LAB05: Counters and Timers

Lab5A: Multi-Mode Timer

Module: multi_mode_timer
Purpose: It can act as a **basic countdown timer**, a **repeating timer**, or a **PWM generator**, depending on the selected mode. This makes it reusable across different digital systems where different timing behaviors are needed.

Modes supported:

1. **OFF mode (00)** – Timer is disabled.
2. **ONE-SHOT mode (01)** – Counts down once from reload_val to zero when triggered, then stops.
3. **PERIODIC mode (10)** – Continuously reloads and counts down, generating periodic timeout pulses.
4. **PWM mode (11)** – Generates a PWM waveform with period = reload_val and duty cycle = compare_val/reload_val.

Signals

Inputs

- clk → System clock input.
- rst_n → Active-low reset.
- mode → Selects the operating mode (00=OFF, 01=One-shot, 10=Periodic, 11=PWM).
- prescaler → Clock divider, used only in PWM mode to slow down counting.
- reload_val → The maximum count value for one-shot/periodic, or the period for PWM.
- compare_val → Duty cycle control for PWM (active-high portion of the cycle).
- start → Trigger signal to start one-shot/periodic operation.

Outputs

- timeout → Asserted when timer expires (end of countdown or PWM period).
- pwm_out → PWM output signal (active high for compare_val cycles).
- current_count → Current counter value, useful for debugging or monitoring.

Architecture:

The multi-mode timer architecture is composed of the following blocks:

1. **Clock and Prescaler:**

- The timer receives a base clock (e.g., 1 MHz).
- A prescaler divides the input clock frequency to achieve longer timing intervals.
- The prescaler value is programmable.

2. **Counter Register:**

- A 32-bit counter is loaded with a reload value when the timer starts or resets.

- The counter decrements on each prescaler tick.
- The current counter value is available for monitoring.

3. **Mode Control Logic:**

- Determines the behavior of the timer based on the mode input.
- Supports mode transitions (resetting the counter and prescaler whenever mode changes).
- Handles enabling and disabling of the timer.

4. **Timeout Generator:**

- In one-shot mode, generates a single pulse when the counter reaches zero.
- In periodic mode, generates a pulse at every counter rollover.

5. **PWM Generator:**

- In PWM mode, the counter compares with the compare_val to produce pwm_out.
- The reload value defines the PWM period, and the compare value defines the duty cycle.

6. **Control Signals:**

- start → initializes the counter and activates the selected mode.
- timeout → asserted for one clock cycle when a countdown completes.
- pwm_out → PWM waveform output in PWM mode.

Working Principle:

1. **Initialization:**-

- When reset(rst_n), all registers are cleared.
- When start = 1, the timer loads reload_val into the counter, resets prescaler, and enters the selected mode.

2. **Counting:**

- On each clock cycle, the prescaler increments.
- When prescalar equals prescalar value, it resets and decrements the main counter.

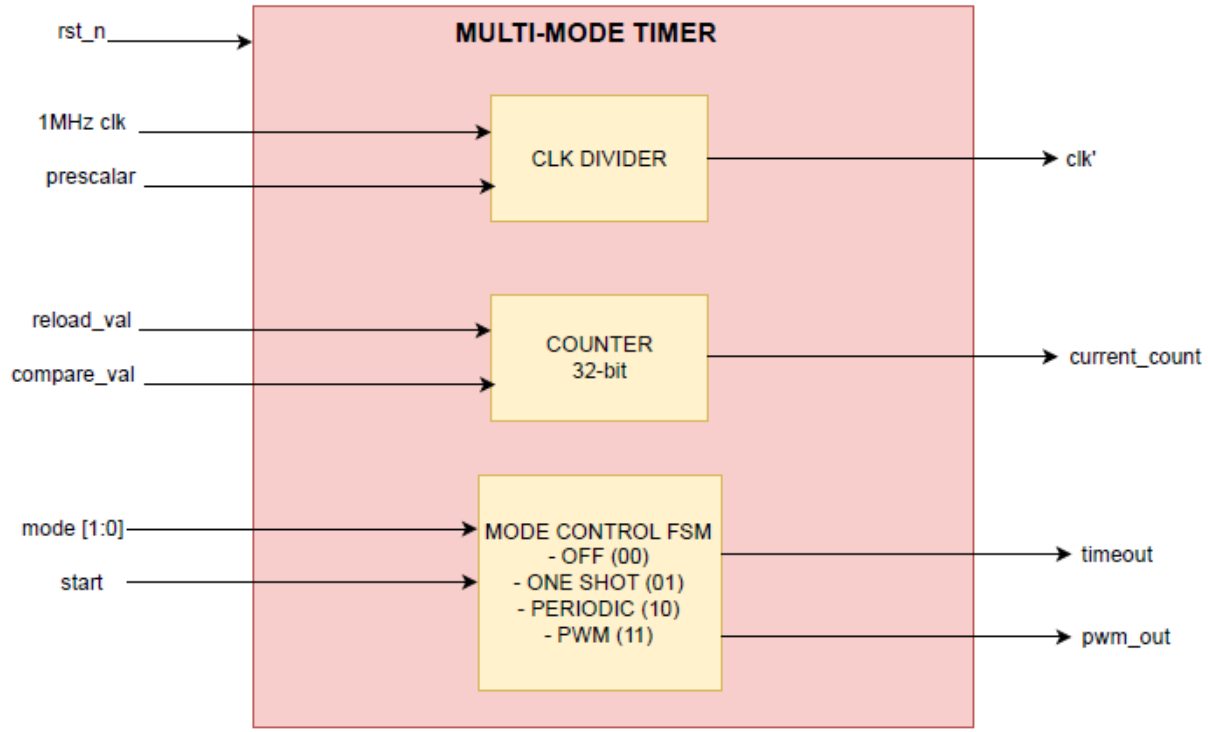
3. **Mode-specific behavior:**

- **Off Mode:** No activity, counter holds.
- **One-shot Mode:** Counter decrements until zero, asserts timeout, then stops.
- **Periodic Mode:** Counter decrements until zero, asserts timeout, then reloads automatically and continues.
- **PWM Mode:** Counter decrements; when it rolls over, it reloads. pwm_out stays HIGH if counter < compare_val, else LOW.

4. **Mode Change Handling:**

- If mode changes during operation, the timer resets: Counter reloads with reload_val, prescaler resets and PWM output clears.
- The new mode starts fresh, avoiding glitches.

Module Diagram:



Quality Checklist (specific to this module)

. **Consistent naming** – clear, descriptive signal names (reload_val, compare_val, etc.).
. **Proper hierarchy** – all logic encapsulated in one module, testbench is separate.
. **All outputs driven** – timeout, pwm_out, and current_count are always assigned.
. **No unintended latches** – all logic inside always_ff with reset.
. **Reset strategy** – rst_n cleanly resets all outputs and counters.
. **No combinational loops** – sequential logic only.