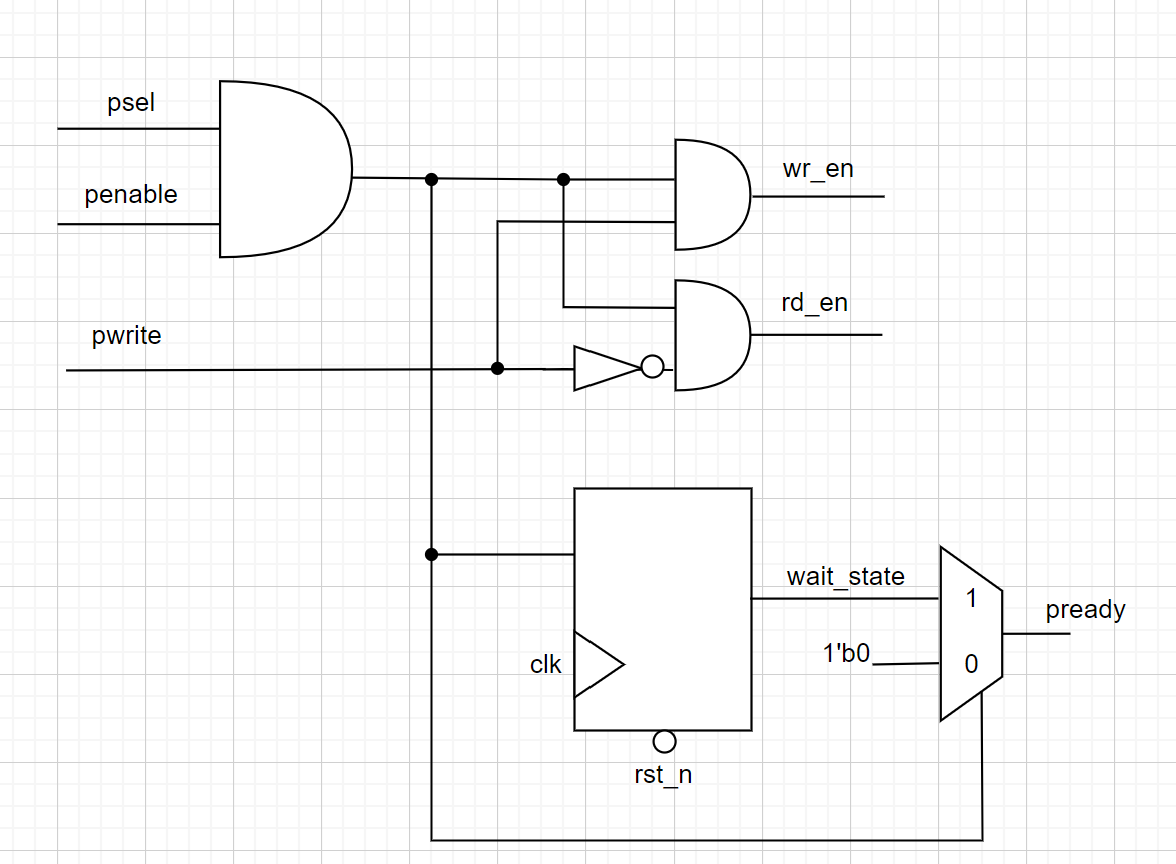
pwrite

rst\_n

pready

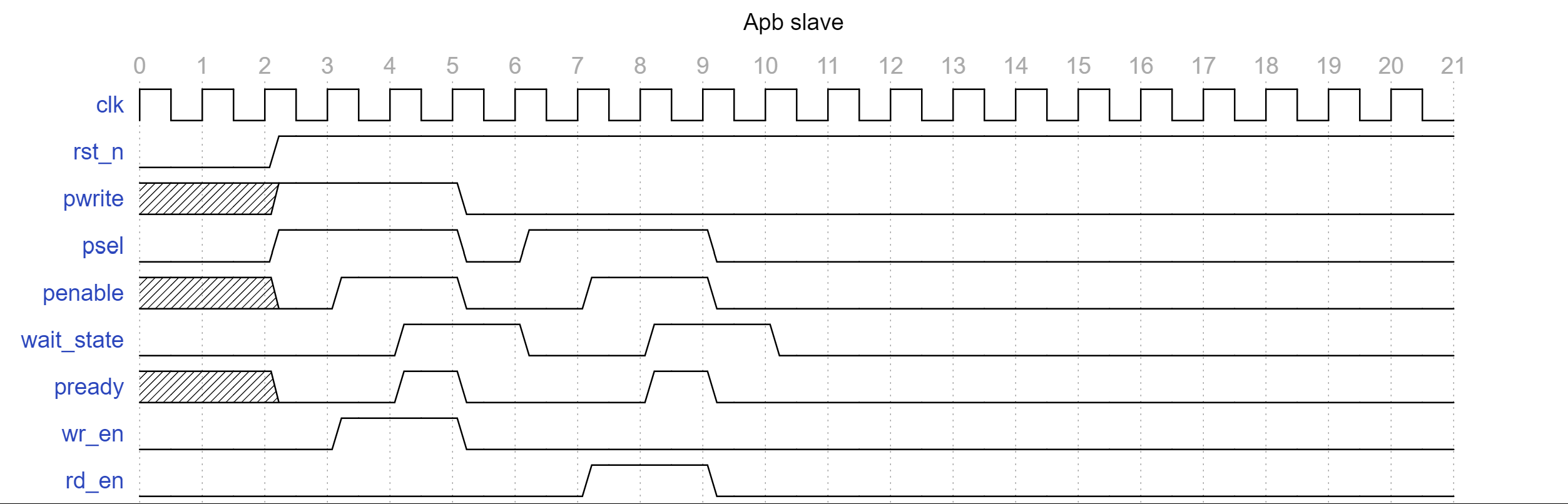
penable

Block diagram 2: APB slave



Logic diagram 1: APB slave

* The APB slave acts as a brigde between the APB bus and the internal register block. It decodes the address and control signals from the bus to determine whether to perform a read or write operation.
* Write enable or read enable is asserted when psel and penable is equal to 1
* Pready asserted after wr\_en or rd\_en 1 cycle (triggered when wait state = 1 and immediately negated when psel or penable = 1)



Waveform 1: APB slave

1. Register

### 2.1 Write/read transfer

Block diagram 3: Write/read transfer

Write/ read transfer

addr[7:0]

psel

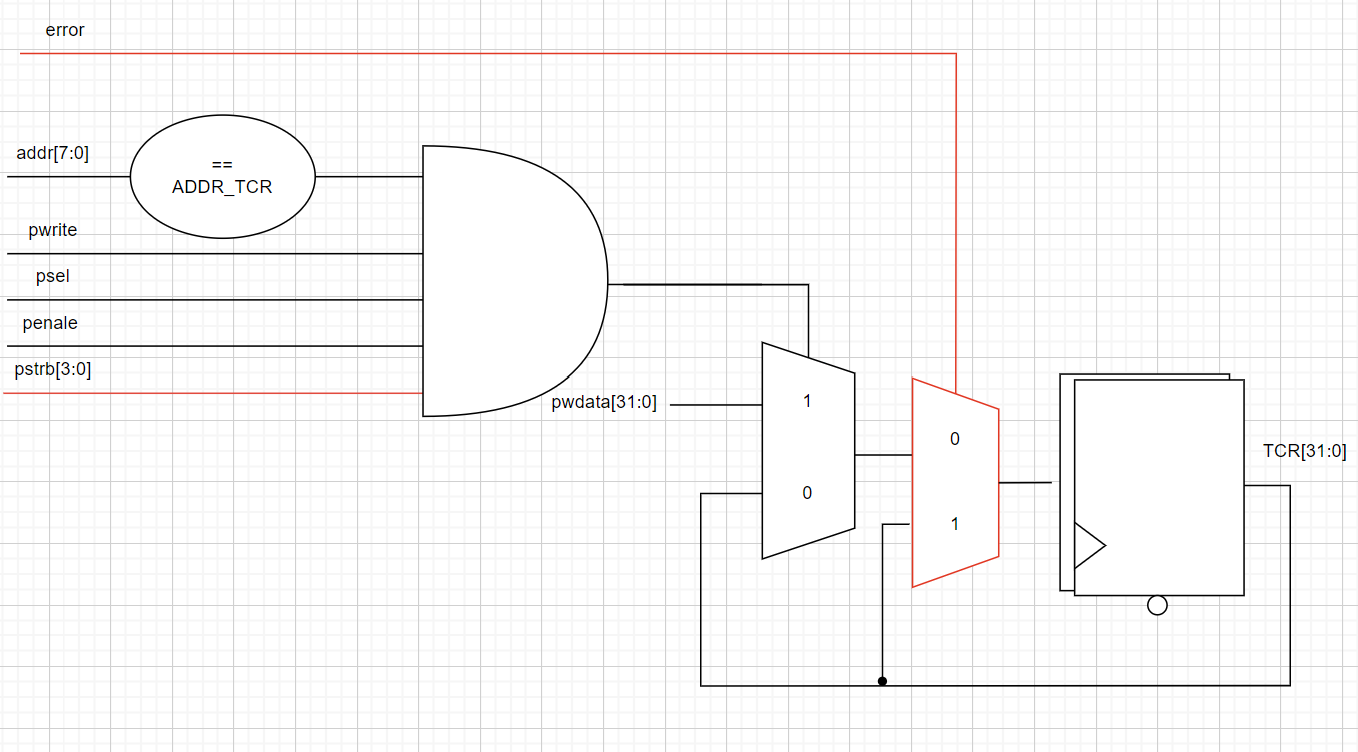
prdata[31:0]

penable

pwrite

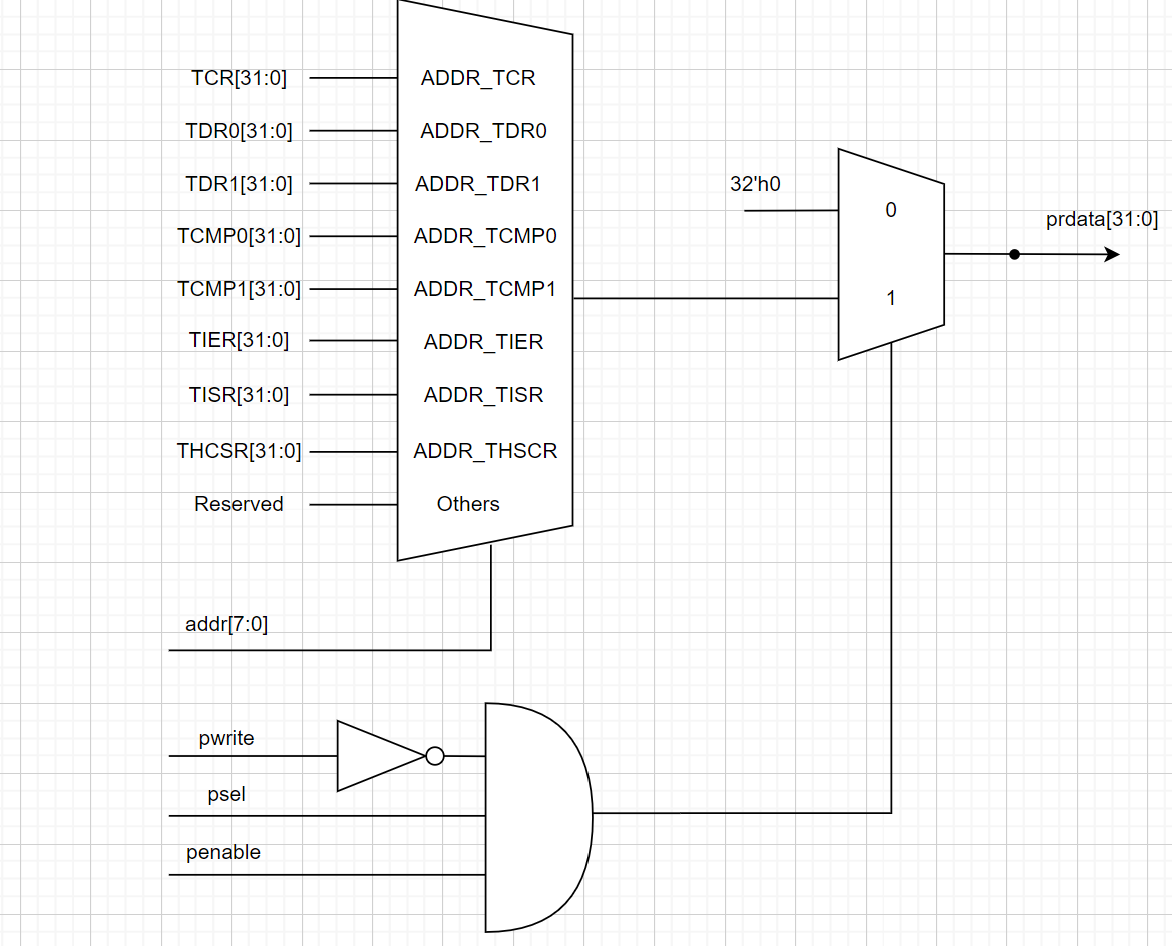
pwdata[31:0]

\*Write transfer



Logic diagram 2: Write transfer

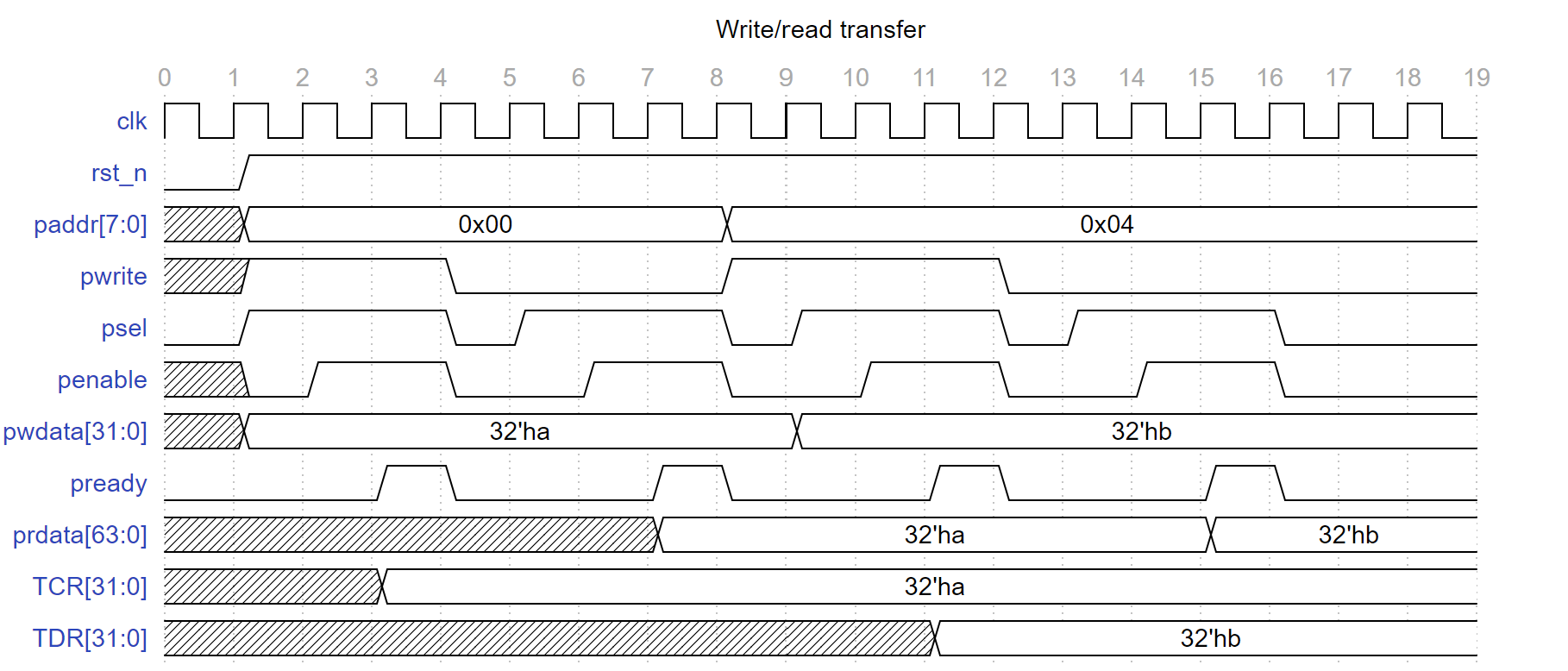
\*Read transfer



Logic diagram 3: Read transfer

- **Read Transfer**: In a read transfer, the APB master sends the peripheral address, and the peripheral returns the requested data via the PRDATA signal.

- **Write Transfer**: During a write transfer, the APB master sends data to a peripheral using the PWDATA signal along with the peripheral address and sets the PWRITE signal to indicate a write operation.



Waveform 2: Write/read transfer

### 2.2 Register TCR ( Timer Control Register)

wr\_en

Register TCR

timer\_en

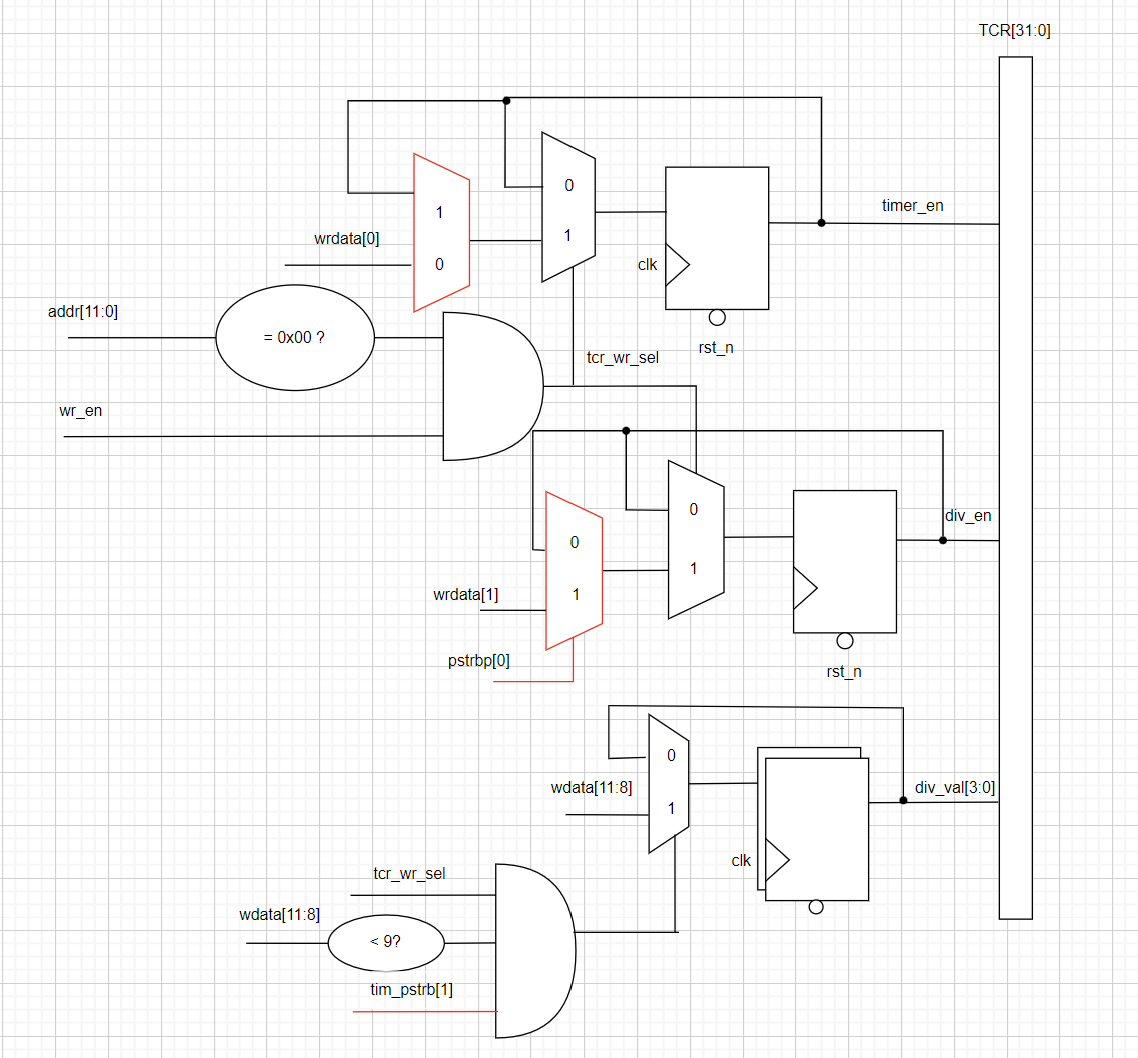
Block diagram 4: Register TCR

wdata[31:0]

div\_en

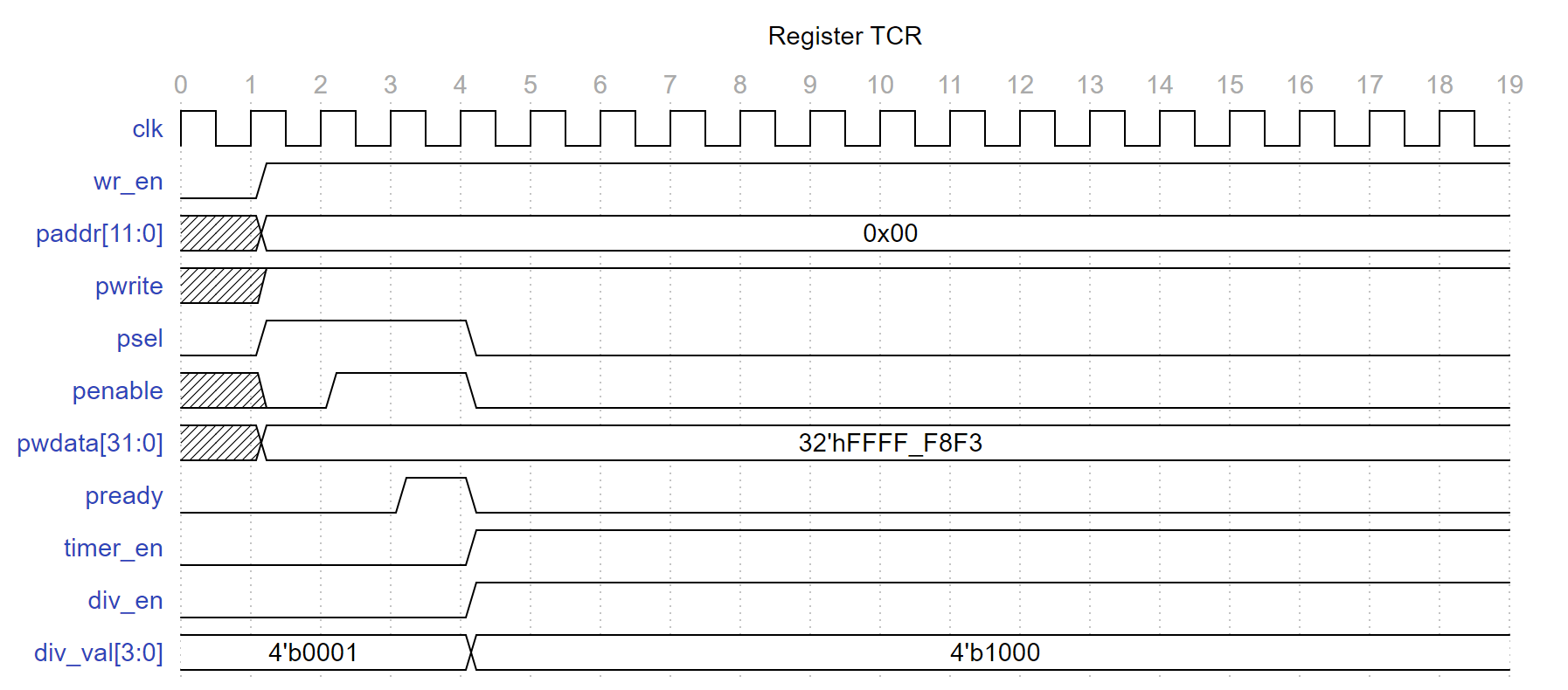
addr[11:0]

div\_val[3:0]



Logic diagram 4: Register TCR

* When wr\_en and addr = 0x00
  + wdata[0] is equivalent to timer\_en
  + wdata[1] is equivalent to div\_en
  + wdata[11:8] is equivalent to div\_val[3:0]



Waveform 3: Register TCR

### 2.3 Byte access

addr[7:0]

pwrite

Block diagram 5: Byte access

Byte access

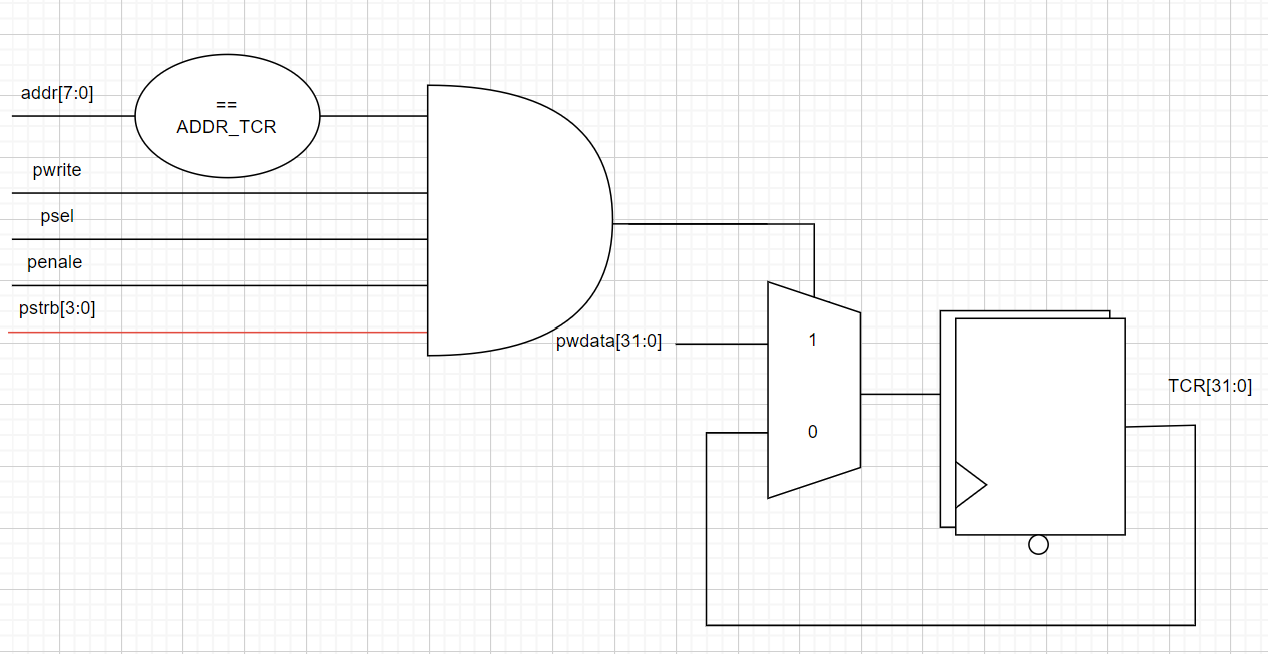
psel

penable

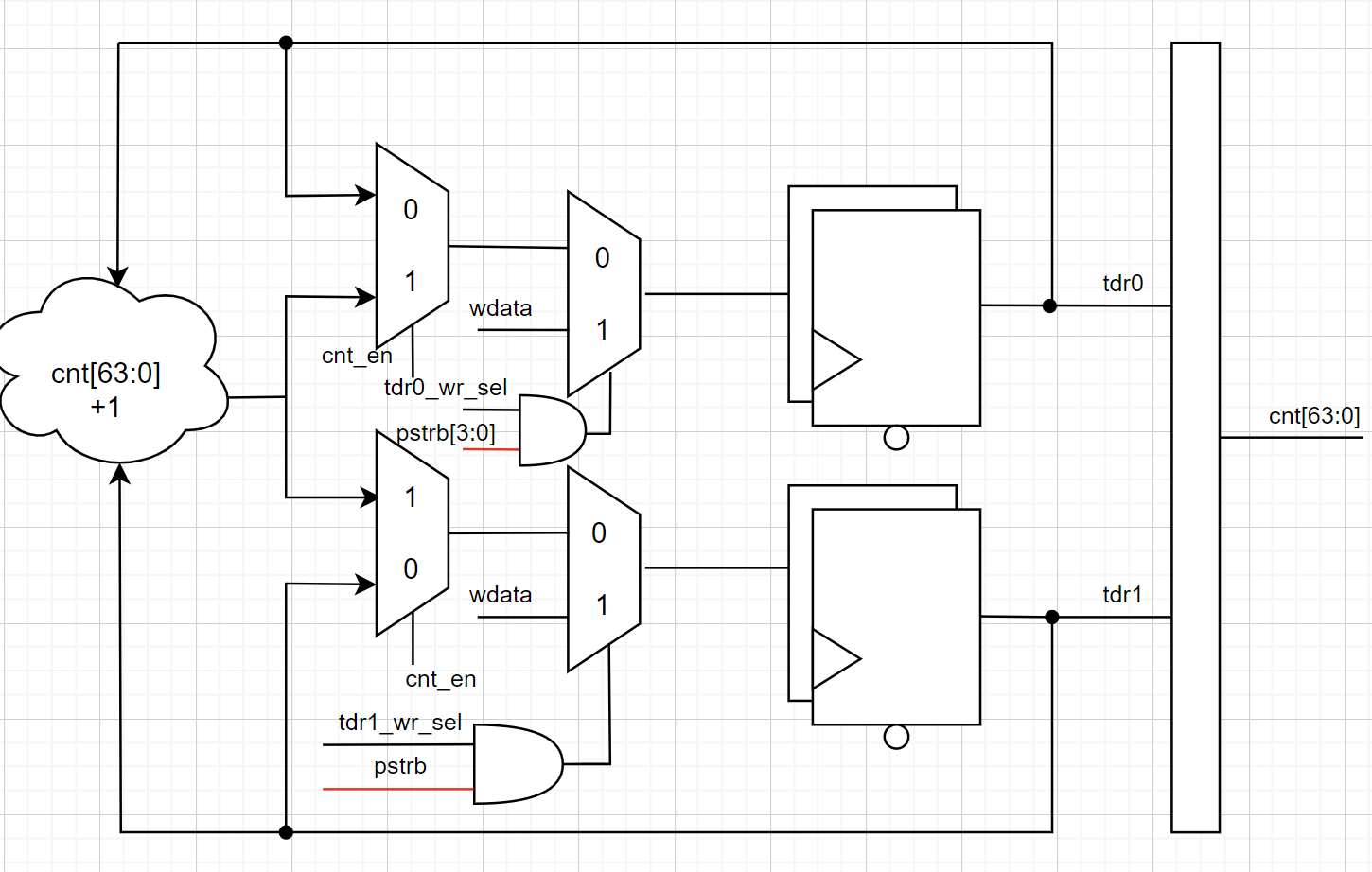
register[31:0]

pstrb[3:0]

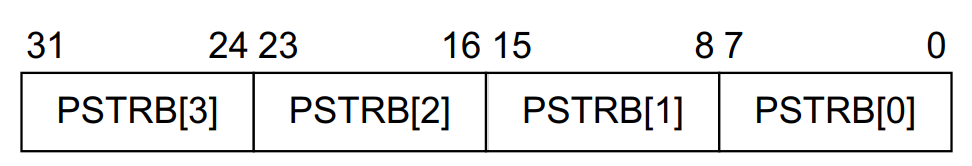
ư

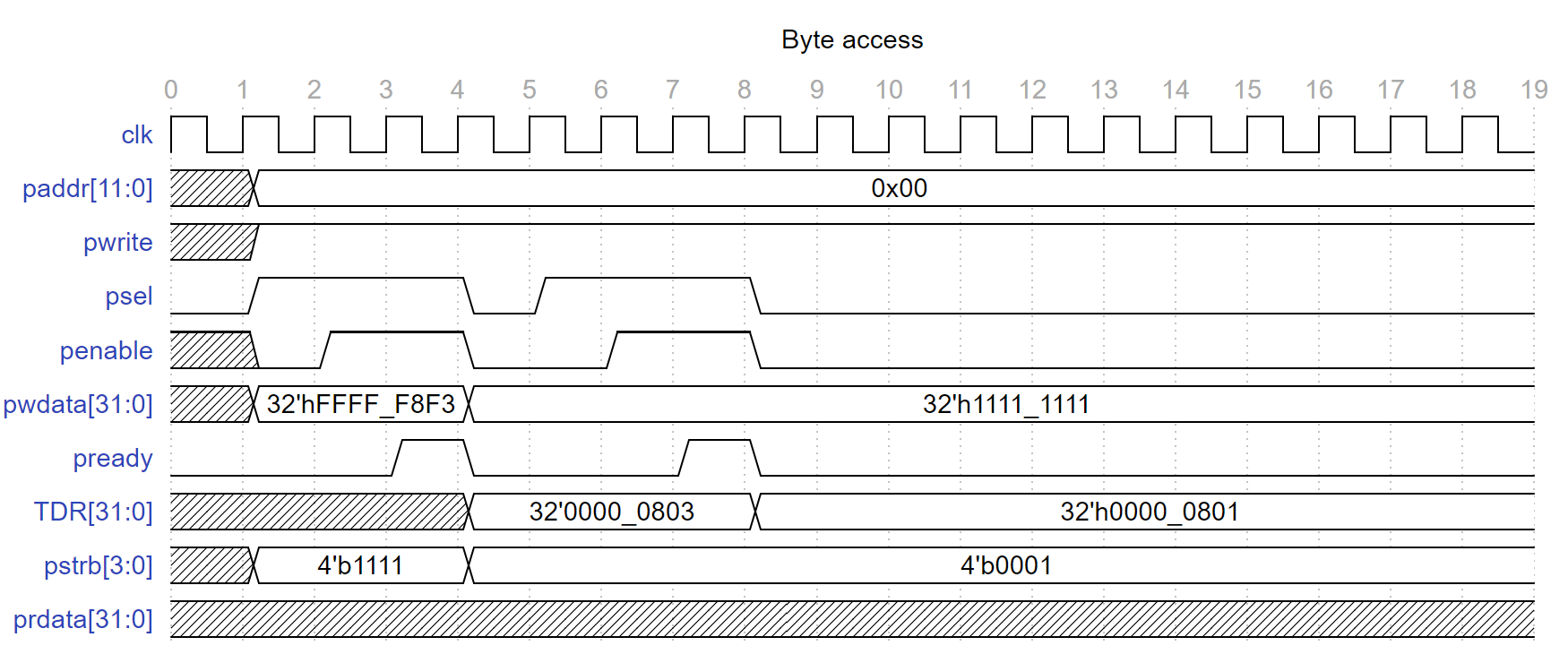


Logic diagram 5: Byte access (1)



Logic diagram 6: Byte access (2)

* Bus can access to individual bytes in the register
* **PSTRB** enables sparse data transfer on the write data bus. Each **PSTRB** corresponds to 1 byte of the write data bus.
* When asserted HIGH, **PSTRB** indicates that the corresponding byte lane of the write data bus contains valid information.
* ****There is one write strobe for each 8 bits of the write data bus, so **PSTRB[n]** corresponds to **PWDATA[(8n + 7):(8n)]**.



Logic diagram 7: Byte access

### 2.4 Error handling

pwdata[11:8] & pwdata[1]

Block diagram 6: Error handling

Error handling

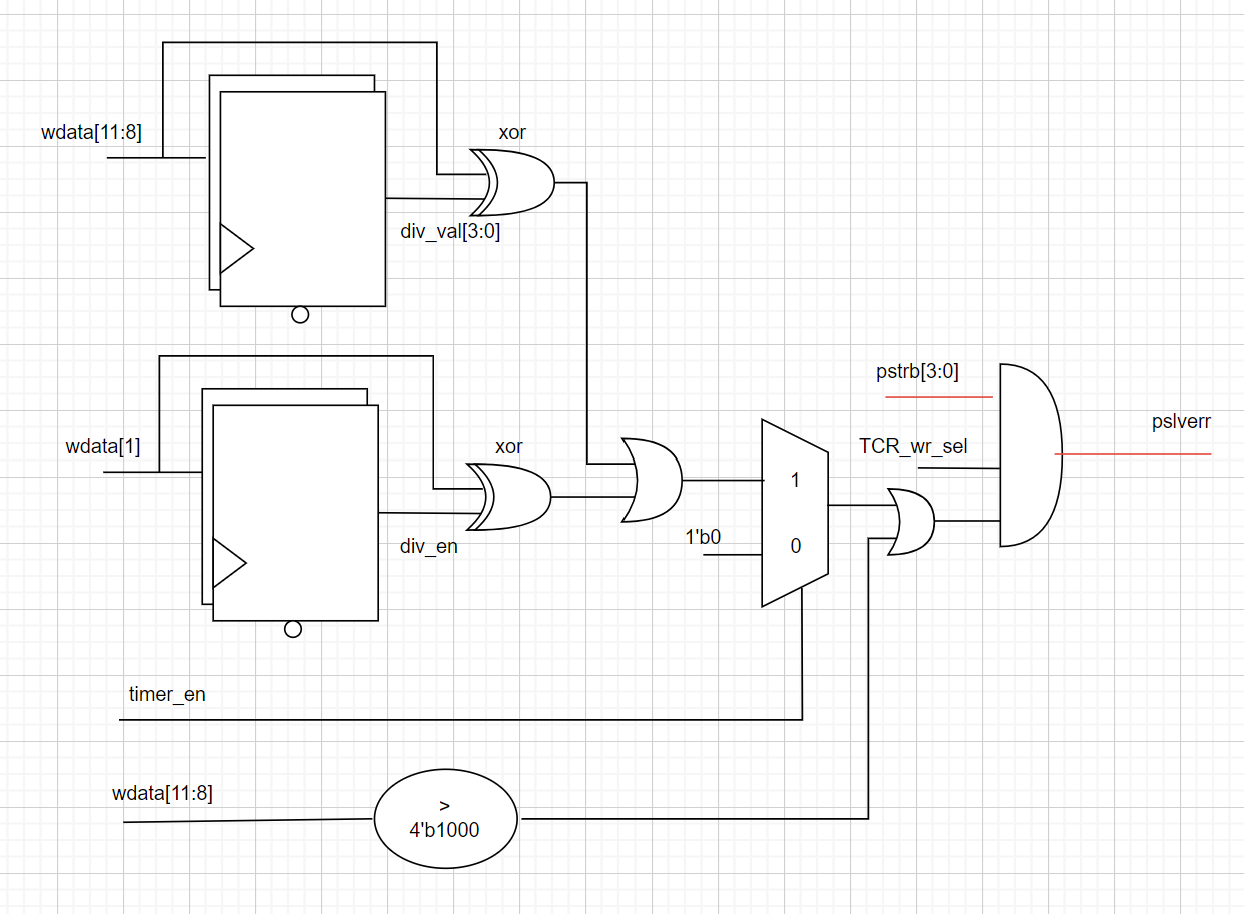
addr[7:0]

pwrite

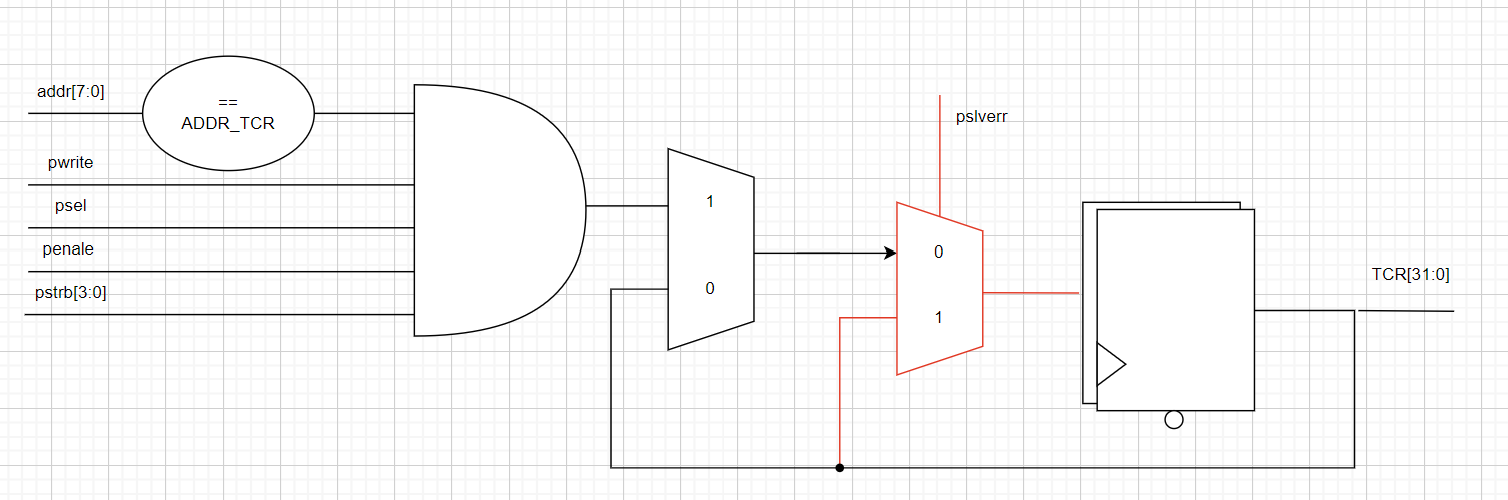
pslverr

timer\_en

TCR\_wr\_sel

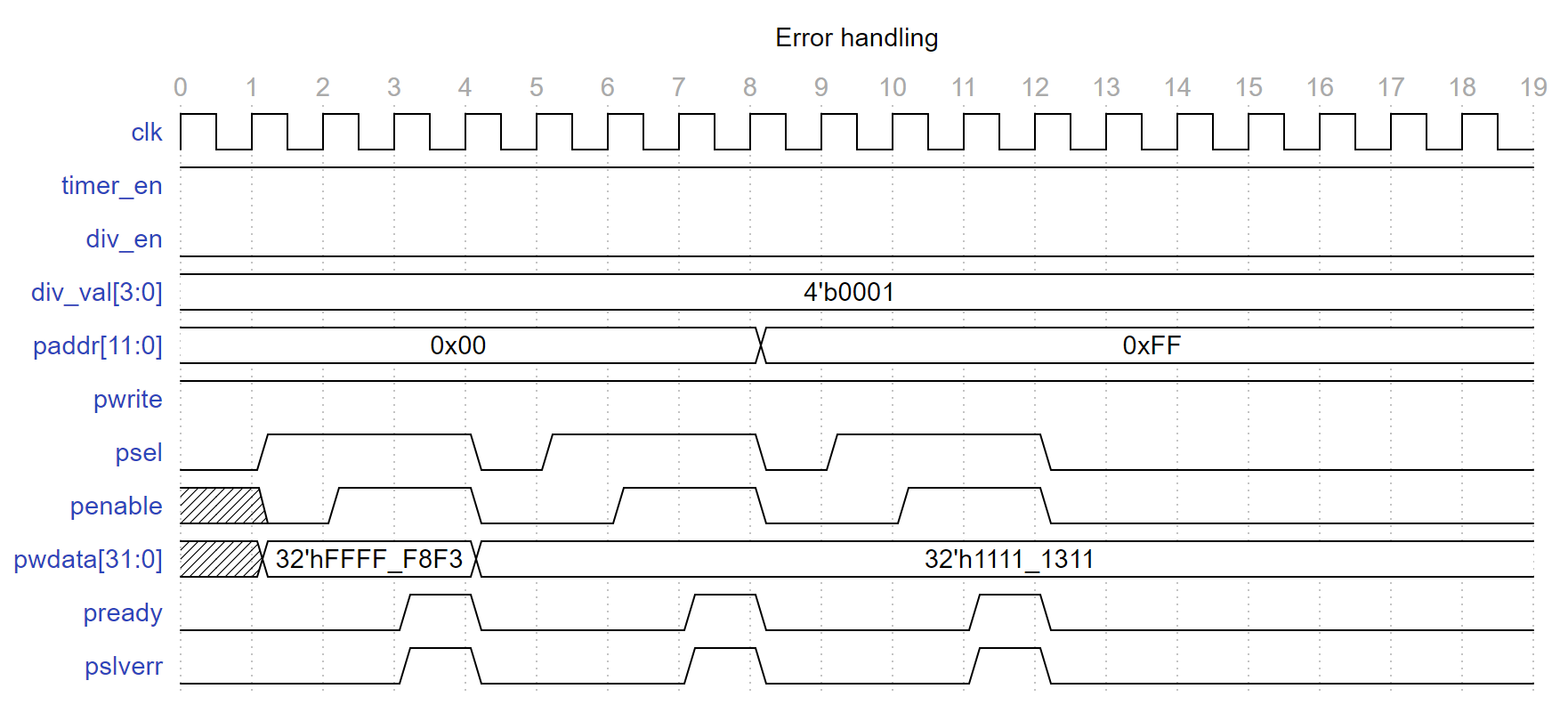


Logic diagram 8: Error handling (1)



Logic diagram 9: Error handling (2)

* Support error handling for some prohibited access:
  + Write prohibited value to TCR.div\_val
  + Div\_en and div\_val changes during timer is operatings
  + Access to invalid address
* When error occurs, data is not writen to register bit/fields



Waveform 4: Error handling

## Counter control

div\_en

Block diagram 7: Counter control

Counter control

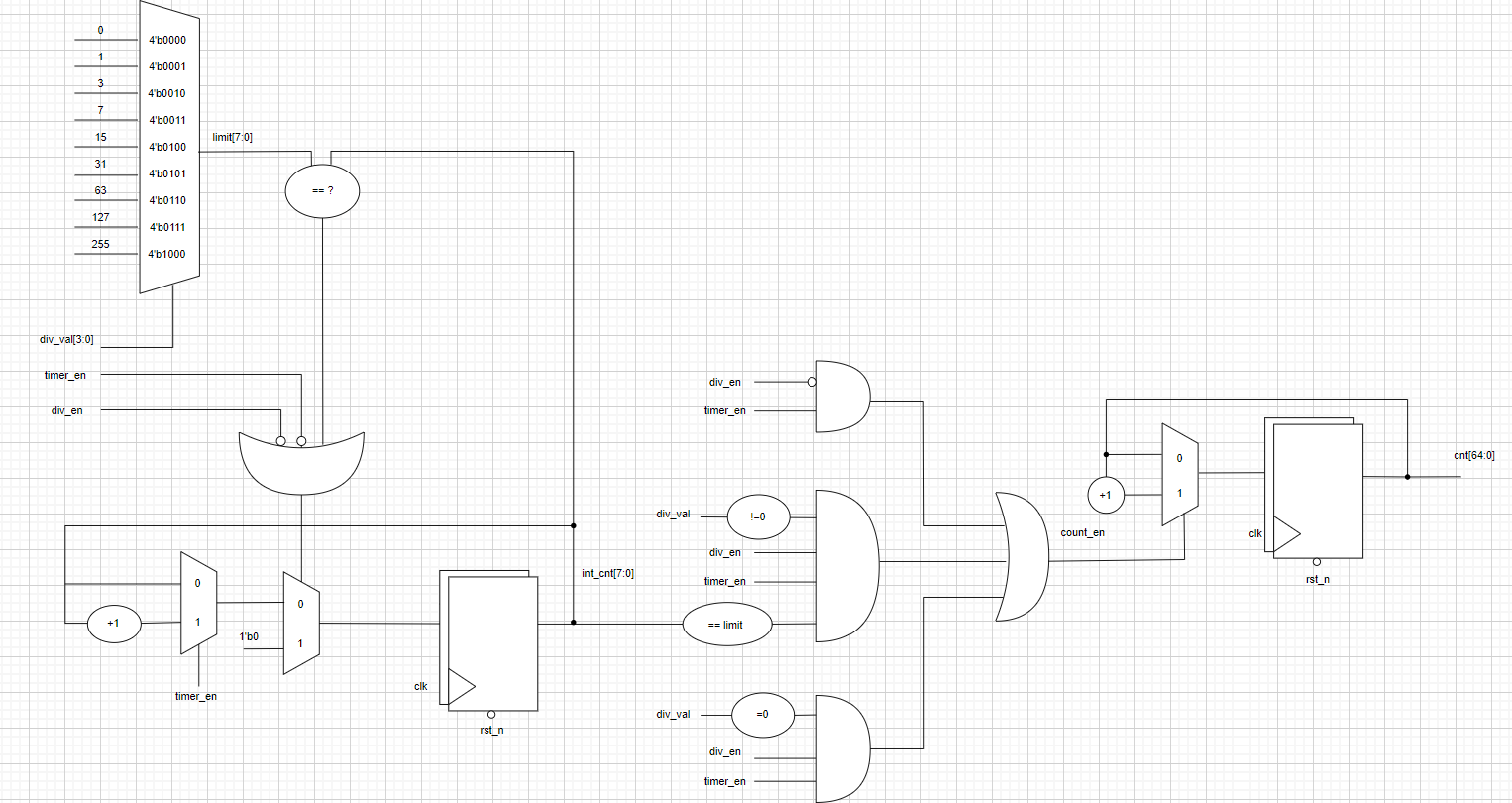
timer\_en

div\_val[3:0]

cnt\_en

halt\_req

debug\_mode

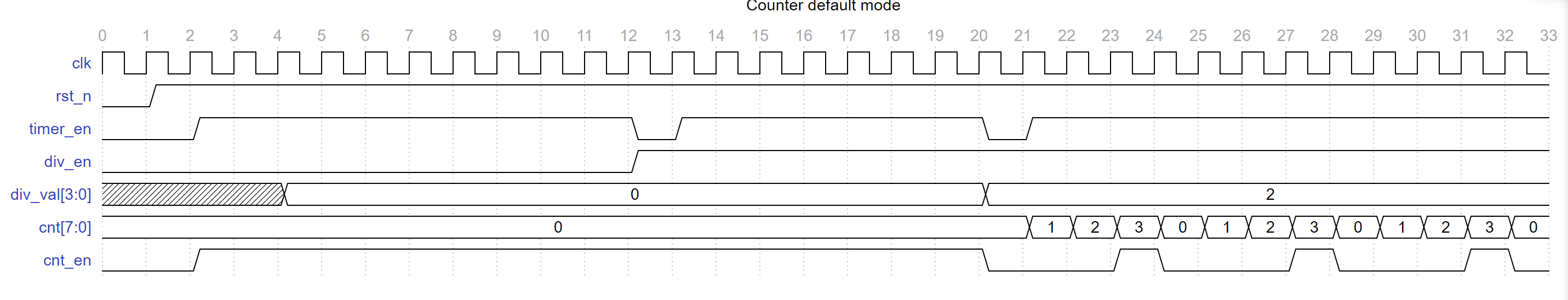


Counter block

Counter control block

Waveform 5: Counter control

* Counter enable (cnt\_en) is the signal that enables the counter to count up on the rising edge of the clock
* Counter enable (cnt\_en) is asserted when
  + Div\_en disbale and time enable is asserted
  + Div\_en and time\_en are asserted and div\_en is zero
  + Div\_en and time\_en are asserted, div\_en is not zero and cnt[7:0] matchs limit value



Waveform 6: Counter enable

1. Counter

cnt\_en

Counter

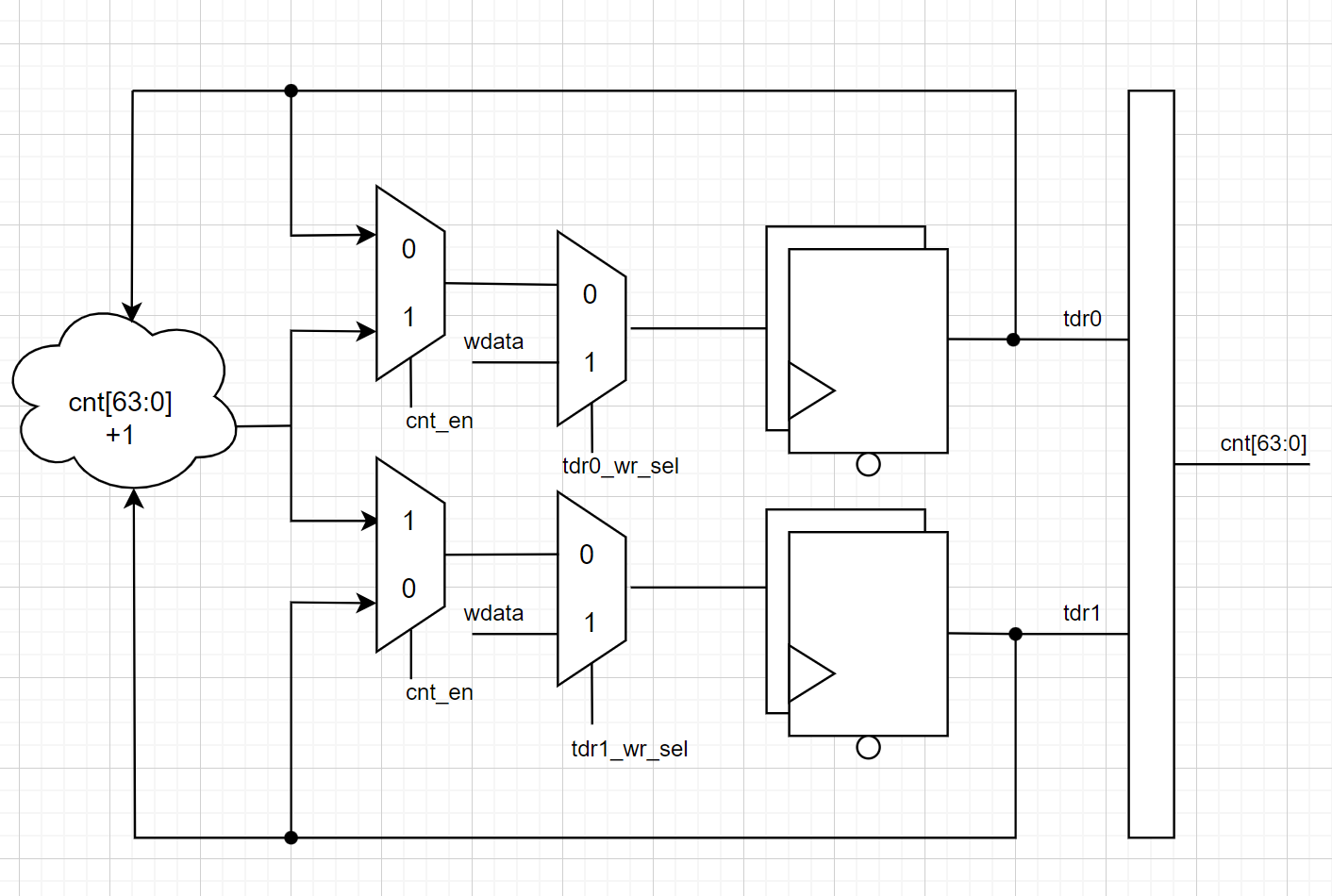
wdata[31:0]

cnt[63:0]

tcr0\_wr\_sel

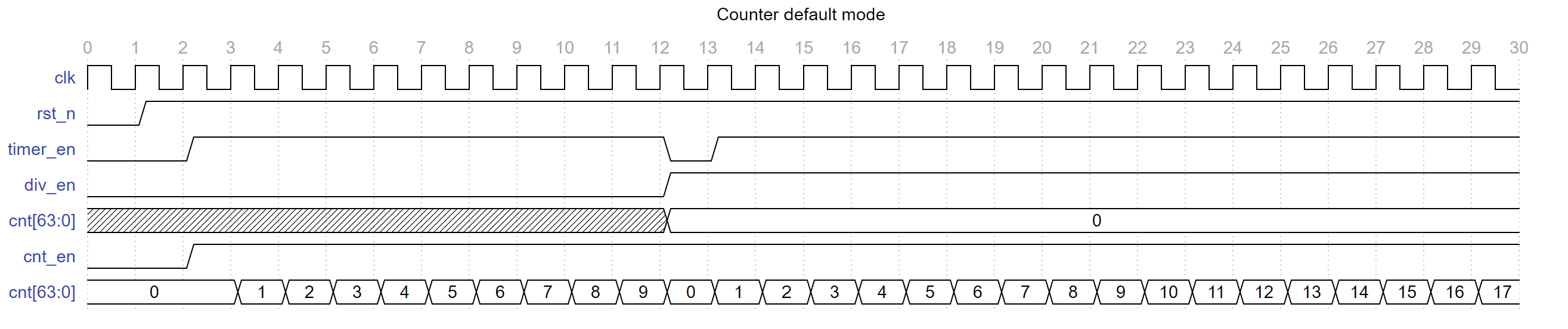
tcr1\_wr\_sel

Block diagram 8: Counter



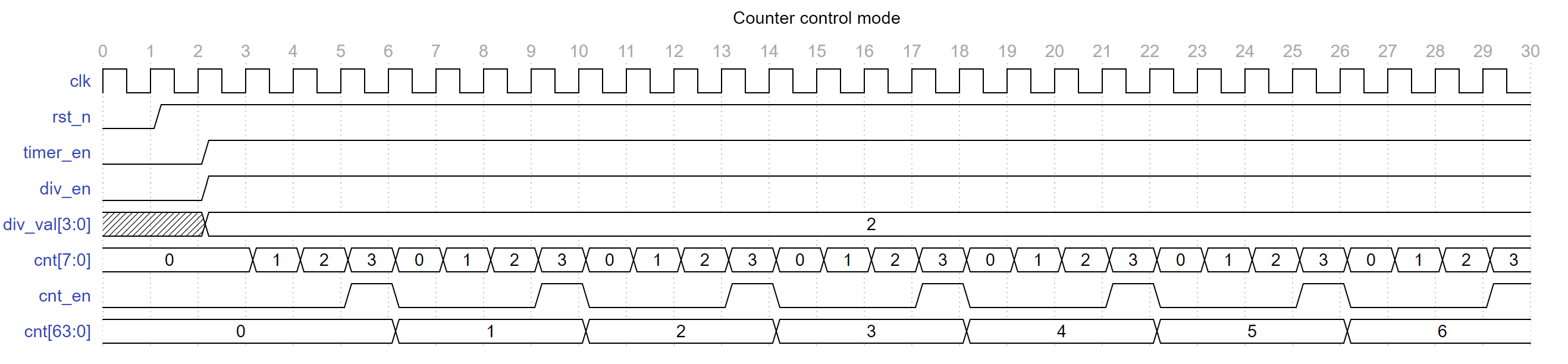
Logic diagram 10: Counter

* When there ís a cnt\_en signal, the counter will count normally
* Counter mode
  + Default mode: the counter’s speed is same as cycle clock



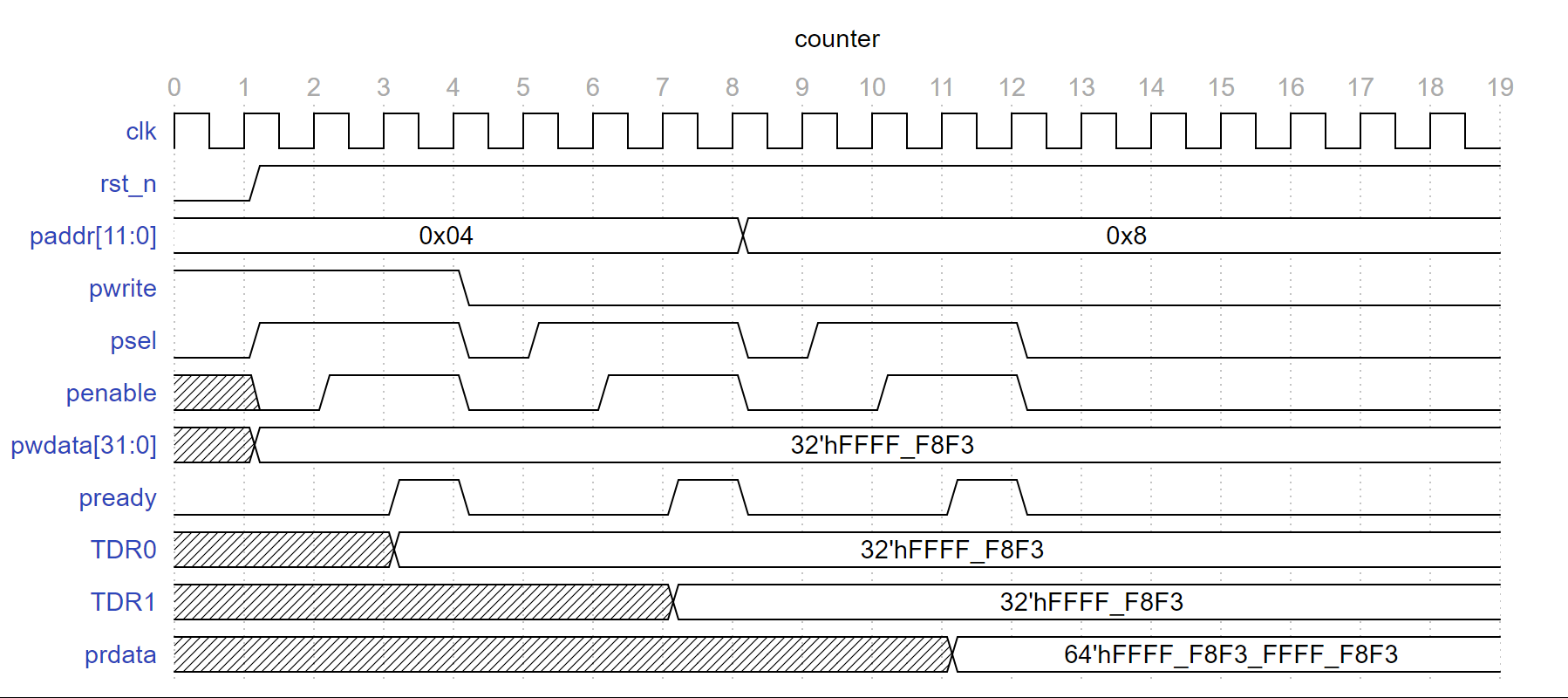
Waveform 7: Counter default mode

* + Control mode: when enabled by writing 1 to TCR.div\_en bit, the counter speed is determined by the divisor value of TCT.div\_val[3:0]



Waveform 8: Counter control mode

* We can read counter’s value by using read transfer of apb protocol
* The counter’s value can be updated immediately by written the value into the register timer data register 0 (TDR1) and timer data register 1 (TDR1)



Waveform 9: Counter

1. Timer interrupt

cnt[63:0]

Block diagram 9: Timer interrupt

Timer interrupt

tim\_int

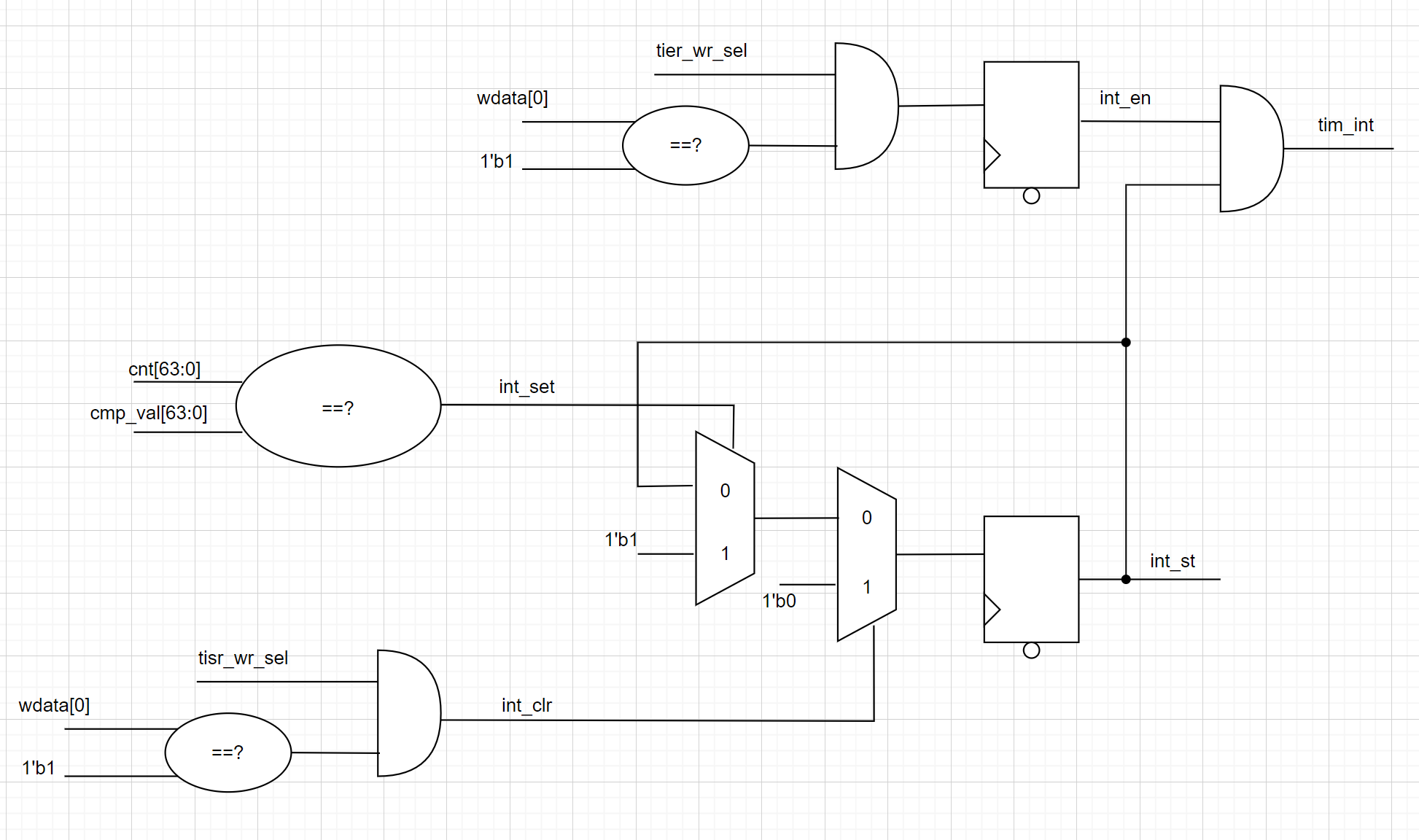
cmp\_val[63:0]

int\_st

wdata[31:0]

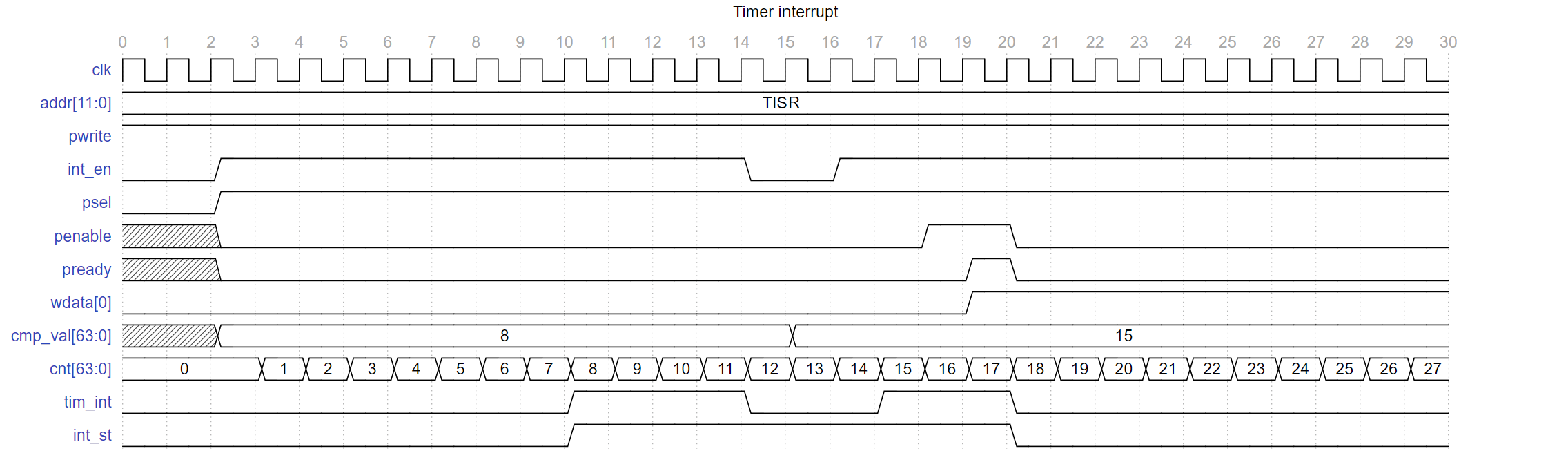
int\_en

tisr\_wr\_sel



Logic diagram 11: Timer interrupt

* Timer interrupt (tim\_int) ís asserted when interrupt enable (int\_en) and counter’s value matches the compare value
* Once asserted, the timer interrupt (tim\_int) remains unchange until it is cleared by writing 1 to TISR.int\_st bit or the interrupts is disabled



Waveform 10: Timer interrupt

1. Halted mode

paused

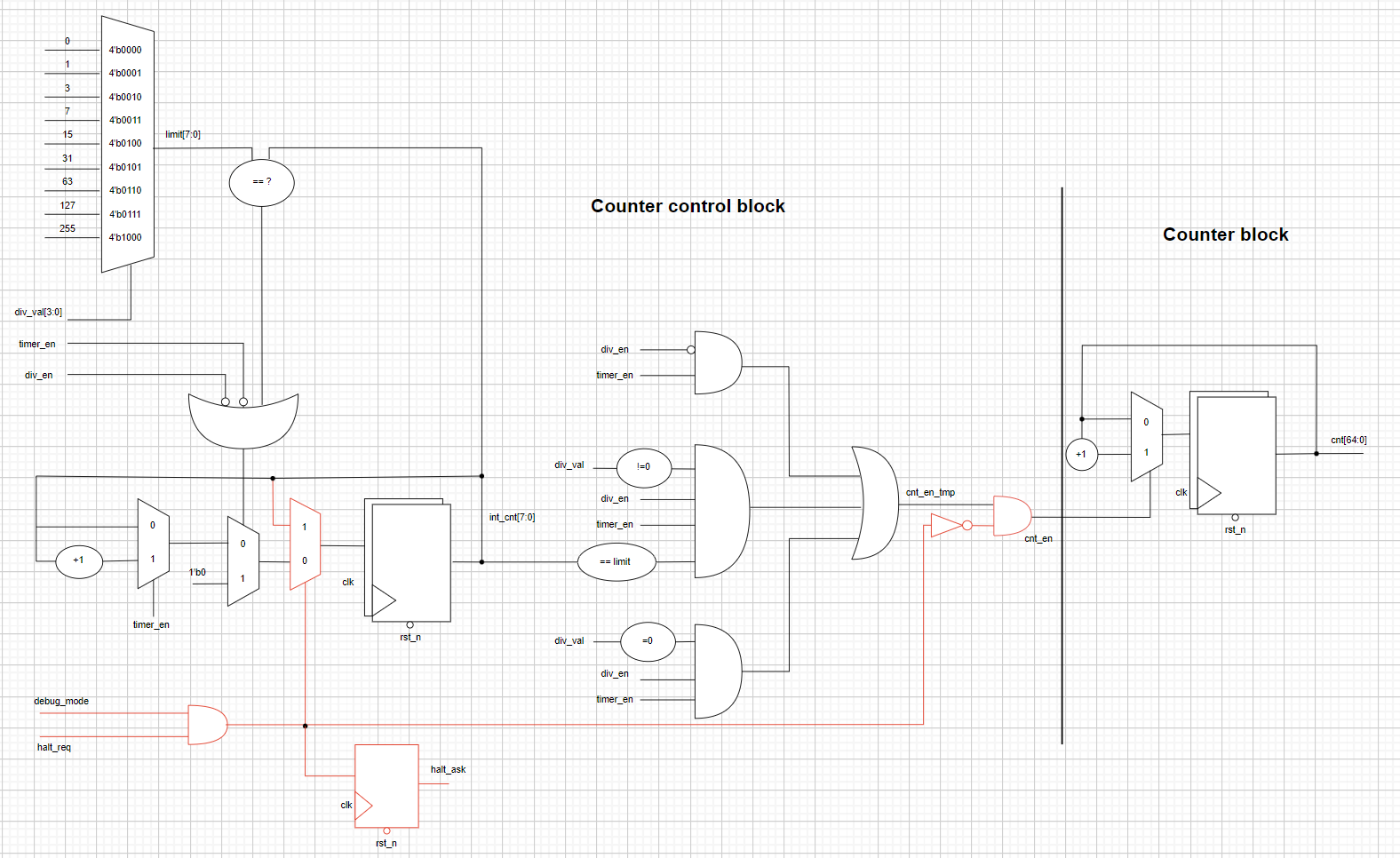
debug\_mode

Block diagram 10: Halt mode

Halt mode

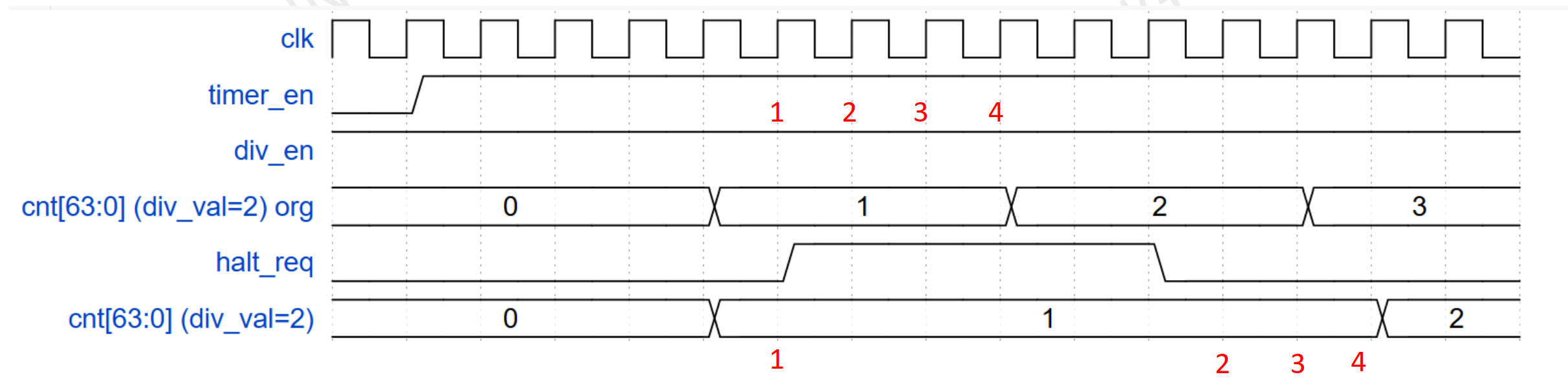
halt\_ask

halt\_req



Logic diagram 12: Halt mode

* Counter can be halted in debug mode when both below conditions occur:
  + Input debug\_mode signal is High.
  + THCR.halt\_req is 1
* THCR.halt ask 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 wavefor example (div\_val=2)

****

Logic diagram 13: Halt mode

1. Register Desciptioin
2. Register summary

Base address: 0x4000\_1000

|  |  |  |
| --- | --- | --- |
| **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 |  |

1. Register specification

### 2.1 Timer Control Register -TCR

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bit** | **Name** | **Type** | **Default value** | **Description** |
| 31:12 | Reserved | - | 20’h0 | Reserved |
| 11:8 | div\_val | 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, (\*)prohibit settings. When setting the prohibit value, div\_val is not changed. Note: user must not change div\_en 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 | timer\_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 |

### 2.2 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. |

### 2.3 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. |

### 2.4 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 |

### 2.5 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 |

### 2.6 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 |

### 2.7 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 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 |

### 2.8 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 | 1’b0 | Timer halt request 0: no halt req. 1: timer is requested to halt. |