BIRZEIT UNIVERSITY

Faculty of Engineering and Technology

Electrical and Computer Engineering Department

Chip Design Verification ENCS5337 - SEC 1

Verification Plan

Prepared by

Abdelhameed Awad 1212478

Jehad Halahla 1201467

Masa Itmazi 1200814

Instructor

Dr. Ayman Hroub

BIRZEIT

May – 2024

Table of Contents

Functions to be verified

Compression Function:

The chip compresses input data from the (**data_in**) which is 80 bits in size, by comparing it with stored data in its memory. If a match is found, it sends the index of the stored data as the compressed version (**compressed_out**), and sets the (**response)** flag accordingly; otherwise, it stores the input data and assigns it a new index, changes reflect on response signal. Compressed data is 8 bits long. If memory is full, an error is signaled through the **(response)** bits.

Decompression Function:

The chip decompresses data by checking if the received compressed data **(compressed_in)** matches an index in its memory. If so, it retrieves the corresponding decompressed data, and streams it to **(decompressed_out)**; otherwise, it reports an error through the **(response)** bits. This ensures accurate restoration of data.

Error Handling:

Error Handling is really important in this project, there are multiple cases where we need to report an error signal '11' shown at the **response** at an invalid or an error case from the below cases:

1. **Compression error reporting:**
   - The compression fails when the memory is full.
   - The compression succeeds otherwise.
2. **Decompression error reporting:**
   - Decompression fails when the **compressed_in** data is larger than the index register, or in short, this index is not there.
   - Otherwise the decompression succeeds.
3. **Invalid Command "11":**
   - The response is an error if the command is not valid (command = "11")

4. **No operation (command = ‘00’):**
    - The response is 00 in this case (response = 00).
    - This Function doesn’t have any apparent error cases.

5. **Reset Functionality:**
    - This reset functionality is triggered when the reset input signal (synchronous reset).
    - The response for the reset is set to a “no valid output” (response = ‘00’).

Methods of Verification

1. Simulation-Based Verification:

- Verification will primarily rely on simulations to validate the functionality of the chip.
- Simulations involve running test scenarios and observing the behavior of the chip under various conditions.

2. Black Box Reference Model:

- A reference model will be developed to mimic the behavior of the chip under test.
- This reference model operates as a black box, abstracting away internal details of the chip's implementation such as the **clock**.
- It generates expected outputs based on given inputs without knowledge of the chip's internal workings.
- We will compare the reference model outputs to the RTL **offline.**

3. **Stimuli Generation:**
    - Test stimuli will be generated using a combination of pseudo-random and directed testing techniques.
    - Pseudo-random stimuli provide a broad range of inputs to explore various scenarios.
    - Directed tests target specific edge or corner cases to ensure thorough testing of critical functionalities.
4. **Offline Output Comparison:**
    - Both the chip under test and the reference model will be stimulated with the same set of test vectors.
    - Output responses generated by both the chip and the reference model will be captured and stored offline.
5. **Comparison Analysis:**
    - Offline output responses from the chip and the reference model will be compared.
    - Discrepancies or differences between the outputs will be identified, indicating potential bugs or deviations in the chip's behavior.
6. **Edge and Corner Cases Testing:**
    - Directed tests will specifically target edge or corner cases, such as boundary values or exceptional scenarios.
    - These tests aim to ensure that the chip behaves correctly under challenging conditions, enhancing its robustness and reliability.

7. **Verification Result:**
    - If the output responses of the chip match those of the reference model within acceptable tolerances, verification is deemed successful.
    - Any discrepancies found require further investigation and debugging to resolve potential issues in the chip's implementation.

This simulation-based verification approach offers a comprehensive method to validate the functionality and correctness of the chip under test, leveraging both black-box modeling and targeted testing strategies.

Completion Criteria

1. **Functional Coverage:**

- Ensure that the **data Compression/Decompression functionalities** are working 100%, with all the edge cases that are going to be described.
- Ensure that the reset works correctly and gives the desired results.
- Ensure that **no valid command/ no operation functions are** covered and give the desired results.
- Check the response for all the tested cases, and make sure its consistent.

2. **Code Coverage:**
    - We are aiming for a 95% - 100% code coverage for the System Verilog RTL code provided.

Required Resources

- UVM Environment:
  - Mentor Questa or Cadence Incisive simulation tool with UVM libraries installed
  - UVM-compliant methodology for testbench development
  - UVM-compatible assertions and functional coverage utilities
- Synopsis:
  - Synopsys VCS or Design Compiler for RTL simulation and synthesis
  - DesignWare Verification IP (VIP) for protocol-specific verification
  - Formality for formal verification checks if needed
- Waveform Viewer:
  - Synopsys Verdi or Mentor Graphics ModelSim for waveform visualization
  - Ability to analyze and debug signals at various levels of abstraction
  - Integration with simulation tools for easy waveform generation and debugging
- SystemVerilog Environment:
  - Any SystemVerilog-compliant simulator like Mentor Graphics Questa or Cadence Xcelium, or even EDA playground.
  - Support for SystemVerilog constructs including classes, interfaces, assertions, and coverage
  - Compatibility with UVM for seamless integration with the verification environment
  -

- Other Requirements:
    - Code coverage tool like Cadence Perspec for tracking functional coverage metrics
    - Assertions library such as SystemVerilog Assertions (SVA) for specifying design properties
    - Debugging tools like Synopsys VC SpyGlass for RTL linting and CDC analysis
    - Test case management system for organizing and managing verification test suites
    - Version control system like Git for collaborative development and tracking changes

Test Scenarios

Table 1:Test Scenario Table

Topic	Test #	Description
No Operation	1.1	Command is NoOp → '00'
Compression	2.1	Compression of data that is not in memory
	2.2	Compression that is already present in memory
	2.3	Compression when the memory is full and data is already in memory
	2.4	Compression when the memory is full and data is not already in memory
Decompression	3.1	Decompression when the data is found in memory
	3.2	Decompression when the data is not found in memory
Invalid operation	4.1	Command is '11'
Reset	5.1	Reset at the No OP
	5.2	Reset at Compression
	5.3	Reset at Decompression
	5.4	Reset at Invalid Command