TFE4171 Design of Digital Systems 2 Spring 2019

Semester Project

First delivery time: Friday 12. April, 23:59
Redelivery will be accepted until Friday 26. April 23:59

Project description

You, verification engineers for SuperSoC AG, have been tasked with the verification of an untested HDLC (High-level Data Link Control) module to connect two devices together as part of a large project. You are provided with an HDLC implementation and in order to meet a hastily imposed implementation deadline thanks to your well-intentioned colleagues in the marketing department. It is important to retrofit the code with SystemVerilog Assertions to ensure that the design meets the specification before it is integrated.

You are given a lot of flexibility in what assertions you must write, but you must justify each assertion and provide coverage reports generated by the functional and code coverage facilities of SystemVerilog and Questasim. A simple testbench is included which can be used as a starting point.

The deliverables for the project are 1) a report (not more than 20 pages) which describes the verification approach taken by your team and 2) a zip file containing the assertion code including top-level modules and anything necessary to run the checks.

Specifically, the report should include:

- An introduction to the problem.
- A short description of how the HDLC module works.
- Your verification methodology:
  - A description of assertions added.
  - Assertion and coverage reports with a discussion.
  - Corner cases encountered.
  - Sources to property checker modules as appendices.

Grading

The project counts for 20% of the final grade and should be completed in groups of two students, as registered in the beginning of the course. Both students will get the same points. The points will be awarded based on the report delivered and additional analysis of source code and other supplementary materials if needed.

Learning outcomes

- Experience working on part of a larger project.
- Adding assertions at module design time.
- Adding assertions to legacy modules.
- Functional coverage:
  - Creating a functional coverage matrix.
  - Using cover, covergroups, coverpoints, and bins.

Specifications that needs to be verified

1. Correct data in RX buffer according to RX input.
   The buffer should contain up to 128 bytes (this includes the 2 FCS bytes, but not the flags).
2. Attempting to read RX buffer after aborted frame, frame error or dropped frame should result in zeros.
3. Correct bits set in RX status/control register after receiving frame.
   Remember to check all bits. I.e. after an abort the *Rx_ Overflow* bit should be 0, unless an overflow also occured.
4. Correct TX output according to written TX buffer.
5. Start and end of frame pattern generation (Start and end flag: 0111_1110).
6. Zero insertion and removal for transparent transmission.
7. Idle pattern generation and checking (1111_1111 when not operating).
8. Abort pattern generation and checking (1111_1110).
   Remember that the 0 must be sent first.
9. When aborting frame during transmission, Tx_AbortedTrans should be asserted.
10. Abort pattern detected during valid frame should generate Rx_AbortSignal.
11. CRC generation and Checking.
12. When a whole RX frame has been received, check if end of frame is generated.
13. When receiving more than 128 bytes, Rx_Overflow should be asserted.
14. Rx_FrameSize should equal the number of bytes received in a frame (max. 126 bytes = 128 bytes in buffer − 2 FCS bytes).
15. Rx_Ready should indicate byte(s) in RX buffer is ready to be read.
16. Non-byte aligned data or error in FCS checking should result in frame error.
17. Tx_Done should be asserted when the entire TX buffer has been read for transmission.
18. Tx_Full should be asserted after writing 126 or more bytes to the TX buffer (overflow).

Getting started

It is important to get started with the project early. The following are general steps you can follow to get started.

1. Ensure QuestaSim is set up correctly by having completed Exercises 1 and 2.
2. Copy the term project files from the source directory
   `cp -arv /home/courses/desdigsys2/2019s/dd2master/termproject/part-b/hdlc .`
3. Read and understand how the HDLC module works.
4. Expand testbench files with additional signals needed for verification.
5. Create tasks with input stimulus.
6. Write coverage and assertions.

Tips

- Write immediate assertions in test program for RX input vs. read data from RX buffer, and written data to TX buffer vs. TX output.
- Check correct values in registers before and/or after receive/transmit by using the read/write-tasks provided in the testbench.
- When verifying CRC, write an algorithm of the computation in a task and compare using immediate assertions.
- To view waveform of simulation, write `vsim -view vsim.wlf &` in command window to open Questasim. If the simulation is rerun, type `dataset restart -f` in transcript window in Questasim to update it.

Resources

- Questasim User Manual
- Karianne K. Kragseth. HDLC module Design Description. November 15, 2017. (on Blackboard)
- HDLC at Wikipedia.org `https://en.wikipedia.org/wiki/HDLC`
- SystemVerilog Language Reference Manual 3.1a (on Blackboard)
- Modelsim 6.0 Quick guide (contains all the commands used when scripting). `https://users.ece.cmu.edu/~kbiswas/qk_guide.pdf`
- CRC calculation at Wikipedia.org. `https://en.wikipedia.org/wiki/Computation_of_cyclic_redundancy_checks` and `https://en.wikipedia.org/wiki/Cyclic_redundancy_check`