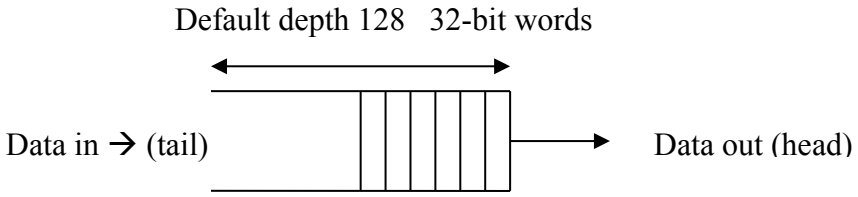
CSc/EEE 273
Spring 2019 Dr. Arad
***Term Project* (1000 points)**
11:59 pm May 1, 2019
10% Late Penalty
No late reports will be accepted after 11:59 pm on May 3, 2019

The objective of this project is to design, validate and synthesize a parameterized first-in first-out (FIFO) buffer in Verilog. The default depth of the FIFO is 128 32-bit words. It supports flush, *insert*, and *remove* operations. During the insert operation, a new word is added to the tail (end) of the buffer and an internal write pointer is incremented on positive edge of ***clk_in*** clock. The data at the head of the FIFO is read during the remove operation, and an internal read pointer is incremented on positive edge ***clk_out*** clock. A *flush* clears all words in the buffer. The buffer is considered empty after a flush. Here are the ports for the FIFO:

- 1-bit input ***reset:*** *An* asynchronous global system reset. Every sequential element in the system goes through a reset when this signal is asserted. The content of the FIFO is cleared. The status of the FIFO will be empty after a reset.
- 1-bit input ***flush***: When asserted, the FIFO is cleared, and the read/write pointers are reset on positive edge of clock ***clk_in***. The FIFO status is empty.
- 1-bit input ***insert:*** When asserted, data on ***data_in*** port is stored at the tail of the FIFO on positive edge of clock ***clk_in***
- 1-bit input ***remove***: When asserted, data at the head of the FIFO is removed from the FIFO and is placed on the on the ***data_out*** output port on positive edge of ***clk_out***
- 32-bit input ***data_in****:* input data port
- 32-bit output ***data_out****:* output data port
- 1-bit output ***full***: asserted on the positive edge of ***clk_in*** clock when the buffer is full and there is no more room for data
- 1-bit output ***empty***: asserted on the positive edge of ***clk_out*** clock when the buffer is empty (no data in the buffer)

Full and empty flags are used by the external environment for proper use of the FIFO



Project Phases:

1. **Design**

Your design must support the features stated above. You should consider time and area efficiency in your design. Use FSMs where appropriate. The design requires two out-of-phase clocks. ***clk_in*** is twice as fast as ***clk_out***. It is important to use synchronizers when signals cross clock domains.

2. Verification

You should develop a test fixture that can validate the FIFO using the automated verification techniques discussed in the course. You need to rely on code coverage to assess the extent of your verification.

3. Synthesis

You must synthesize your final design using 90nm technology library provided by Synopsys. Apply 0.3 ns and 0.4 as maximum and minimum delays to all ports. You must change the design to address any potential synthesis error and violation. Your report must include attempts for improving synthesis results through revising the design.

Team Work

You must work in groups of two students on this project. It is your responsibility to find your partner willing to work with you. Groups must work independently. No collaboration is allowed among groups at any level. If you have any question regarding the project you need to contact the instructor. No credit will be given for a project that is not the original work of the partners.

Deliverables

A. Report

On the due date each group must submit a typed report consisting of the items listed below as part of a single PDF file to Canvas. Each teammate should submit a copy of the same report.

1. A Cover sheet (Include Csc/EEE 273, Term Project, your name(s), percentage of overall contribution by each partner, date).
2. Table of contents (with page number).
3. Progress report (A template will be provided)
4. A professionally drawn diagram which shows the block diagram of your design. Properly label each component.
5. A professionally drawn diagram for any FSM used in the design.
6. A professionally drawn diagram which shows the block diagram of your testbench. Properly label each component.
7. Source code for each component of the DUT and its test fixture. Add sufficient comments to each module.
8. Simulation results, which verify the pre-synthesis functionality of each component and the overall system.
9. Code Coverage results.
10. Synthesis script used to synthesis the design

11. Synthesis reports created by Synopsys Design Compiler.

B. Softcopy submission

Guidelines for electronic submission of the Verilog code for DUT, testbench, and other synthesis and simulation related files will be provided later. These will be used to re-simulate and re-synthesize your deign.

A project demo is required during the last week of the semester. A schedule for the presentations will be provided once the teams are set.

Grading Policy

Project Report & Presentation	20%
Functionality of the individual components	30%
Functionality of the overall design	25%
Synthesis	25%