Lab 5A: Multi-Mode Timer

Objective

The objective of this lab is to design and verify a **32-bit programmable multi-mode timer**. The timer should support different operating modes, including **one-shot, periodic, and PWM generation**, while also incorporating a prescaler for input clock division and the ability to generate interrupts.

Specification

The key specifications of the timer are:

- **32-bit programmable timer** with multiple modes:
    - **One-shot Mode**: Counts down once and stops at zero.
    - **Periodic Mode**: Automatically reloads and restarts counting after reaching zero.
    - **PWM Mode**: Generates a PWM waveform with programmable duty cycle.
- **1 MHz input clock** with a **programmable prescaler** for flexible timing.
- **Reload mechanism** for setting initial counter value.
- **Compare register** for duty cycle calculation in PWM mode.
- **Timeout flag** asserted when the counter reaches zero (usable for interrupt generation).

Design Approach

1. **Prescaler**
    - A prescaler was designed to divide the 1 MHz clock.
    - It generates a **tick signal** that drives the main counter based on the programmed prescaler value.

2. **Mode Control Logic**
    - The timer uses a **2-bit mode input**:
        - `00`: Off Mode
        - `01`: One-Shot Mode
        - `10`: Periodic Mode
        - `11`: PWM Mode
    - A **finite state approach** was adopted to handle transitions and ensure correct behavior across modes.
3. **Counter Operation**
    - The counter loads from the reload value.
    - In one-shot mode, it decrements to zero and then stops.
    - In periodic mode, it reloads automatically after reaching zero.
    - In PWM mode, it decrements and reloads, while driving a PWM signal.
4. **PWM Duty Cycle Calculation**
    - A **compare value** is used to determine the duty cycle.
    - The PWM output is asserted high when the counter value is less than or equal to the compare value.
5. **Timeout Flag**
    - A timeout signal is asserted whenever the counter reaches zero in any active mode (except Off).
    - This can be extended to generate interrupts in a processor-based system.

Simulation and Verification

- **Simulation Environment**:
    - A testbench was created with a **1 MHz clock (1 µs period)**.
    - Different modes were tested sequentially: PWM, one-shot, and periodic.
    - Reload and compare values were programmed to validate correct behavior.
- **Observed Behavior**:
    - **PWM Mode**: A square wave was generated with the expected duty cycle (e.g., ~30% for chosen parameters).
    - **One-Shot Mode**: The timer counted down once and stopped at zero.
    - **Periodic Mode**: The timer repeatedly reloaded and restarted, producing continuous operation.
    - **Timeout Flag** was asserted correctly at counter zero.

Results and Discussion

- The multi-mode timer worked correctly in **all operating modes**.
- The **prescaler** allowed flexible clock division, enabling timing adjustments.
- **PWM output** matched the expected duty cycle, confirming correct compare logic.
- The **timeout flag** provides a reliable mechanism for interrupt-driven designs.
- The timer design is modular and can be integrated into larger systems, such as **embedded processors or SoC designs**.
- Future extensions could include:
    - Adding **capture/compare features**.
    - Enabling **multiple channels of PWM**.
    - Supporting **variable resolution prescalers**.

Conclusion

In this lab, a **32-bit multi-mode timer** was designed and verified in SystemVerilog. The timer successfully operated in **one-shot, periodic, and PWM modes**, with correct handling of reload values, duty cycle control, and timeout generation. Simulation results confirmed the correctness of the design. This project strengthened understanding of **timer architectures, prescalers, PWM generation, and FSM-based mode control**, which are widely used in digital and embedded systems.