System Verilog Assertions

LAB Material

Ashok B. Mehta

http://www.defineview.com

Copyright Notice

DefineView Consulting

http://www.defineview.com

Ashok B. Mehta

501 Pine Wood Lane

Los Gatos, CA 95032

(408) 309-1556

Email: ashok@defineview.com

Lab 3 …

good old fifo …

LAB 3 : FIFO

LAB Overview

A simple synchronous FIFO design is presented. FIFOs are some of the most commonly used design elements which require close scrutiny. FIFO assertions deployed directly at the source of a FIFO can greatly reduce the time to debug since these assertions point to the exact instance of fifo where an assertion fires.

LAB Objectives

1. *You will learn how to model various FIFO assertions that will be applicable to most any FIFO.*

2. *You will learn use of boolean expressions and sampled value functions as part of this exercise.*

LAB Design Under Test (DUT)

A simple synchronous FIFO design is presented as the DUT.

- *FIFO is 8 bit wide and 8 deep.*

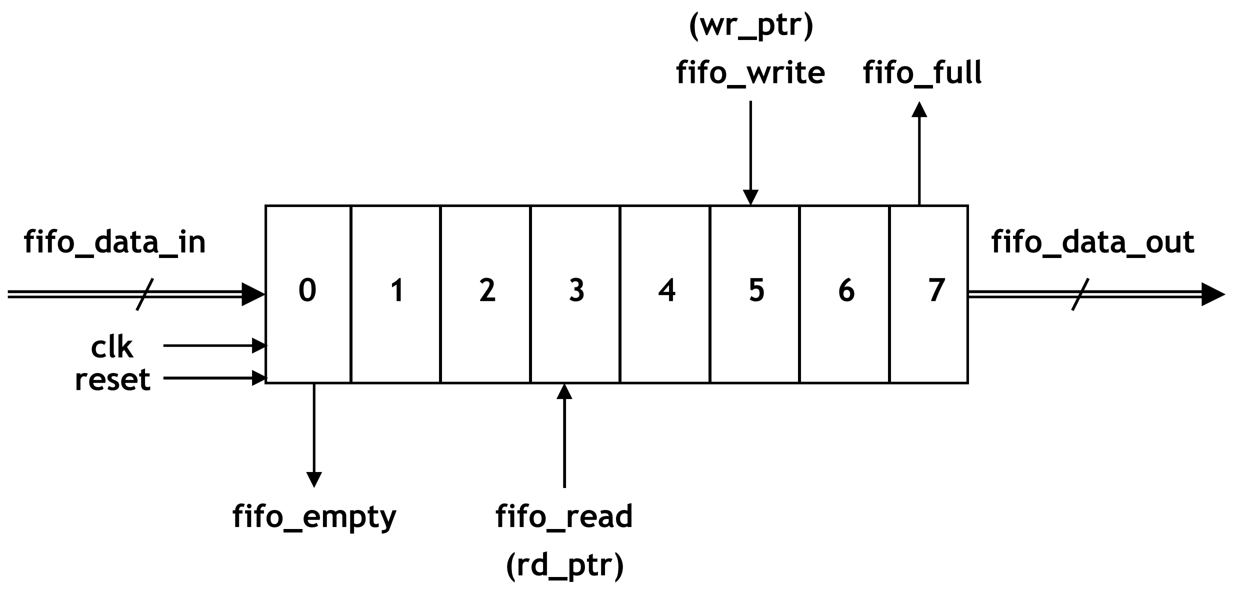
- *FIFO INPUTS*
  fifo_write, fifo_read, clk, rst_ and fifo_data_in[7:0]

- *FIFO OUTPUTS*
  fifo_full, fifo_empty, fifo_data_out[7:0]

LAB 3 : FIFO

FIFO Specs

- *fifo maintains a wr_ptr and a rd_ptr*
    - *wr_ptr increments by 1 everytime a write is posted to the fifo on a fifo_write request*
    - *rd_ptr increments by 1 everytime a read is posted to the fifo on a fifo_read request*
- *fifo maintains a 'cnt' that increments on a write and decrements on a read. It is used to signal fifo_full and fifo_empty conditions as follows*
    - *When fifo 'cnt' is >= 7, fifo_full is asserted*
    - *When fifo 'cnt' is 0, fifo_empty is asserted.*



LAB 3 : FIFO

LAB: Database

FILES:

1. *fifo.v :: Verilog RTL for 'fifo'*

2. *fifo_property.sv :: SVA file for fifo assertion.s*
   This is the file in which you will add your assertions.

3. *test_fifo.sv :: Testbench for the fifo.*
   Note the use of 'bind' in this testbench.

LAB: Assertions to Code

Code assertions to check for the following conditions in the 'fifo' design.

CHECK # 1. Check that on reset
rd_ptr=0; wr_ptr=0; cnt=0; fifo_empty=1 and fifo_full=0

CHECK # 2. Check that fifo_empty is asserted when fifo 'cnt' is 0.
Disable this property 'iff (!rst)'

CHECK # 3. Check that fifo_full is asserted any time fifo 'cnt' is greater than 7.
Disable this property 'iff (!rst)'

CHECK # 4. Check that if fifo is full and you attempt to write (but not read) that the wr_ptr does not change.

CHECK # 5. Check that if fifo is empty and you attempt to read (but not write) that the rd_ptr does not change.

CHECK # 6. Write a property to Warn on write to a full fifo

CHECK # 7. Write a property to Warn on read from an empty fifo

LAB 3 : FIFO

LAB: How to compile/simulate - step by step instructions

Follow the steps below to add your assertion for each check.

Then compile/simulate with each of your assertions and see that your results match with those stored in the ./.solution directory

Here's step by step instructions...

1. % cd <myDir>/SVA_LAB/LAB3

2. First run the design without any bugs introduced in it.

   % run_nobugs

   - This will create the file test_fifo_nobugs.log
   - Study this log to familiarize yourself with how the fifo works; when it reaches fifo_full, fifo_empty conditions, etc.

The remaining flow of the exericise is such that when you run any of the following steps, a specific bug is introduced in the design that your assertion should catch.

3. % vi fifo_property.sv

   - Look for `ifdef check1
   - Remove the 'DUMMY' property and code your property as specified in the comments
   - save the file and run the following simulation.

   % run_check1

   - If you have coded the property correct, you should see a failure for the CHECK #1 specified above.
   - Simulation will create test_fifo_check1.log
   - Compare test_fifo_check1.log with .solution/test_fifo_check1.log and see if your results match with the log in the .solution directory.
     - If your results don't match revisit your property and repeat step 3.

CONTINUED ➔

LAB 3 : FIFO

LAB: How to compile/simulate - step by step instructions - continued .

4. % vi fifo_property.sv

   - Look for `` `ifdef check2 ``
   - Remove the 'DUMMY' property and code your property as specified in the comments
   - save the file and run the following simulation.

   ```
   % run_check2
   ```

   - If you have coded the property correct, you should see a failure for the CHECK #2 specified above.
   - Simulation will create test_fifo_check2.log
   - Compare test_fifo_check2.log with .solution/test_fifo_check2.log and see if your results match with the log in the .solution directory.
   - If your results don't match revisit your property and repeat step 4.

5. % vi fifo_property.sv
   - Look for `` `ifdef check3 ``
   - Remove the 'DUMMY' property and code your property as specified in the comments
   - save the file and run the following simulation.

   ```
   % run_check3
   ```

   - If you have coded the property correct, you should see a failure for the CHECK #3 specified above.
   - Simulation will create test_fifo_check3.log
   - Compare test_fifo_check3.log with .solution/test_fifo_check3.log and see if your results match with the log in the .solution directory.
   - If your results don't match revisit your property and repeat step 5

CONTINUED ➔

LAB 3 : FIFO

LAB: How to compile/simulate - step by step instructions - continued .

6. % vi fifo_property.sv

   - Look for `ifdef check4
   - Remove the 'DUMMY' property and code your property as specified in the comments
   - save the file and run the following simulation.

   ```
   % run_check4
   ```

   - If you have coded the property correct, you should see a failure for the CHECK #4 specified above.
   - Simulation will create test_fifo_check4.log
   - Compare test_fifo_check4.log with .solution/test_fifo_check4.log and see if your results match with the log in the .solution directory.
   - If your results don't match revisit your property and repeat step 6.

7. % vi fifo_property.sv
   - Look for `ifdef check5
   - Remove the 'DUMMY' property and code your property as specified in the comments
   - save the file and run the following simulation.

   ```
   % run_check5
   ```

   - If you have coded the property correct, you should see a failure for the CHECK #5 specified above.
   - Simulation will create test_fifo_check5.log
   - Compare test_fifo_check5.log with .solution/test_fifo_check5.log and see if your results match with the log in the .solution directory.
   - If your results don't match revisit your property and repeat step 7

CONTINUED ➔

LAB 3 : FIFO

LAB: How to compile/simulate - step by step instructions - continued .

8. % vi fifo_property.sv

   - Look for `` `ifdef check6 ``
   - Remove the 'DUMMY' property and code your property as specified in the comments
   - save the file and run the following simulation.

   ```
   % run_check6
   ```

   - If you have coded the property correct, you should see a failure for the CHECK #6 specified above.
   - Simulation will create test_fifo_check6.log
   - Compare test_fifo_check6.log with .solution/test_fifo_check6.log and see if your results match with the log in the .solution directory.
   - If your results don't match revisit your property and repeat step 8.

9. % vi fifo_property.sv
   - Look for `` `ifdef check7 ``
   - Remove the 'DUMMY' property and code your property as specified in the comments
   - write the file and run the following simulation.

   ```
   % run_check7
   ```

   - If you have coded the property correct, you should see a failure for the CHECK #7 specified above.
   - Simulation will create test_fifo_check7.log
   - Compare test_fifo_check7.log with .solution/test_fifo_check7.log and see if your results match with the log in the .solution directory.
   - If your results don't match revisit your property and repeat step 9

DONE … CONGRATULATIONS