Verification Plan for Advanced Counter

Author: Omar Ashraf Abd El Mongy

This plan outlines the methodology for verifying the Advanced Counter design using a **System Verilog testbench with directed test cases, functional, code coverage, and waveform analysis**.

1. Verification Goals

- Verify correct functionality of the **4-bit counter** under all possible input conditions.
- Identify and fix any **design bugs**.
- Ensure **full functional coverage** by exercising all counter behaviors.

2. Testbench Components

The testbench will include the following:

1. **DUT Instance** – Instantiates the AdvancedCounter module.
2. **Clock & Reset Generator** – Provides periodic clock and reset sequences.
3. **Stimulus Generator** – Drives test cases for different input conditions.
4. **Monitor & Checker** – Observes count and checks expected behavior.
5. **Functional Coverage Model** – Ensures all design scenarios are tested.

3. Test Scenarios (Directed & Random Tests)

Test ID	Test Name	Description	Expected Result
TC_01	Reset Test	Assert rst, ensure count resets to 0.	count = 4'b0000
TC_02	Load Test	Set load = 1, load a specific value, and verify.	count = load_value
TC_03	Hold Test	Set hold = 1, verify counter holds its value.	count remains unchanged
TC_04	Increment Test	Set en_inc = 1, ensure counter increments correctly.	count = count + 1
TC_05	Decrement Test	Set en_dec = 1, ensure counter decrements correctly.	count = count - 1
TC_06	Simultaneous Load & Hold	load = 1 and hold = 1 at the same time.	**Priority: Load takes effect**
TC_07	Simultaneous Increment & Decrement	Set en_inc = 1 and en_dec = 1 simultaneously.	**No change in count**
TC_08	Boundary Test (Max Value)	Start from 4'b1111 and increment.	Remains 4'b1111 (no overflow)
TC_09	Boundary Test (Min Value)	Start from 4'b0000 and decrement.	Remains 4'b0000 (no underflow)

4. Debugging Strategy

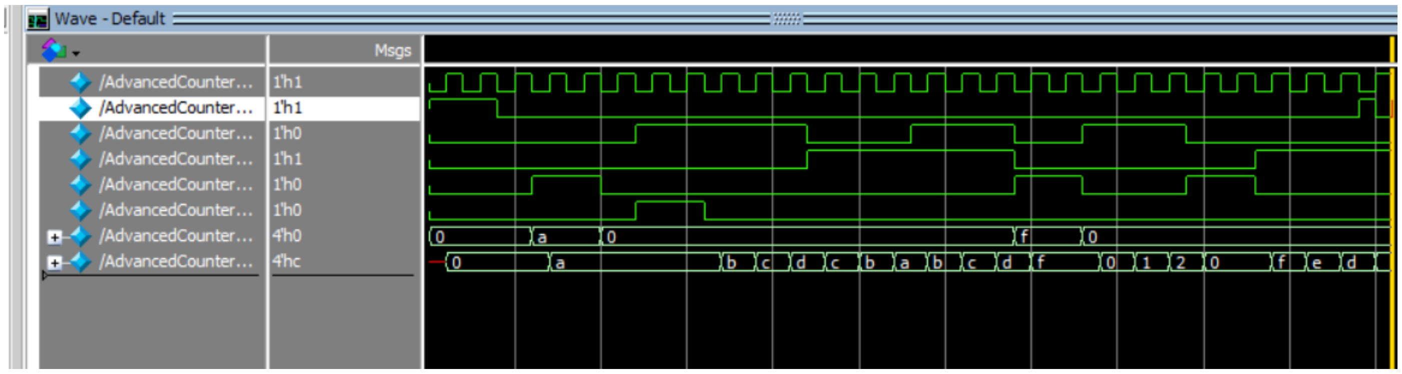
- **Waveform Analysis** using **Quesasim** to verify counter transitions.
- **Print Statements ($monitor, $display)** to track counter updates.
- **Functional Coverage Analysis** to ensure all cases are tested.

- GitHub Repository

The code for both the **Advanced Counter** design and its corresponding testbench is available on **GitHub**. You can access the repository using the following link:

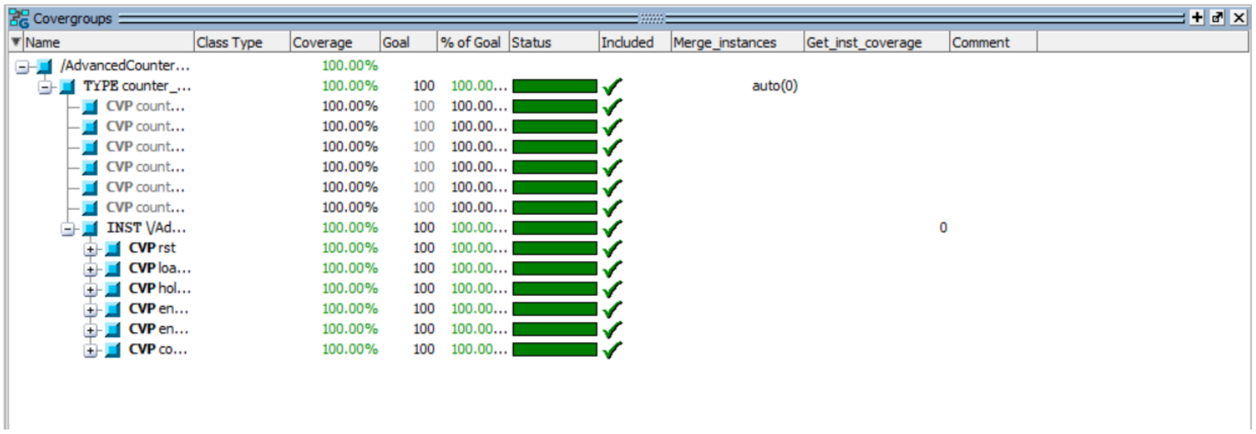
GitHub Repository - AdvancedCounter

- Waveform Generation


Wave - Default

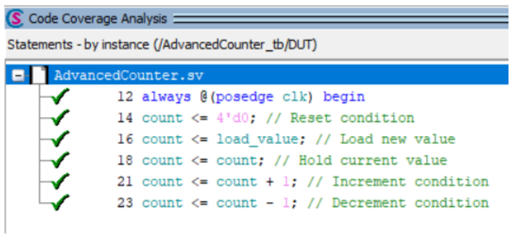
	Msgs
/AdvancedCounter...	1'h1
/AdvancedCounter...	1'h1
/AdvancedCounter...	1'h0
/AdvancedCounter...	1'h1
/AdvancedCounter...	1'h0
/AdvancedCounter...	1'h0
/AdvancedCounter...	4'h0
/AdvancedCounter...	4'hc

- Functional Collection


Covergroups

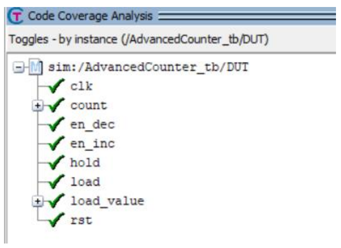
Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
/AdvancedCounter...		100.00%							
TYPE counter_...		100.00%	100	100.00...	🟩	✓	auto(0)		
CVP count...		100.00%	100	100.00...	🟩	✓			
CVP count...		100.00%	100	100.00...	🟩	✓			
CVP count...		100.00%	100	100.00...	🟩	✓			
CVP count...		100.00%	100	100.00...	🟩	✓			
CVP count...		100.00%	100	100.00...	🟩	✓			
CVP count...		100.00%	100	100.00...	🟩	✓			
INST \/Ad...		100.00%	100	100.00...	🟩	✓		0	
CVP rst		100.00%	100	100.00...	🟩	✓			
CVP loa...		100.00%	100	100.00...	🟩	✓			
CVP hol...		100.00%	100	100.00...	🟩	✓			
CVP en...		100.00%	100	100.00...	🟩	✓			
CVP en...		100.00%	100	100.00...	🟩	✓			
CVP co...		100.00%	100	100.00...	🟩	✓			

- Code Collection


Code Coverage Analysis

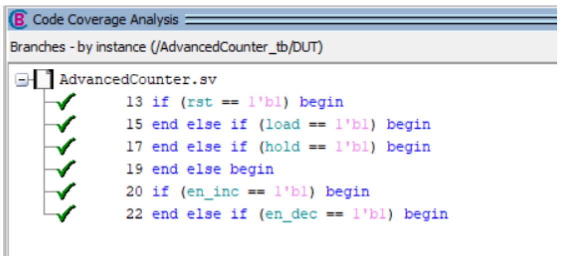
Statements - by instance (/AdvancedCounter_tb/DUT)

```
AdvancedCounter.sv
✓ always @(posedge clk) begin
✓ count <= 4'd0; // Reset condition
✓ count <= load_value; // Load new value
✓ count <= count; // Hold current value
✓ count <= count + 1; // Increment condition
✓ count <= count - 1; // Decrement condition
```


Code Coverage Analysis

Toggles - by instance (/AdvancedCounter_tb/DUT)

```
sim:/AdvancedCounter_tb/DUT
  ✓ clk
  ✓ count
  ✓ en_dec
  ✓ en_inc
  ✓ hold
  ✓ load
  ✓ load_value
  ✓ rst
```


Code Coverage Analysis

Branches - by instance (/AdvancedCounter_tb/DUT)

```
AdvancedCounter.sv
✓ if (rst == 1'b1) begin
✓ end else if (load == 1'b1) begin
✓ end else if (hold == 1'b1) begin
✓ end else begin
✓ if (en_inc == 1'b1) begin
✓ end else if (en_dec == 1'b1) begin
```