MOD10 Loadable Up-Down Counter Design and Verification

By

Pavan Kumar Gunnamsetti

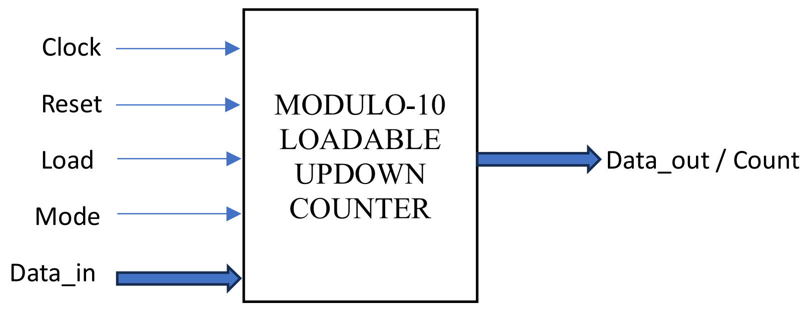
pavankumargunnamsetti@gmail.com

https://www.linkedin.com/in/pavankumargunnamsetti

https://github.com/Pavankumargunnamsetti

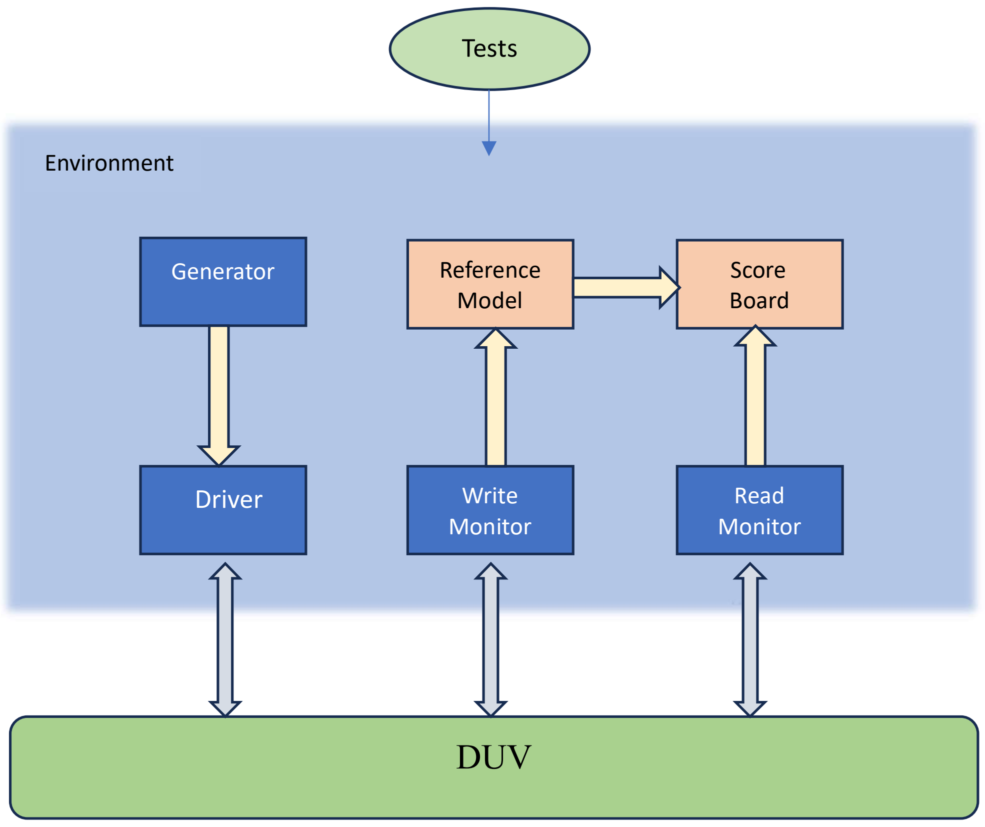
Verification of Modulo-10 Loadable Updown Counter

1. Block Diagram:-



A Modulo-10 Loadable Updown Counter is a digital counter that cycles through values 0 to 9, with the ability to load a specific value(data_in) when load signal is high, and count both upwards and downwards with respect to mode signal, when it is high, counter counts up, when it is not, counter counts down.

2. TB Environment



3. Verification Goals

- Verify the correct counting sequence for both up and down modes.
- Ensure proper loading of a specific value.
- Check for correct operation at the modulo boundary (resetting to 0 after 9 when counting up, and to 9 after 0 when counting down).
- Validate proper response to control signals (load, up/down, enable, and reset).

4. Testbench Setup

- DUT (Design Under Test): Instantiate the modulo 10 up/down counter.
- Clock Generation: Create a clock signal for synchronous operation.
- Reset Signal: Generate a reset signal to initialize the counter.
- Control Signals: Create signals for load, up/down, enable, and data input.

5. Strategies

1. **Up Counting**
    - Set the mode signal to 1.
    - Verify the count sequence: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, ...
2. **Down Counting**
    - Set the up/down signal to 'down'.
    - Verify the count sequence: 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, 9, 8, ...
3. **Load Functionality**
    - Load a specific value (e.g., 5).
    - Verify that the counter starts counting from the loaded value in both up and down modes.
4. **Boundary Conditions**

    1. **Up Counting at Boundary**
        - Count up to 9.
        - Verify that the next count is 0.
    2. **Down Counting at Boundary**
        - Count down to 0.
        - Verify that the next count is 9.

6. Control Signals

Reset

- Assert the reset signal.
- Verify that the counter resets to 0.

Load

- Verify that the counter load with data_in when load is high.

Mode

- Verify that the counter counts up when mode is high.
- Verify that the counter counts down when mode is low

7. Transaction

- Base class :- random variables : mode, load, reset, data_in

8. Transactors

- **Generator :-** generates random transactions and send the address of the packet to write driver through mailbox
- **Write Driver :-** collects the packet from Generator through mailbox and drives reset, load, mode, data_in to DUV by converting packet level to low level signals by drive method
- **Reference Model :-** collects the write monitor packet address from write monitor through mailbox and by load, mode, reset and data_in in packet it do counting by using static variable count , and passes that verification output packet to scoreboard
- **Read monitor :-** samples design output data_out from DUV and convert that low level signal to packet level, and passes that packet to Scoreboard
- **Write monitor :-** samples the input signals back from duv which are data_in , load, mode, reset and converts them to packet level and sends that packet address to reference model through mailbox.
- **Scoreboard :-** collects both the object addresses from read monitor and reference model through mailboxes and compare verification output and design output, and based on that it generates report

5. Coverage model

Functional Coverage: Ensure that all counting sequences, load operations, and control signal interactions are covered.

The following coverpoints and crosses have to be included in coverage

- Mode
- Load
- Reset
- Data_out /count
- Data_in – illegal_bins are from 10-15
- Load X Mode X Data_out

6. Call backs

- SB : Callback trigger the coverage model

7. Simulation and Results Analysis

- Simulate the testbench with different test cases.
- Analyze waveforms and logs to verify correct behavior.
- Ensure 100% coverage of the defined scenarios.