**Introduction**

SHA-3 stands for Secure Hash Algorithm-3. The input to the Hash Function is known as message and the output to the Hash Function is known as digest. At every instance, the length of the digest will be fixed but the length of the message can vary based on the user input. Hash Functions are always performed on bit strings i.e., binary data. There are basically six modes of operation in SHA-3 Hash Function, if combined, known as permutation engine, including four cryptographic functions and two XORs.

The first design of SHA-3 consists of din and dout of 200bit width and the second design has 64bit width. Total bits in the 5x5 matrix are 1600 bits, so design-1 requires (200²) 8 clock cycles to form the input matrix, whereas the design-2 requires (64²*5) total of 25 cycles to form the input matrix. The design-1 requires 16 clock cycles of latency to check for the first output, 8 clock cycles to provide data and another 8 for 24 cycles of permutation. Therefore from 17ᵗʰ clock cycle, where pushout is high, the output is ready to get sampled. Similarly, for the design-2 it samples the output when the output pins firstout and pushout are high.

**Methodology**

**SHA Block Verification**

**UVM test and UVM factory**

The UVM test is the place where it calls the sequence passing through the driver. Sequence has a pattern to check and verify functionalities of a design. UVM factory is a mechanism to improve flexibility and scalability to the testbench by allowing the user to substitute an existing class object by any of its inherited child class objects.

**SHA scoreboard**

Scoreboard is a verification component that contains checkers and verifies the overall DUT’s functionality at transactional level. A reference model can be used that depicts the DUT and Reference model’s output can be compared with DUT’s output, and it can be part of checker. Reference model is the one that calculates the expected value in form of golden vectors that is compared with the actual values from DUT.

The test is the place where the design one driver is replaced by the second design driver using the set type override and this is written in the build phase. Similarly, taking advantage of the factory override the monitor is also replaced.

**Methodology**

**SHA-3 sequences**

For design-1, we have three kinds of testcase scenarios. 1ˢᵗ case is the expected behavior by providing data when pushout is high. 2ⁿᵈ case is to randomize pushout to a particular weight to check whether design can distinguish valid from invalid data. 3ʳᵈ case is to verify design by providing a whole randomized data.

For design-2 we are verifying with two different test case scenarios. 1ˢᵗ case is to verify design with expected behavior by providing primary valid stimulus. 2ⁿᵈ case is to verify whether design can differentiate valid data from invalid ones with respect to pushout.

**Analysis and Results**

The checking for that data, is part of post processing of the data, and is carried in the report and check phases as a part of UVM phases. The connection between the scoreboards, FIFOs and analysis port from monitors is done in it’s parent’s connect phase, which is the environment’s connect phase in our case.

The waveform above shows the flow of the input sequence from the driver to the DUT and vice versa. The test stimulus was generated for different combination of inputs and is monitored using Scoreboard.

**Summary/Conclusions**

In this project UVM based test bench is developed suitable to test two different flavors of design having the same functionality but different interfaces. This project verifies the functionality of SHA-3 permutation engine which generates cryptographically encoded output. The results of simulations help to analyze design functionality and show that design works as expected to specification. As enhancement automation and remodelling, the test bench can be done to improve the verification of the design. Adding coverage monitors to track the verification closure, using assertions in the verification environment, or verifying the design using formal verification can be noted for future work.

**Key References**


**Acknowledgements**

We wish to express our sincere gratitude to Prof. Lili He, Professor Electrical Engineering for having welcomed us in Electrical Engineering Department. Thanks to her cooperation and advice in completion of our project. We are also thankful to Dr. Morris E. Jones, our project co-advisor for his guidance and continuous support.