ECE-111: Advanced Digital Design Project: Homework 7 (Bonus)

Designs	Asynchronous FIFO (bonus)
Deadline	May 26, 2022 at 11:59pm
Max. late days	0

Overview

Homework 7 will be a bonus where you will develop a synthesizable SystemVerilog code for an Asynchronous FIFO.

We have provided a folder called **Lab7.zip** which contains the following:

Homework-7:

1. async_fifo.sv partial design template code
2. async_fifo_testbench.sv full code
3. dual_port_ram.sv full code
4. shift_register.sv full code

Assignment Tasks

This assignment requires you to complete the following tasks:

Recommended Tasks:

- Go over discussion video and discussion slides that go over this homework.

For Homework-7:

- **Develop SystemVerilog RTL model for M-bit width and N-depth Asynchronous FIFO:**
  - Asynchronous FIFO RTL model should be configurable to set the following mentioned parameters:
    - FIFO_DEPTH : This is the number of data locations FIFO's internal memory can store. FIFO DEPTH value should be a power of 2 (such as 2, 4, 8, 16, 32, and so on). Default value of FIFO DEPTH is set to 8 in the testbench.
    - DATA_WIDTH : Width of each data element which can be stored in FIFO's internal memory. By default, the value is set to 32 data width in Testbench.

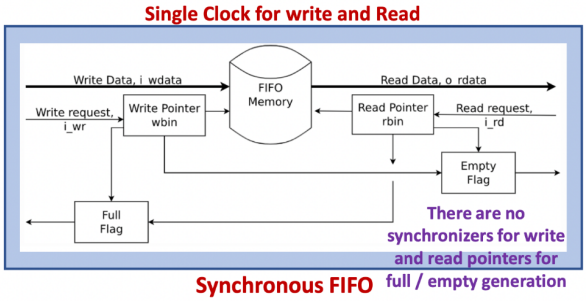
    - Support different clock frequencies for write and read. Write clock should be faster than read clock
        - In the testbench write clock is set to 20 ns (which is 50 Mhz clock) and the read clock is set to 40 ns (which is 25 Mhz)
        - In testbench for above mentioned default write and read clock frequency set NUM_OF_PACKETS=8
            - NUMBER_OF_PACKETS : Number of data elements which need to be transmitted through FIFO. By default set to '8' in the testbench. Use default values provided in testbench which is derived for wr_clk=20ns and rd_clk=40ns
    - Memory inside FIFO should be simple dual port memory with a single clock.
        - Memory should support synchronous write and asynchronous read operation
        - Use dual_port_ram module provided in LAB folder which implements above mentioned requirement
    - Use M-bit wide and N-Stage deep shift register for write and read pointer synchronization
        - Use shift_register module provided in LAB folder which can take M-bit of data and generate N-cycles delayed version of input data

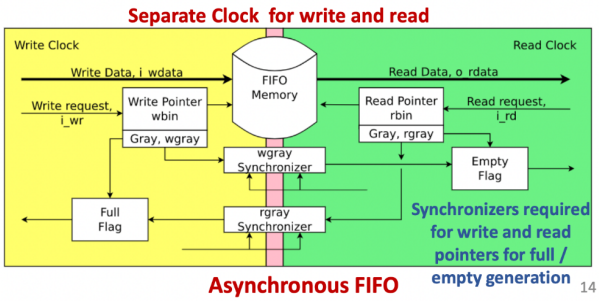
- **Primary Ports for Asynchronous FIFO design:**
    - **input** logic wr_clk, rd_clk : Write and Read Clocks
    - **input** logic reset : Asynchronous and active high reset
    - **input** logic wr_en : write enable, if wr_en == 1, data gets written to FIFO Memory
    - **input** logic rd_en : read_enable, if rd_en == 1, data gets read out from FIFO Memory
    - **input** logic [DATA_WIDTH-1:0] data_in : input data to be written to FIFO Memory
    - **output** logic [DATA_WIDTH-1:0] data_out : data read out from FIFO Memory
    - **output** logic fifo_full : indicates FIFO is full and there are no locations inside FIFO memory for further writes
    - **output** logic fifo_empty : indicates FIFO is empty and there are no data available inside FIFO memory for reading
    - **output** logic fifo_almost_full : one cycle early indication of FIFO_FULL (fifo is not full yet, it will be next cycle)
    - **output** logic fifo_almost_empty : one cycle early indication of FIFO_EMPTY (fifo is not empty yet, it will be next cycle)

- **Name of the asynchronous fifo module:** async_fifo

- Use below mentioned modules provided:
    - shift_register:
        - To implement 2-FF synchronizer for write and read pointer
        - To implement 1 bit delayed version of early fifo empty and fifo full
    - dual_port_ram:
        - For FIFO Memory Implementation

- **Basic difference between Synchronous and Asynchronous FIFO:**
  - In case of synchronous FIFO both write and read operation is performed on the same clock
  - In case of asynchronous FIFO, write operation and read operation of asynchronous FIFO are asynchronous to each other.
  - **Write and read clock can run independently with the same or different frequency!**





Submission Requirements

Submit a report on Canvas in PDF format which includes the following :

For Homework-7:

- SystemVerilog design code snapshot for async_fifo, shift_register and dual_port_ram modules
- Synthesis resource usage snapshot generated from Quartus for async_fifo top level module

- Simulation snapshot and explain simulation results of async_fifo
    - Describe how data is sent to async fifo and read from async_fifo. Explain how async fifo works.
    - Explain write and read pointer synchronization how it is done
    - Explain how full and empty flags are generated
    - Explain role of dual_port_ram in asycn_fifo design and how read and write operation is performed to dual_port_ram
- Explanation of FPGA resource usage in the report is not required.