Exercise 07: FIFO

Single Clock FIFO

FIFO (First-In, First-Out) is an object that allows to process stored entries in order where the first (oldest) stored entry is accessed first.

The FIFO can be used to rate-match data source and sink that operate at different rates. Often FIFOs are used to implement synchronization when transitioning from one clock domain to another (Clock Domain Crossing).

Implement Single Clock FIFO with the following interface:

```vhdl
entity scfifo is
generic (
    g_DATA_WIDTH : positive := 8;
    g_ADDR_WIDTH : positive := 4--;
);
port (
    o_rdata     : out   std_logic_vector(g_DATA_WIDTH-1 downto 0);
    i_rack      : in    std_logic;
    o_rempty    : out   std_logic;

    i_wdata     : in    std_logic_vector(g_DATA_WIDTH-1 downto 0);
    i_we        : in    std_logic;
    o_wfull     : out   std_logic;

    i_reset_n   : in    std_logic;
    i_clk       : in    std_logic--;
);
end entity;
```

The `o_rdata`, `i_rack` and `o_rempty` are read side ports for data, read acknowledge and read empty respectively. The `i_wdata`, `i_we` and `o_wfull` are write side ports for data, write enable and write full respectively.

The `g_DATA_WIDTH` defines the data width and `g_ADDR_WIDTH` is depth of the FIFO in power of 2 units (i.e. `DEPTH = 2^g_ADDR_WIDTH`).

The FIFO operates in so-called fall-through mode where the oldest valid entry is immediately visible on the read port. The FIFO is typically implemented with circular buffer where read position corresponds to current oldest value and write position corresponds to next empty slot. The read acknowledge advances the read position (discarding the current oldest value) and write enable advances write position.

The read empty signal shows that there is no data in the FIFO and any read attempt should be ignored. The write full signal shows that there is no free slots in the buffer and any write attempt should be ignored.

Instructions:

- Define an array (buffer) of appropriate size for FIFO entries and two registers for read and write pointers.
- When read and write pointers are equals the FIFO is in empty state.
- When write pointer is just behind the read pointer the FIFO is full.
- Implement clocked process with appropriate reset logic that writes to a buffer and updates read/write pointers.
- The read and write pointer should be incremented when read acknowledge and write enable are asserted respectively (take into account read empty and write full signals).

- The assignment to read data port should be performed outside clocked process such that current oldest value is seen on that port.

Test with the provided testbench and top file. Note that testbench uses `g_DATA_WIDTH := 5` and `g_ADDR_WIDTH := 3`.

Questions:

- What is the effective depth of the FIFO (maximum number of stored items) with the above definition of write full signal?
- The Dual Clock FIFO requires separate clocks for read and write sides. What needs to be considered when implementing such a FIFO?

The labs take place in Staudingerweg 9 building: group 1 in room 4-516 on Thurday and group 2 in Newton room on Friday.

Send your solutions through MOODLE (https://moodle.uni-mainz.de).

The solution should include:

- Source code (only **your** `.vhd` files)
- Brief description if applicable, questions, etc.
- Screen shot of the simulation (use simulation range of 1 microsecond)
- Submit your files in one `.zip` archive