Lab 3A: Programmable Counter

Objective

The objective of this lab is to design, implement, and verify an **8-bit programmable counter** with configurable limits. The counter should support both **up and down counting**, include **load and reset functionality**, and provide **status outputs** for zero detection and terminal count.

Design Requirements

- **Data Width**: 8-bit counter (0 to 255).
- **Programmable Limit**: Maximum count value is configurable using `max_count`.
- **Control Inputs**:
    - **load**: Loads a specific value into the counter.
    - **enable**: Enables/disables counting.
    - **up_down**: Selects counting direction. (`1` = up, `0` = down).
    - **reset**: Resets the counter to zero (synchronous).
- **Status Outputs**:
    - **zero detect**: Asserted when the counter reaches `0`.
    - **terminal count (tc)**: Asserted when the counter reaches `max_count` in up mode or `0` in down mode.
- **Operation**: Fully synchronous with clock.

Design Methodology

1 Counter Architecture

- The design uses a **Finite State Machine (FSM)** with four states:
    1. **IDLE**: Counter holds its value or resets.
    2. **LOAD**: Counter loads a specified value.
    3. **COUNT_UP**: Counter increments until `max_count`.
    4. **COUNT_DOWN**: Counter decrements until `0`.
- Transitions between states depend on **control inputs** (`load`, `enable`, `up_down`).

2 Control Logic

- **Reset**: Puts counter in `IDLE` state with `count = 0`.
- **Load**: Loads the `load_value`. If `load_value > max_count`, counter is clamped at `max_count`.
- **Enable + Up/Down**: Enables counting either upward or downward depending on direction.
- **Runtime Changes in max_count**: Counter checks if its current value exceeds new `max_count` and adjusts accordingly.

3 Reset Strategy

- **Synchronous Reset**: Counter resets on the rising edge of the clock when `rst_n` is low.
- This ensures predictable operation and avoids metastability issues.

4 Handling Metastability

- Since `load`, `enable`, and `up_down` are control signals, they are assumed to be synchronized with the system clock.
- If they originate from external asynchronous sources, synchronization flip-flops should be added.

5 Testbench Methodology

The design was verified using a **SystemVerilog testbench** with the following scenarios:

1. **Reset Condition** – counter resets to `0`.
2. **Load Operation** – counter loads value correctly, clamped if greater than `max_count`.
3. **Count Up** – counter increments from load/start value up to `max_count`.
4. **Count Down** – counter decrements down to `0`.
5. **Runtime Change of max_count** – counter adjusts automatically if `max_count` is modified during operation.
6. **Status Outputs** – verified correctness of **zero detect** and **terminal count** signals.

Results

1 Functional Verification

The programmable counter behaved as expected in all test cases:

- **Reset**: Output cleared to zero.
- **Load**: Correctly loaded user-defined values.
- **Count Up**: Incremented properly until terminal count.
- **Count Down**: Decremented properly until zero.
- **Runtime Change of max_count**: Counter adjusted to stay within new bounds.
- **Status Outputs**:
    - `zero` asserted only when `count == 0`.
    - `tc` asserted when `count == max_count` (up mode) or `count == 0` (down mode).

2 Sample Simulation Cases

- Load value = `2`, max_count = `5` → Counter incremented `2 → 3 → 4 → 5` then reset to `0`.
- Change max_count to `8` during operation → Counter respected new limit and continued counting.
- Down-count mode with load value `7`, max_count = `8` → Counter decremented `7 → 6 → ... → 0`, then wrapped to `8`.

Observations

- FSM-based design provides **clear control flow** and makes debugging easier.
- Handling runtime changes of `max_count` ensures flexibility.
- Synchronous reset guarantees stable operation.
- Counter correctly distinguishes between **shift-like resets (to 0)** and **programmable loads (custom values)**.

Conclusion

The **8-bit programmable up/down counter** was successfully designed, implemented, and verified. The counter supports **programmable limits, load functionality, enable/disable control, and direction selection**. Simulation confirmed correct operation under various test scenarios. The design can be extended for **higher bit-widths** or **additional features (e.g., pause/resume, interrupt generation)** for more advanced applications.