**STIMER DESIGN**

1. Overview
2. Feature
3. Block diagram
4. Input/Outputs pin
5. Register specification
6. Read/write register control
7. Input/Ouputs pin
8. Functional/Protocol
9. State machine control for APB protocol
10. Timing chart
11. Control logic
12. Input/Outputs pin
13. Functional/Protocol
14. Design circuit
15. Timing chart
16. Timer couter (TCNT)
17. Input/Outputs pin
18. Functional/Protocol
19. Design circuit
20. Timing chart
21. Overflow/Underflow comparison
22. Input/Outputs pin
23. Functional/Protocol
24. Design circuit
25. Timing chart
26. Overview
27. Feature

8-bit counter/timer, Used APB bus for this design and synchronous design

Supports **incrementing** (count up) or **decrementing** (count down) of 8-bit values.

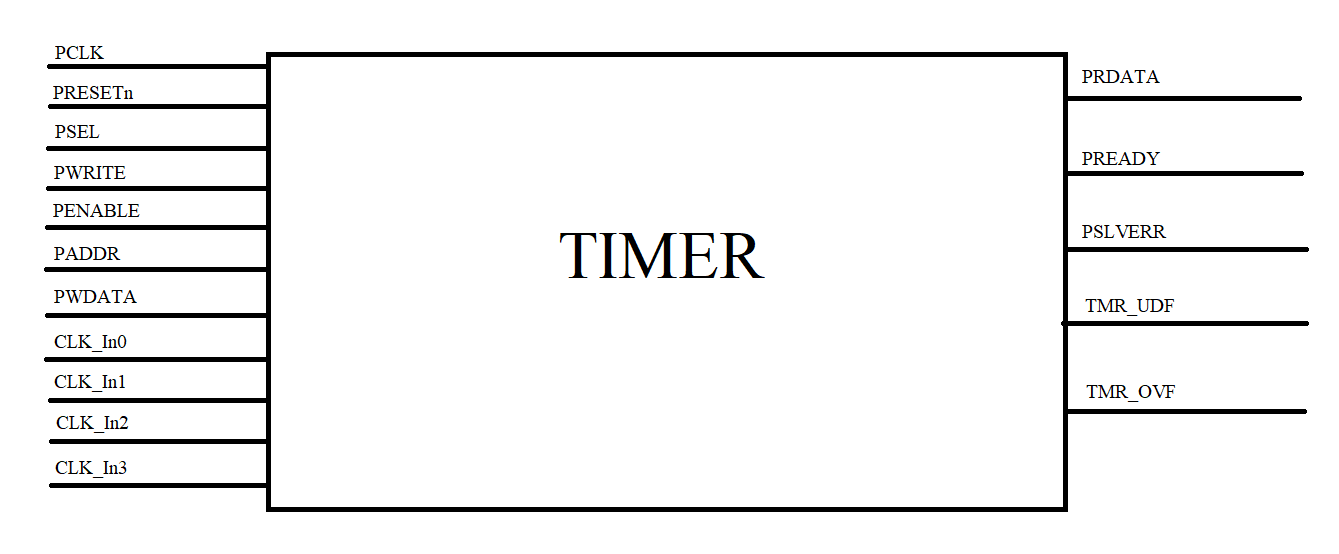
Can operate in counting modes with different internal clock frequencies (PCLKx1,PCLKx2, PCLKx4,PCLKx8).

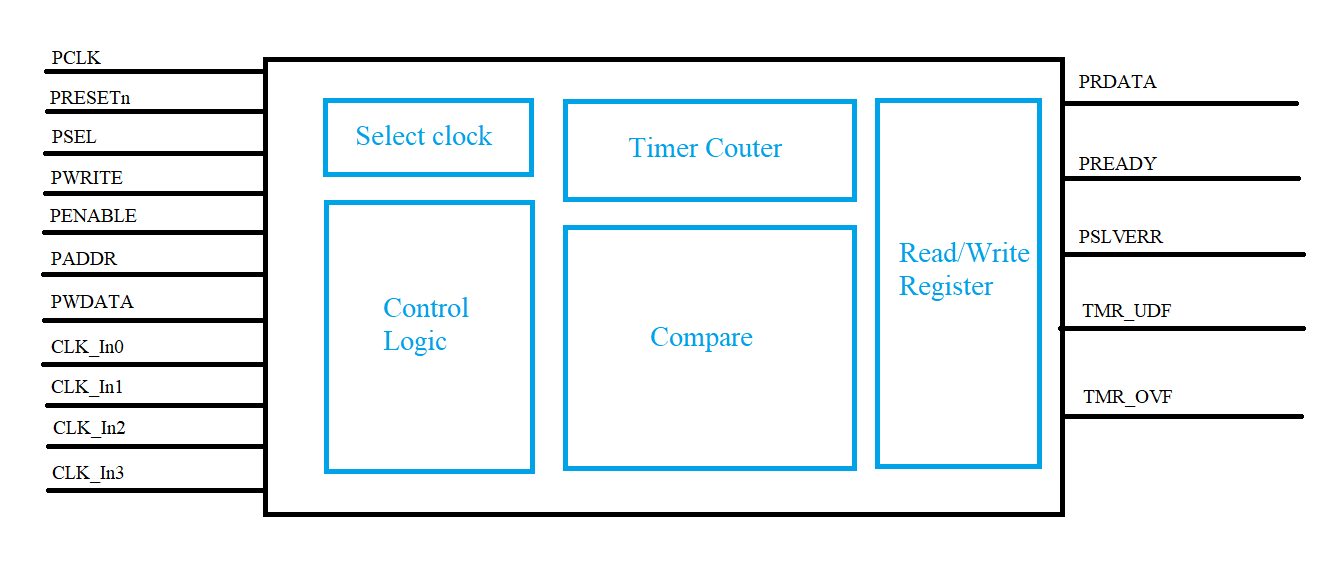
Includes the following registers:

* **TCR (Timer Control Register)**: Controls the operation of the timer.
* **TDR (Timer Data Register)**: Stores the initial preload value.
* **TSR (Timer Status Register)**: Displays the status of the timer.

Detects **overflow** when counting up from 255 and **underflow** when counting down from 0.

1. Block diagram





1. Input/Outputs pin

|  |  |  |  |
| --- | --- | --- | --- |
| PORT name | Attribute | BIT WIDTH | Descripition |
| PLCK | Input | 1 | Provided clock for APB protocol |
| PRESETn | Input | 1 | Active-low synchronous reset for the APB domain. |
| PSEL | Input | 1 | Chip-select generated by the APB bridge |
| PWRITE | Input | 1 | Write direction indicaor  Write cycles when write = 1: Read cycles when read = 0: |
| PENABLE | Input | 1 | Enable-phase qualifier; together with PSEL forms the ENABLE phase during wwhich the actual read or write handshake |
| PADD[7:0] | Input | 8 | Register address within the timer |
| PWDATA | Input | 8 | Write data. This bus is driven by the peripheral bus bridge unit during write cycles when PWRITE is HIGH. |
| PRDATA | Output | 8 | Read data. This bus is driven by the timer during a read cycle; valid in the ENABLE phase of a read transfer |
| PREADY | Output | 1 | Asserted HIGH to indicate the slave is ready to complete the current APB transfer; when LOW during ENABLE phase, the master inserts wait states. |
| PSLVERR | Output | 1 | Slave error indicator; HIGH signals an invalid access or internal fault on the completed transfer |
| TMR URF | Output | 1 | Provided timer counter status when counter count from 00 to FF |
| TMR OVF | Output | 1 | Provided timer counter status when counter count from FF to 00 |

1. Register specification

TDR

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| TDR[7] | TDR[6] | TDR[5] | TDR[4] | TDR[3] | TDR[2] | TDR[1] | TDR[0] |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

|  |  |  |
| --- | --- | --- |
| Bit Name | F/V | Description |
| Tdr[7:0] | R/W | Reload value 8-bit |

TCR

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| Load | Reserved | Up/Dw | En | Reserved | Reserved | Cks1 | Cks0 |

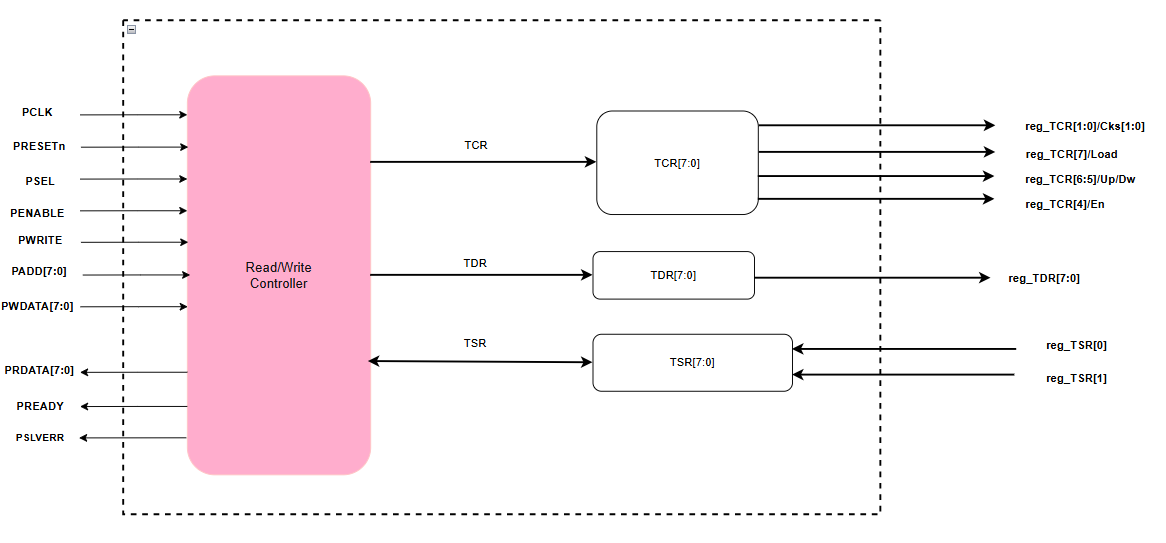
|  |  |  |
| --- | --- | --- |
| Bit Name | F/V | Description |
| Load[7] | R/W | Manual load data from TDR to TCNT when it  active High.  1: load data to TCNT  0: Normal operation |
| 6 | Reserved | Reserved |
| Up/Dw[5] | R/W | Control counter up or counter down 0: counter up 1: counter down |
| En[4] | R/W | 0 : disable 1: enable |
| 3:2 | Reserved | Reserved |
| Cks[1:0] | R/W | Select internal clocks for circuit 00 : T\*2 01 : T\*4 10 : T\*8 11: T\*16 |

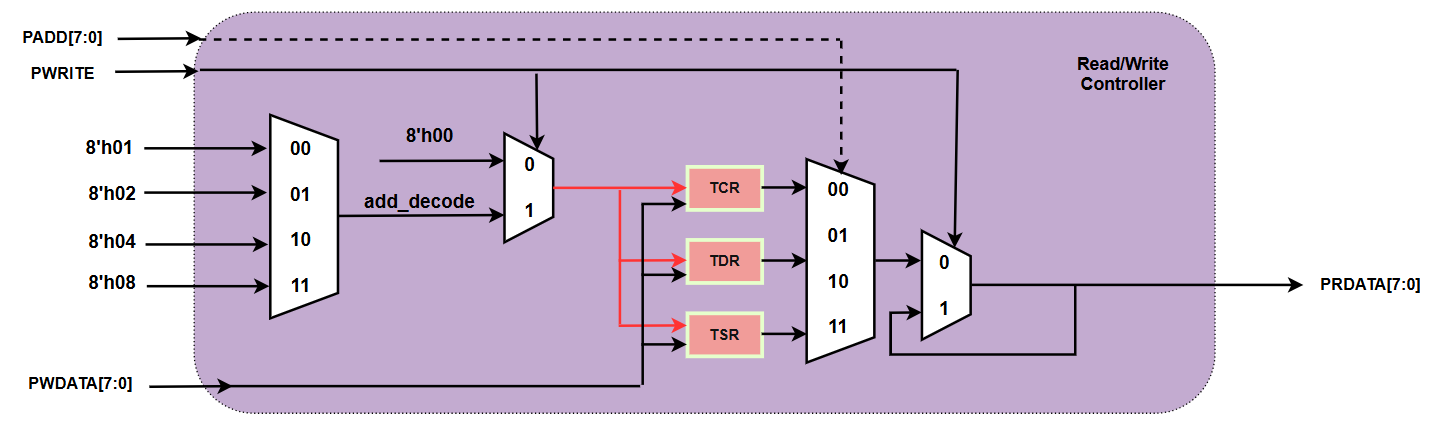
TSR

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | TMR\_UDF | TMR\_OVR |

|  |  |  |
| --- | --- | --- |
| Bit name | R/W | Description |
| 7:2 | R | Reserved |
| TMR\_UDF[1] | R/W\* | Timer counter underflow when counter 8’h00 down to 8’hff: This bit is only set by hardware, clear by software |
| TMR\_OVR[0] | R/W\* | Timer counter overflow when counter 8’hFF to 8’h00: This bit is only set by hardware, clear by software |

1. Read/write register control
2. Input/Ouputs pin



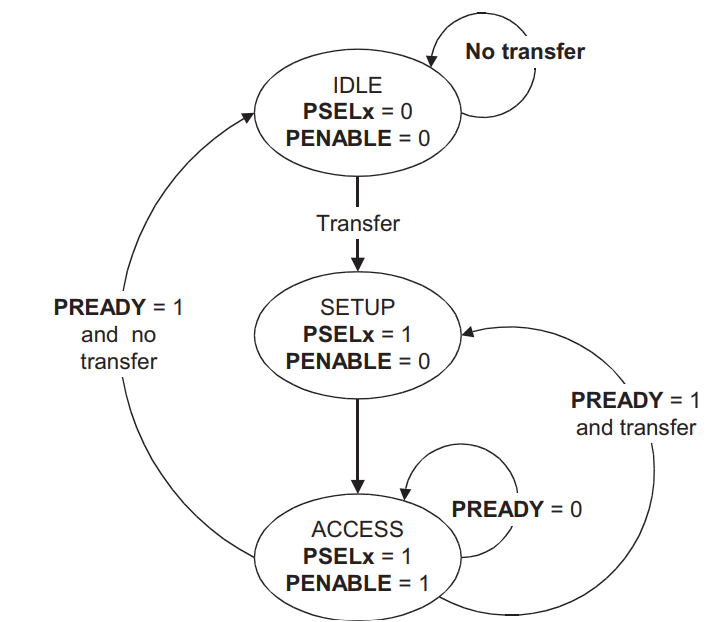


1. Functional/Protocol

The **APB protocol** is used for register read/write operations:

* **Write**: When PSEL = 1, PWRITE = 1, and PENABLE = 1, the data from PWDATA is written to the register at address PADDR.
* **Read**: When PSEL = 1, PWRITE = 0, and PENABLE = 1, the data from the register at PADDR is output to PRDATA.

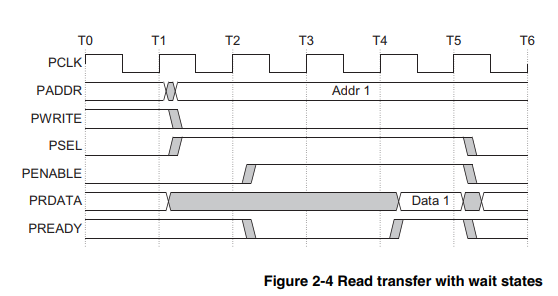
1. State machine control for APB protocol



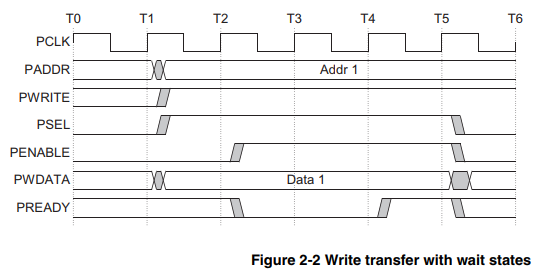
|  |  |
| --- | --- |
| IDLE | This is the default state of the APB |
| SETU | When a transfer is required the bus moves into the SETUP state, where  the appropriate select signal, PSELx, is asserted. The bus only remains  in the SETUP state for one clock cycle and always moves to the ACCESS  state on the next rising edge of the clock |
| ACCESS | The enable signal, PENABLE, is asserted in the ACCESS state. The  address, write, select, and write data signals must remain stable during  the transition from the SETUP to ACCESS state.  Exit from the ACCESS state is controlled by the PREADY signal from  the slave:   * If PREADY is held LOW by the slave then the peripheral bus   remains in the ACCESS state.   * If PREADY is driven HIGH by the slave then the ACCESS state is   exited and the bus returns to the IDLE state if no more transfers arerequired. Alternatively, the bus moves directly to the SETUP state  if another transfer follows. |

1. Timing chart

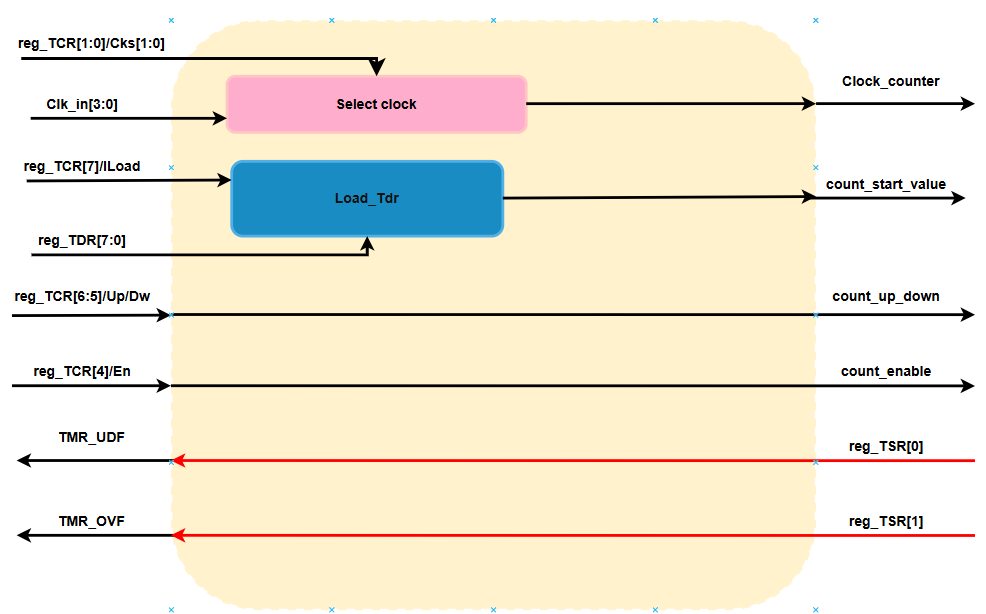
Read transaction wait state



Write transaction wait state



1. Control logic
2. Input/Outputs pin



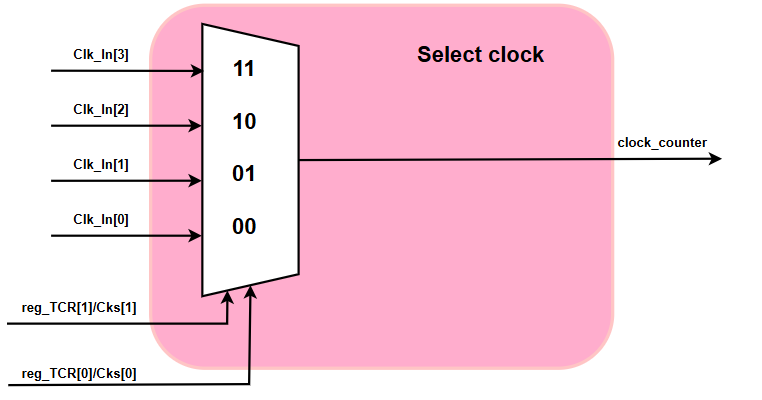
|  |  |  |
| --- | --- | --- |
| PORT name | Attribute | Descripition |
| reg\_TCR[1:0] / Cks[1:0] | Input | Clock select bits from the TCR. These two bits determine which one of the Clk\_in[3:0] signals is selected as the input clock for the timer |
| Clk\_in[3:0] | Input | 4-bit clock input signals.  Select internal clocks for circuit 00 : T\*2 01 : T\*4 10 : T\*8 11: T\*16 |
| reg\_TCR[7] / Load | Input | Control bit to load a value into the counter.  Manual load data from TDR to TCNT when it active High. 1: load data to TCNT 0: Normal operation. |
| reg\_TCR[5] / UpDw | Input | 2 bits to control the counting direction (up or down).  Control counter up or counter down 0: counter up 1: counter down |
| reg\_TCR[4] / En | Input | Enable bit to turn the counter on or off.  0 : disable 1: enable |
| reg\_TSR[0]: | Input | Status bit (possibly overflow), written in red |
| reg\_TSR[1]: | Input | Status bit (possibly underflow), written in red |
| Clock\_counter | Output | The selected clock signal. |
| count\_start\_value | Output | Initial value for the counter. |
| count\_Up\_down | Output | Signal indicating the counting direction  0: Count up  1: Count down |
| count\_enable | Output | Enable signal for the counter  1: disable 0: enable |
| TMR\_UDF | Output | Overflow signal |
| TMR\_OVF | Output | Underflow signal |

1. Functional/Protocol

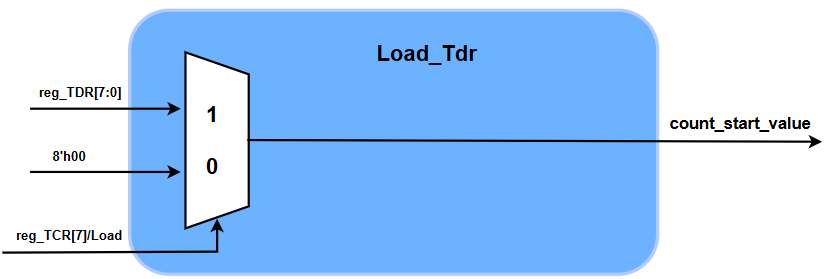
This block using for control logic inside timer IP with control signal list down as below:

* Load bit: Load counter value from tdr register for counter starting value when enable count.
* Up/Dw bit: Control counter up or counter down
* En bit: Enable / freeze the counter
* TSR bit[1]: Timer counter underflow when counter 8’h00 down to 8’hff
* TSR bit[0]: Timer counter overflow when counter 8’hFF to 8’h00

1. Design circuit



When the software writes to the two bits Cks[1:0] in the TCR register, the MUX will switch to get the clock from one of the corresponding sources, then output clock\_counter to control the acceleration rate of the 8-bit counter.

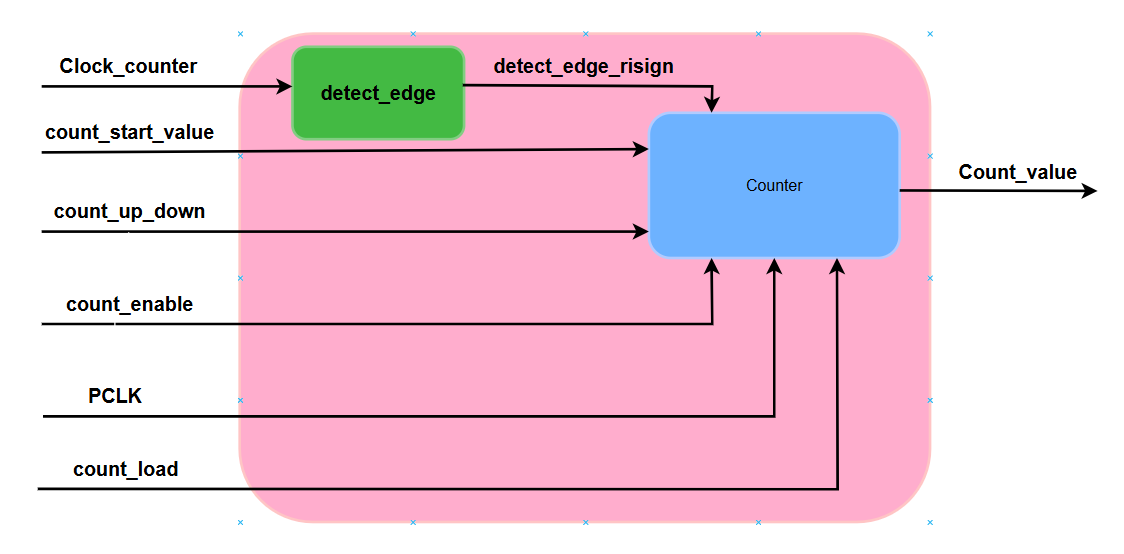


When the software wants to reset (reload) the timer with a new value, it writes Load = 1 and simultaneously puts data into reg\_TDR. The MUX immediately passes reg\_TDR straight down as the initial value.

1. Timing chart

No need due to all circuits using inside control logics are combination logic. These logics will be update value immediately without clock signal.

1. Timer couter(TCNT)
2. Input/Outputs pin



|  |  |  |
| --- | --- | --- |
| PORT name | Attribute | Descripition |
| Clock\_counter | Input | The selected clock signal. |
| count\_start\_value | Input | Initial value for the counter. |
| count\_Up\_down | Input | Signal indicating the counting direction  0: Count up  1: Count down |
| count\_enable | Input | Enable signal for the counter (1: count, 0: freeze). |
| count\_load | Input | Signal to load value from TDR into TCNT  1: load data to TCNT 0: Normal operation |
| detect\_edge\_rising | Output | An output from an edge detection circuit within the TCNT, signaling when a rising edge of the clock is detected |
| TCNT\_out | Output | The current 8-bit value of the counter, ranging from 0 to 255. This is the main output reflecting the counter’s state. |

1. Functional/Protocol

The Timer Counter (TCNT) is an 8-bit counter designed to increment or decrement a value based on a clock signal and control inputs. Its operation follows this protocol:

**Initialization**: The counter can be initialized by loading the count\_start\_value into its register when the count\_load signal is set to 1. This sets the starting point for counting.

**Clock Operation**: The counter uses the Clock\_counter signal (often synchronized with PCLK) as its timing reference. The clock frequency can be scaled (e.g., divided by 2, 4, 8, or 16) depending on the system configuration, though this detail may vary by implementation.

**Counting Process**:

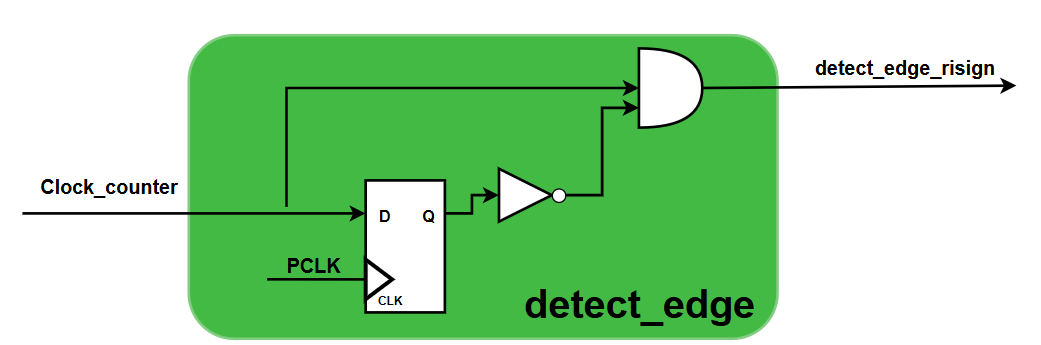
* When count\_enable is 1, the counter updates its value on each rising edge of the clock (as detected by detect\_edge\_rising).
* If count\_up\_down is 0, the counter increments (e.g., from 5 to 6).
* If count\_up\_down is 1, the counter decrements (e.g., from 5 to 4).
* When count\_enable is 0, the counter holds its current value without updating.

**Boundary Conditions**:

* When counting up and reaching 255, the counter wraps around to 0 (overflow).
* When counting down and reaching 0, the counter wraps around to 255 (underflow).
* These events may trigger status flags (e.g., overflow or underflow) in some systems, though not explicitly shown in all diagrams.

The TCNT operates synchronously, ensuring that updates occur only on clock edges when enabled, making it suitable for timing and counting applications in digital systems.

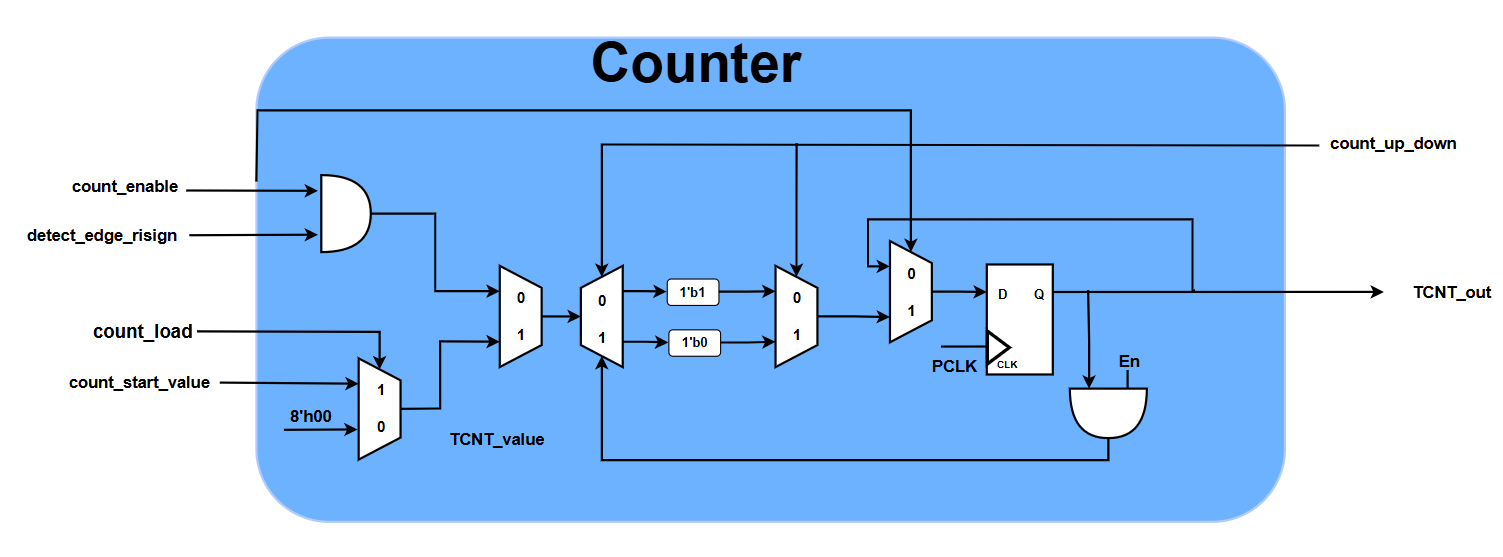
1. Design circuit



 Composed of a D flip-flop and an AND gate.

 The D flip-flop’s D input is connected to Clock\_counter, and its clock input is driven by PCLK. The Q output stores the previous state of Clock\_counter, while the inverted Q̅ output is fed to the AND gate.

 The AND gate takes Clock\_counter and Q̅ as inputs, producing detect\_edge\_rising as output. This signal goes high only when Clock\_counter transitions from 0 to 1 (rising edge).



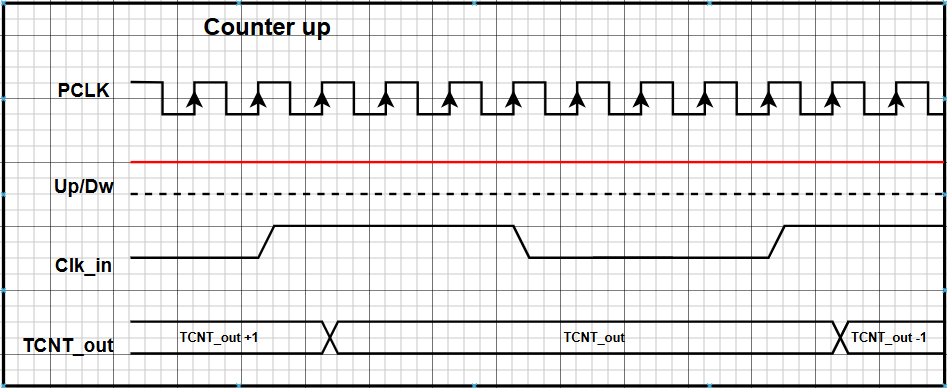
 An 8-bit register (often labeled TCNT\_value) holds the current count.

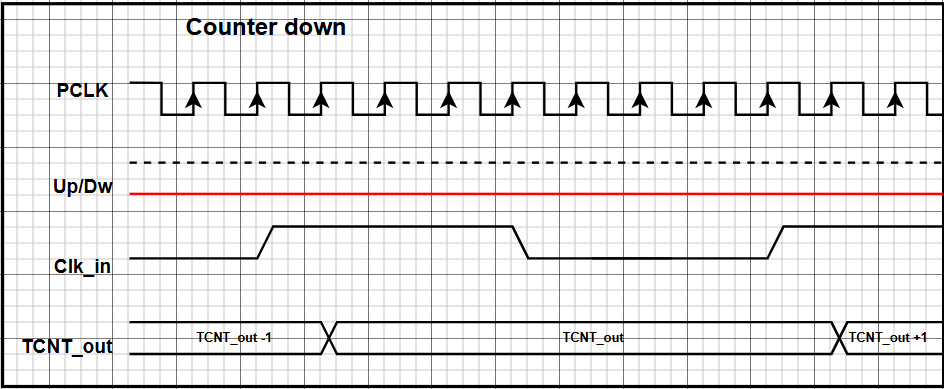
 A multiplexer controlled by count\_load selects between the current count and count\_start\_value. When count\_load is 1, the register loads count\_start\_value.

 Another multiplexer, controlled by an internal signal, selects between incrementing (adding 1) or decrementing (subtracting 1) the count, based on count\_up\_down.

 The counter updates its value on the rising edge of PCLK, gated by count\_enable and triggered by detect\_edge\_rising.

1. Timing chart





1. Overflow/Underflow comparison
   * 1. Input/Outputs pin

|  |  |  |
| --- | --- | --- |
| PORT name | Attribute | Descripition |
| Count\_up\_down | Input | Signal indicating the counting direction  0: Count up  1: Count down |
| TCNT\_out | Input | The current 8-bit value of the counter, ranging from 0 to 255. This is the main output reflecting the counter’s state. |
| TMR\_OVF | Output | Timer counter overflow when counter 8’hFF to 8’h00: |
| TMR\_URF | Output | Timer counter underflow when counter 8’h00 down to 8’hff: |

* + 1. Functional/Protocol

The overflow/underflow comparison mechanism operates based on a specific functional protocol, as follows:

**Overflow Detection**:

* When the counter is configured to count up (count\_direction = 1), the system monitors the TCNT\_out.
* If counter\_value reaches its maximum limit (255 or 8'hFF) and the counter attempts to increment further, the overflow condition is triggered.

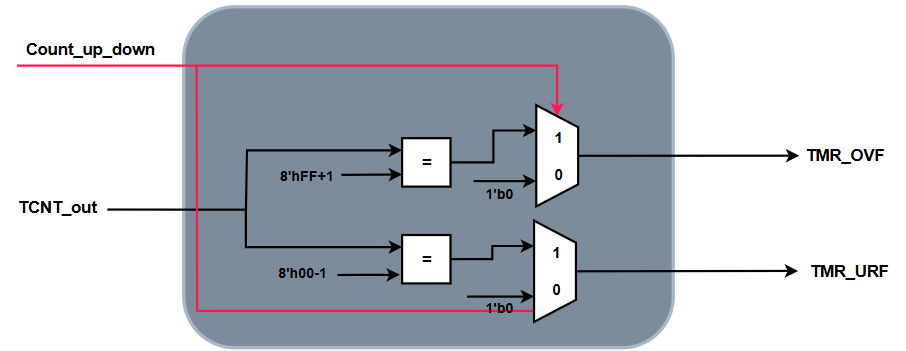
At that point, the overflow flag (TMR\_OVF) is set to 1, indicating that the counter has exceeded its maximum capacity.

**Underflow Detection:**

* When the counter is configured to count down (count\_direction = 0), the system monitors the TCNT\_out.
* If counter\_value reaches its minimum limit ( 0 or 8'h00) and the counter attempts to decrement further, the underflow condition is triggered.
* At that point, the underflow flag (TMR\_UDF) is set to 1, indicating that the counter has gone below its minimum capacity.

**Event Handling:**

* The overflow\_flag and underflow\_flag can be used to trigger interrupts or other system responses, allowing these events to be handled appropriately (resetting the counter, adjusting timing, or logging the event).
  + 1. Design circuit



The design circuit for the overflow/underflow comparison mechanism typically includes the following components:

**Counter Module:**

* An 8-bit counter that increments or decrements its value based on the clock\_signal and count\_direction.
* The counter outputs the current value (counter\_value) for comparison.

**Comparison Logic:**

* **Overflow Comparator:** A comparison circuit checks whether counter\_value equals the maximum value (e.g., 8'hFF). An equality comparator can be used.
* **Underflow Comparator:** A comparison circuit checks whether counter\_value equals the minimum value (e.g., 8'h00).

**Condition Logic:**

* **Overflow Condition:** An AND gate combines the output of the overflow comparator with the count\_direction signal. The overflow\_flag is set only when the counter is counting up and counter\_value = 8'hFF.
* **Underflow Condition:** An AND gate combines the output of the underflow comparator with the inverted count\_direction signal (using a NOT gate). The underflow\_flag is set only when the counter is counting down and counter\_value = 8'h00.

**Flag Generation:**

* The output from the condition logic is used to set the overflow\_flag and underflow\_flag, typically through a flip-flop or latch to ensure stable and clock-synchronized signals.

**Basic Circuit Diagram Flow:**

**Input:** Clock\_signal → Counter → Counter\_value

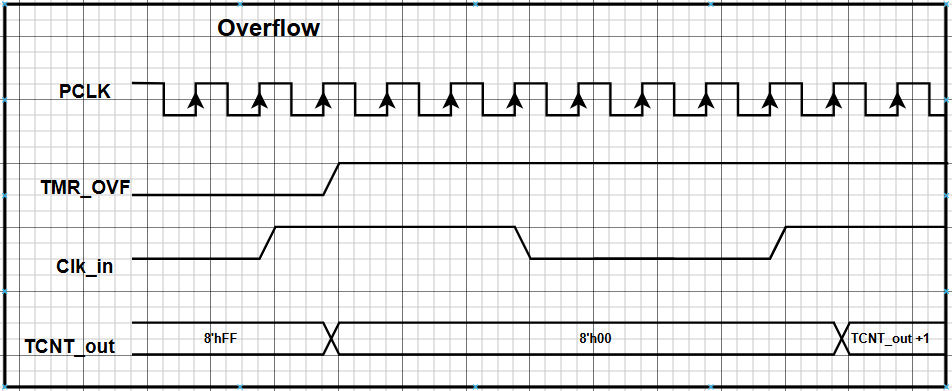
**Comparison:** Counter\_value compared with 8'hFF (for overflow) and 8'h00 (for underflow)

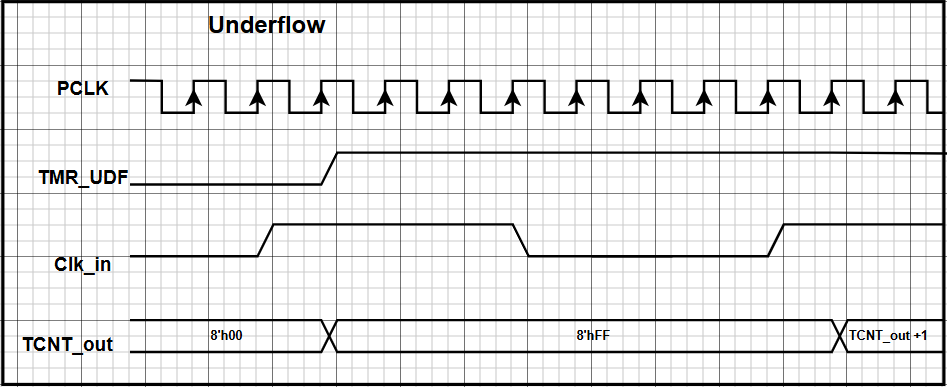
**Conditions:**

**Overflow:** (Counter\_value = 8'hFF) AND (count\_direction = 1) → TMR\_OVF

**Underflow:** (Counter\_value = 8'h00) AND (count\_direction = 0) → TMR\_UDF

* + 1. Timing chart





**Counting Up (**count\_direction = 1**):**

* TCNT\_out increases from 0 to 255 (8'hFF).
* When it reaches 255, the next increment causes an **overflow**.
* At this point, the overflow\_flag (TMR\_OVF) is set high for **one clock cycle**, and counter\_value wraps around to 0.
* The underflow\_flag (TMR\_UDF) remains at 0.

**Counting Down (**count\_direction = 0**):**

* TCNT\_out decreases from 255 down to 0 (8'h00).
* When it reaches 0, the next decrement causes an **underflow**.
* At this point, the underflow\_flag (TMR\_UDF) is set high for **one clock cycle**, and counter\_value wraps around to 255.
* The overflow\_flag (TMR\_OVF) remains at 0.

**Evaluate Code Coverage:**

