**Final Project**

**Requirement**

The timer has following features:

▪ 64-bit count-up.

▪ Address space: 4KB (0x4000\_1000 – 0x4000\_1FFF)

▪ Register set is configured via APB bus (IP is APB slave).

o Only support APB 32-bit transfer with no wait states and no error handling

o Support wait state (1 cycle is enough) and error handling

o Support byte access

o Support halt (stop) in debug mode

▪ System clock frequency is 200 MHz. 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).

**FEATURE LIST**

**Counter**

* Performs counting 64bit based on the system clock frequency by default, with an option to adjust the counting speed using the division factor (div\_val).
* Supports enabling/disabling speed control through div\_en.
* Can clear the count value to its initial state or resume counting based on control signals.
* Provides a halting feature, allowing the counter to stop temporarily while retaining the current count and division settings, and resumes counting when instructed.

**Interrupt**

* Triggers an interrupt when the counter reaches a specified comparison value.
* Allows clearing the interrupt by writing to a specific interrupt status register (TISR).
* The interrupt enable (int\_en) must be set for the interrupt to be active.

**APB bus & APB slave**

* Facilitates 32-bit read/write transfers between the counter module and the APB bus.
* Monitors for errors during transfer operations, such as when invalid values are written to configuration registers or when changes are made to critical parameters (div\_en, div\_val) while counting is active.
* Provides error signaling to the APB master when invalid operations occur, ensuring proper error handling.

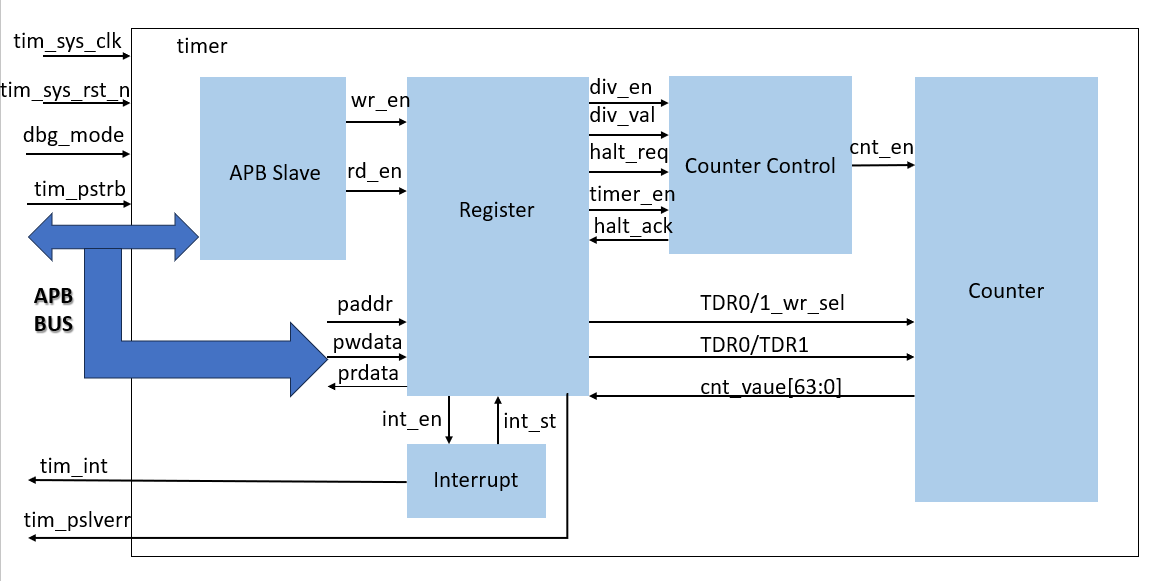
**Register**

* Handles the read/write access to the counter’s internal registers, including control, status, and configuration registers.
* Uses system clock (PCLK) to synchronize all operations and resets registers to default values using the reset signal.
* Provides control signals (Write\_en, Read\_en) to determine whether the operation is a read or write and selects the target register via an address bus.
* Manages data flow for both reading from and writing to the registers, ensuring proper data transfer during operations.

**IO PORT LIST**

|  |  |  |  |
| --- | --- | --- | --- |
| **Signal Name** | **Width** | **Direction** | **Description** |
| sys\_clk | 1 | Input | System clock signal for the timer. The clock frequency is 200 MHz, and it's used to drive all internal logic in the timer module. |
| sys\_rst\_n | 1 | Input | Active-low asynchronous reset signal. When asserted (0), it resets the entire timer module to its initial state, including the counter. |
| tim\_psel | 1 | Input | APB peripheral select signal. It indicates whether the timer module is selected for an APB transaction. This signal must be high for any read or write operation. |
| tim\_pwrite | 1 | Input | APB write enable signal. When high (1), it indicates a write transaction, and when low (0), it indicates a read transaction. |
| tim\_penable | 1 | Input | APB enable signal. It indicates the second phase of an APB transfer, allowing the timer to respond to the APB transaction. This signal is asserted after the tim\_psel and remains high for one clock cycle. |
| tim\_paddr | 31 | Input | APB address bus. The width 31-bit is determined by the address space required to access all registers in the timer. This address is used to select the specific register being accessed. |
| tim\_pwdata | 32 | Input | APB write data bus. It carries the 32-bit data that will be written to the addressed register in the timer when tim\_pwrite is asserted. |
| tim\_prdata | 32 | Output | APB read data bus. It provides the 32-bit data from the addressed register in the timer when a read operation is performed (tim\_pwrite is low). |
| tim\_pstrb | 4 | Input | APB write strobe signal. It specifies which byte lanes are valid for the write operation. Each bit corresponds to one byte . |
| tim\_pready | 1 | Output | APB ready signal. It indicates the timer is ready to complete the current transaction. This signal can introduce wait states if the timer is not ready to respond. |
| tim\_pslverr | 1 | Output | APB slave error signal. It indicates if there is an error during the transaction. It can be asserted if the transaction is invalid. |
| tim\_int | 1 | Output | Timer interrupt signal. It is asserted when a timer interrupt occurs, such as when the counter reaches a certain value or a compare register matches the counter value. |
| dbg\_mode | 1 | Input | Debug mode signal. When asserted, the timer will halt in debug mode. |

**BLOCK DIAGRAM**

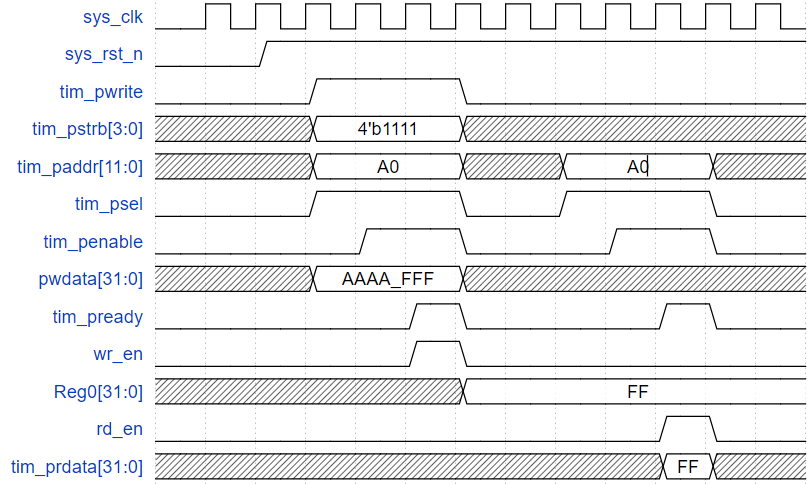
****

**DESCRIPTION**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Module Counter Control & Counter** | | | | | |
| **Input** | **Output** | **Mode** | **Condition** | **Description** | **Note** |
| Wdata  [31:0] | cnt[63:0] |  | System clock | Counter uses system clock frequency (default), controlled with div\_val. |  |
| TDR0/1\_wr |  | Default |  | Counter uses system clock frequency (default), controlled with div\_val. | div\_en and div\_val affect counter behavior. |
| TCR.div\_en |  | Control | TCR.div\_en = 1 | Counter speed control is enabled using div\_val. | div\_en and div\_val must not change when timer\_en = 1. |
| TCR.div\_val[3:0] |  | TCR.div\_val | Sets the division factor for counting speed. | Error response when changing div\_en or div\_val when timer\_en = 1. |
|  |  |  | H->L | Clear count value to initial value. |  |
|  |  |  | L->H | Continue normal counting. |  |
| THCSR |  | Halted |  | Stop counting but maintains previous div\_val settings until halt\_reg = 0. |  |
| hal\_reg |  | Halted | debug\_mode = 1 && THCSR.halt\_reg = 1 | Stop counting but remember the current count value, resumes on halt\_reg = 0. |  |
| hal\_ack |  | THCSR.halt\_ack = 1 | Indicates that the timer is halted. |  |
| **MODULE INTERRUPT** | | | | | |
| **Input** | **Output** | **Mode** | **Condition** | **Discription** | **Note** |
| TIER.int\_en | tim\_int |  | tim\_int = 1 | Interrupt asserted when counter equals compare value. |  |
| TISR.int\_st |  |  | tim\_int = 1 (clear interrupt) | Write 1 to TISR.int\_st to clear the interrupt. |  |
| **Module APB SLAVE** | | | | | |
| **Input** | **Outut** | **Mode** | **Transfers** | **Discription** | **Note** |
| PCLK | PRDATA[31:0] |  | 32-bit transfer read/write |  |  |
| RST\_N | PREADY |  |  |  | READ/WRITE RESERVED RAZ/WI |
| PSEL | wr\_en |  | Write transfers |  |  |
| PENABLE | rd\_en |  | Read transfers | Read transfers |  |
| PWRITE |  |  |  |  |  |
| PADDR  [30:0] |  | Error | Error handling | Invalid values written into TCR.div\_val triggers an error. |  |
| PWDATA  [31:0] |  |  | Write data | Change in div\_en, div\_val when timer\_en = 1 causes error. |  |
| PSLVERR |  |  |  | It indicates if there is an error during the transaction. It can be asserted if the transaction is invalid. | When error occurs, data is not written into register bit/fields |
| PSTRB[3:0] |  |  |  | APB write strobe signal. It specifies which byte lanes are valid for the write operation. |  |

**WAVEFORM APB**

**With wait states:**



**Write transfer**:

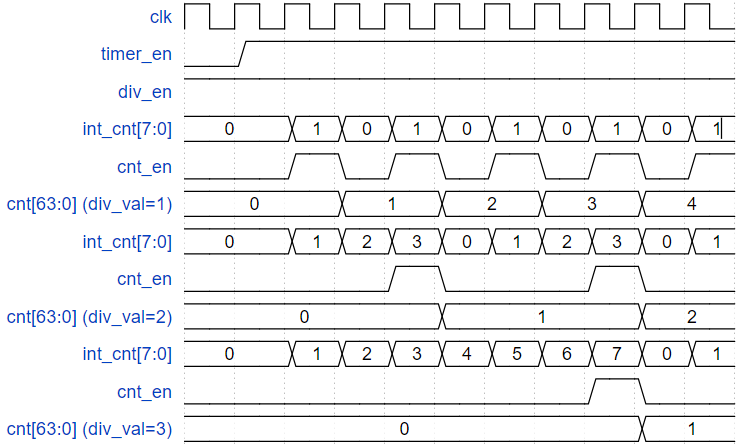
* **pwrite** -> 1, indicating a write operation.
* **psel** & **penable** are asserted sequentially, signaling the start of a transaction.
* **paddr** holds the address A0 and **pwdata** holds the data FF to be written.
* **pready** -> 1 after one cycle, acknowledging the write. The write enable signal (wr\_en) is also asserted.
* The value FF is written to **Reg0**.

**Read transfer**:

* **pwrite** is low (0), indicating a read operation.
* **psel** and **penable** are asserted again.
* **pready** -> 1 after one cycle, acknowledging the read.
* **rd\_en** is asserted, and **prdata** outputs FF as the read data.

**WAVEFORM COUNTER**

**Control mode:**



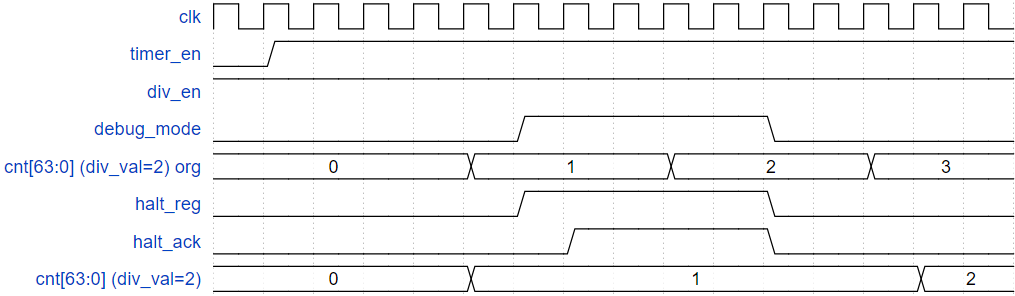
**int\_cnt[7:0]:** alternates between values, synchronized with the counter's progress.

**cnt\_en:** toggles on and off, controlling when the counter increments.

The counter **(cnt[63:0])** behavior:

* For **div\_val=1**, the counter increments every 2 clock cycles (int\_cnt[7:0] values 0, 1).
* For **div\_val=2**, the counter increments every 4 clock cycles (int\_cnt[7:0] values 0, 1, 2, 3), showing slower counting due to the increased division factor.

**Halted mode:**

****

Counter can be halted (stopped) in debug mode when both below conditions occur:

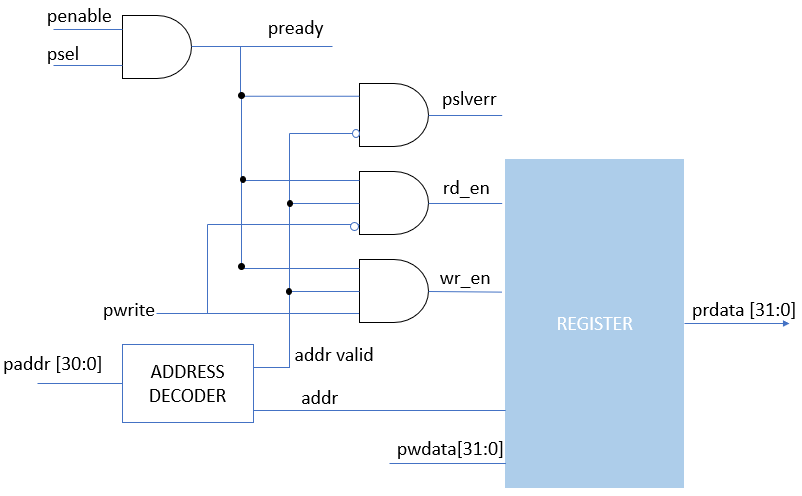
* + Input **debug\_mode** signal is High,
  + **halt\_req** is 1.

**halt\_ack** is **1** after a halt request indicates that the request is accepted.

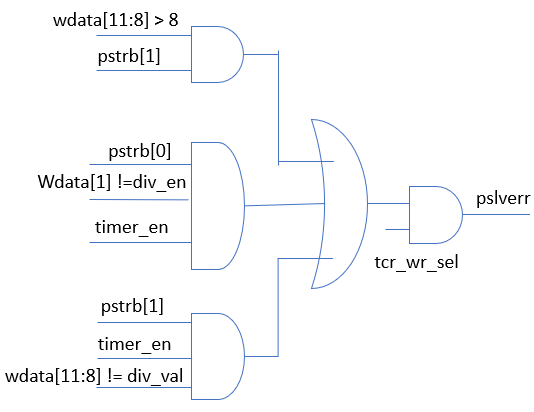
After halted, counter can be resumed to count normally when clearing the halt request to 0.

**LOGIC DIAGRAM**

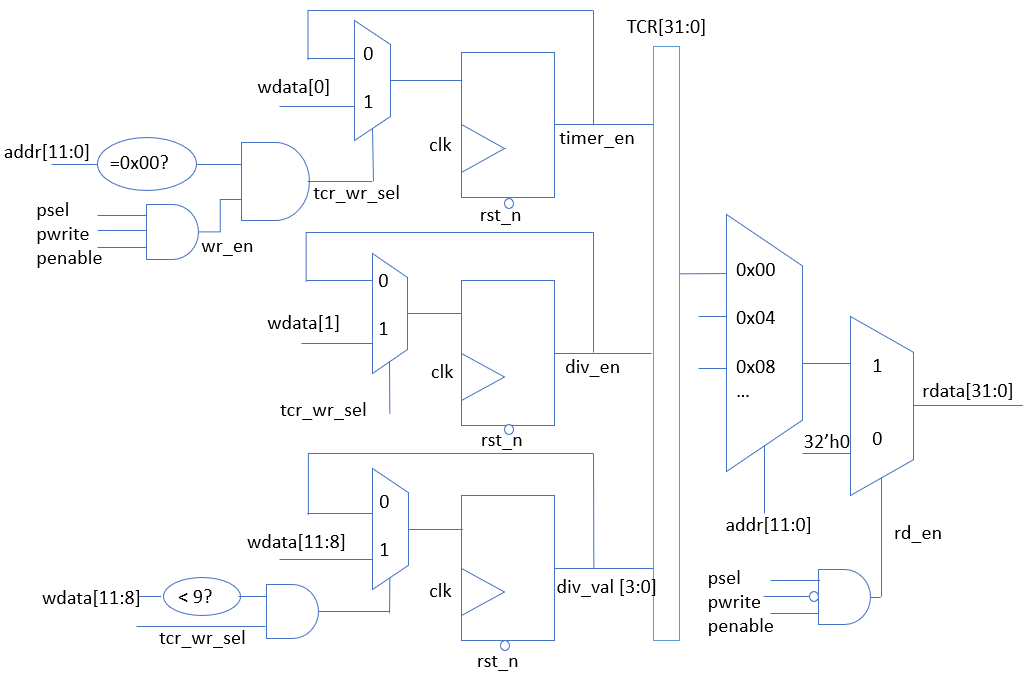
**ABP SLAVE**

****

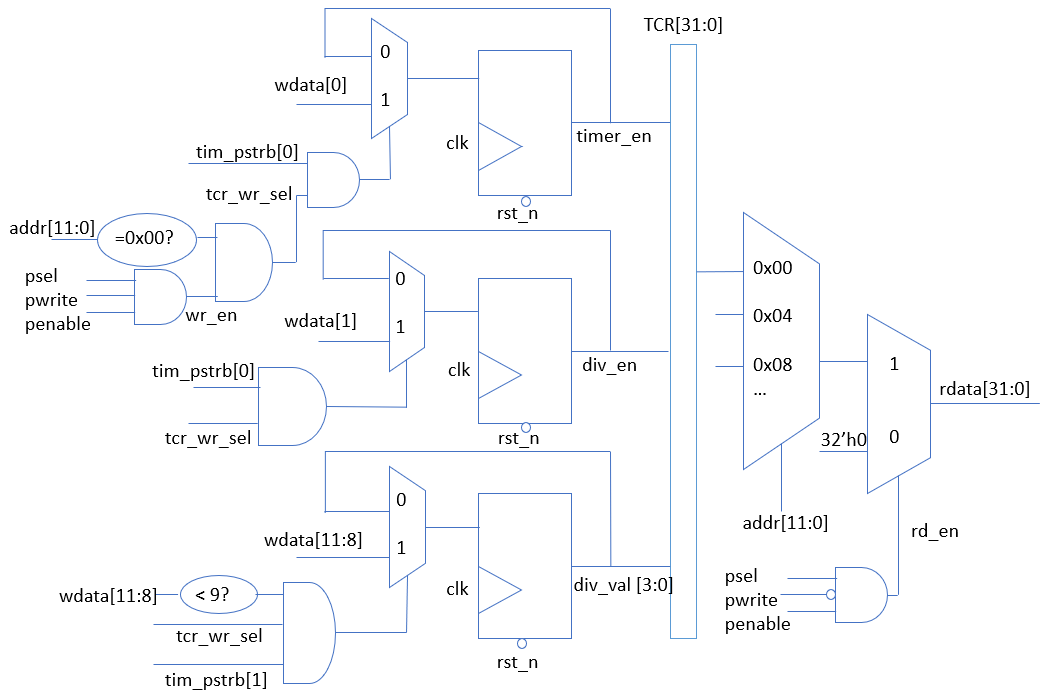
**SLAVE ERROR:**

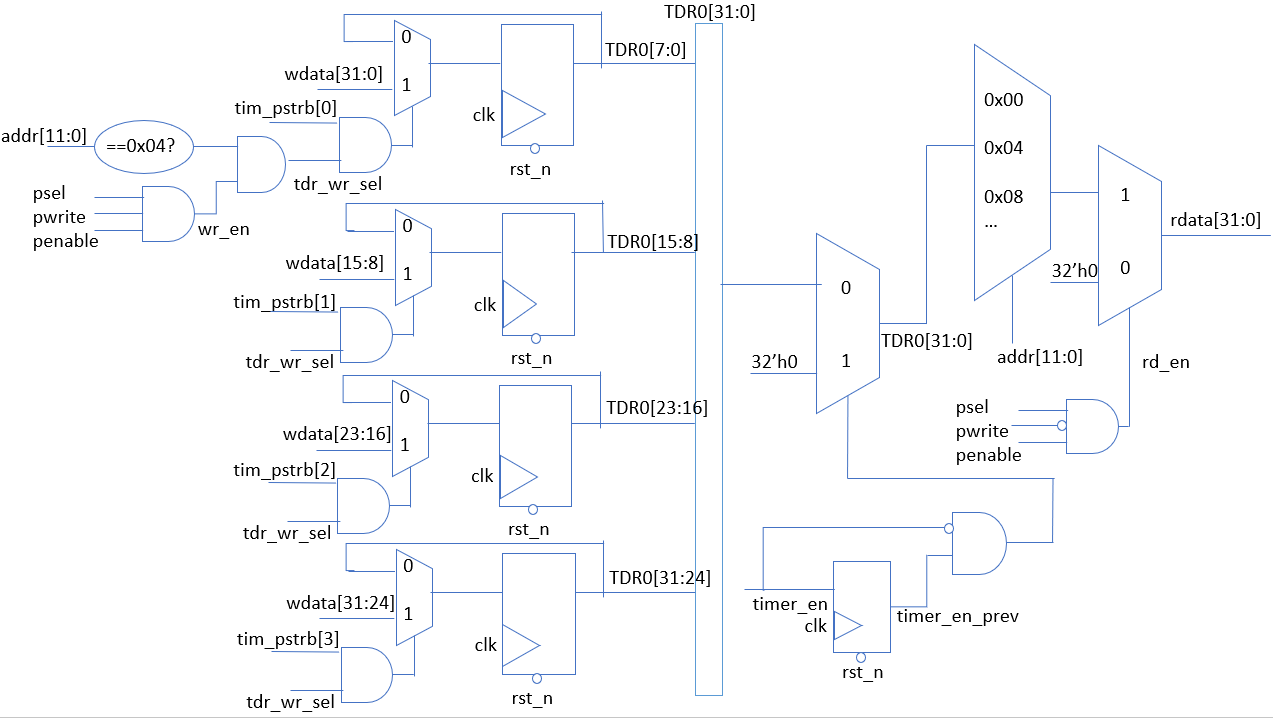
****

**REGISTER FILE**

**TCR (R/W access):**

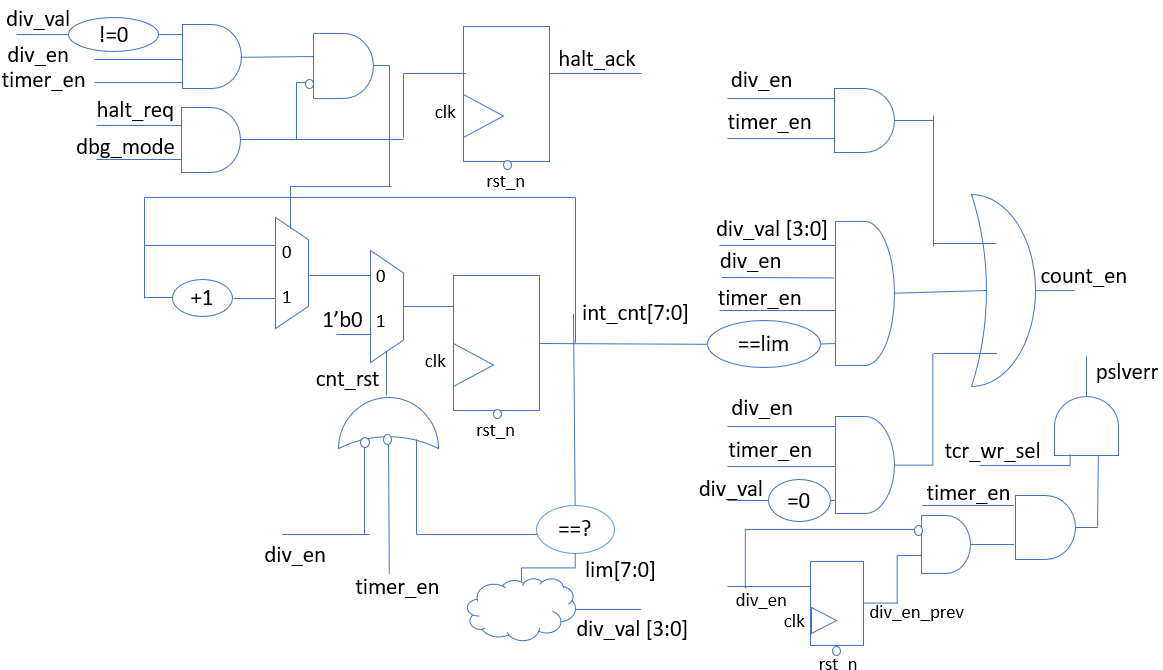
**TCR, TDR0/1 Byte access:**



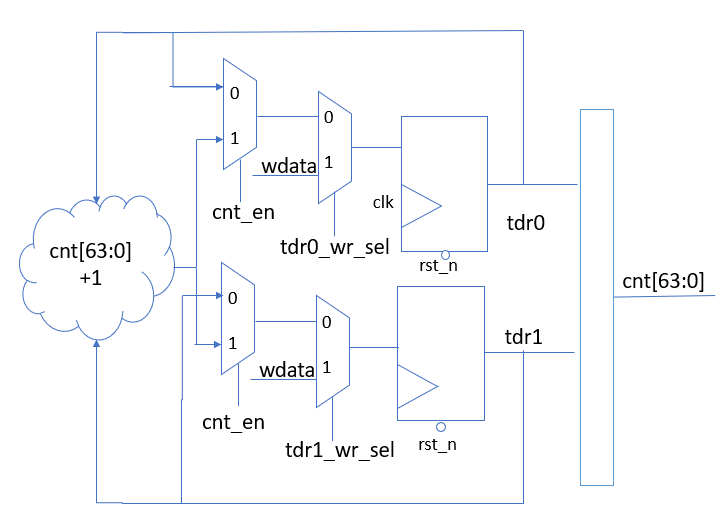


**COUNTER & CONTROL COUNTER**

**Control & halted mode:**

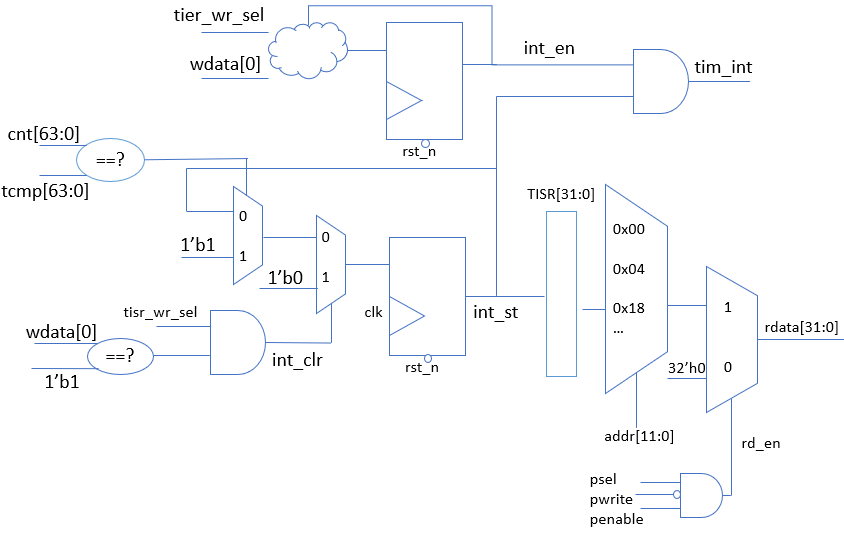
****

**Counter & TDR0/1:**

****

**INTERRUPT**

**INTERRUPT (TIER – TISR):**

****