Lab 4
Randomization

Module ID : CX-301

Design Verification

Instructor : Dr. Abid Rafique

Version 1.1

Document History

The changes and versions of the document are outlined below:

Version	State / Changes	Date	Author
1.0	Initial Draft	Jan, 2024	Qamar Moavia
1.1	Modified with new exercises	26 Jan, 2024	Qamar Moavia

Table of Contents

Objectives

By the end of this lab, students will be able:

- To use SystemVerilog scope-based randomization with constraints.
- To use SystemVerilog class-based randomization with constraints.

Tools

- SystemVerilog
- Synopsys VCS

Instructions for Lab Tasks

The required files for this lab can be found on the

```
share_folder/CX-301-DesignVerification/Labs/Lab4
```

Each lab task must be placed in its own directory. The submission must follow the hierarchy below, with the folder named after the trainee (no spaces), and the file names exactly as listed below.

```
./<yourname>/
      ├── Task1/
      │   ├── mem.sv
      │   ├── mem_interface.sv
      │   ├── mem_test.sv
      │   ├── top.sv
      ├── Task2/
      │   ├── mem.sv
      │   ├── mem_interface.sv
      │   ├── mem_test.sv
      │   ├── top.sv
      │   ├── transaction.sv
```

Along with that you also need to upload your solution on the github as well, and share the link.

TASK 1 : Using Scope-Based Randomization

In this task you will use SystemVerilog scope-based randomization with constraints.

Modify the memory testbench that is provided with this pdf document. In this task, you will use constrained scope-based randomization for data and address values.

Modifying the Memory Testbench

Working in the Task1 directory, perform the following.

1. Add a new **“Data = Random**” test to your testbench. In that test write and check random data for every address using a for loop. Simulate the design to confirm the randomization.**Note:** Use Randomize method for randomizing data.
2. Add a constraint to limit data to be a printable ASCII character (8'h20 - 8'h7F). Modify your read and write memory debug messages to print the character generated (use the %c format specifier). Check your constraint in simulation.
3. Add a constraint to limit data to be A-Z or a-z (8'h41-8'h5a, 8'h61-8'h7a). Check your constraint in simulation.
4. Apply weights to the constraints so that 80% of the time, randomization chooses an uppercase letter and 20% of the time it chooses a lowercase letter. Check your constraint in simulation.

TASK 2 : Using Class-Based Randomization

In this task you will use SystemVerilog class-based randomization with constraints.
Modify your Memory testbench to use constrained class-based randomization.

Adding Class-Based Randomization

Create a new directory named Task2. Copy your Memory and testbench files from Task1 to Task2.

Working in the Task2 directory perform the following:

1. Modify your Memory testbench to declare a class with two random properties for address and data in a new file transaction.sv. Declare the properties as bit arrays and use a rand or `randc` keyword whatever you think is suitable.
2. Add an explicit constructor with arguments to initialize the address and data properties.
3. Modify the Random Data test to use the class `randomize()` method to randomize the address and data properties. Write and read the memory using the class properties. Simulate and debug as needed.
4. Add a constraint block to your class to limit data to be a printable ASCII character (`8'h20 - 8'h7F`). Check your constraint in simulation.
5. Add a constraint to limit data to be `A-Z` or `a-z` (`8'h41-8'h5a, 8'h61-8'h7a`). Check your constraint in simulation.
6. Apply weights to the constraints so that `80%` of the time randomization chooses an uppercase letter and `20%` of the time, it chooses a lowercase letter. Check your constraint in simulation.

Forcing a Randomization Error

7. Force a randomization error by defining conflicting constraints for data.
    a. Run the simulation. Find the randomization failure warning message in the log file and check the conflicting constraints and affected variables are listed.

Adding a Control Knob

8. Add a property to the class to conditionally control the randomization constraints. A control property like this is known as a policy or control knob. Declare the property as an enum type with appropriate values. Use conditional constraints to select one of the following constraints based on the property value:
    a. A printable ASCII character
    b. An uppercase character `A-Z`
    c. A lowercase character `a-z`
    d. Either an uppercase (`80%` probability) or lowercase (`20%` probability) character
9. Simulate and debug as needed.

Good Luck 🙂