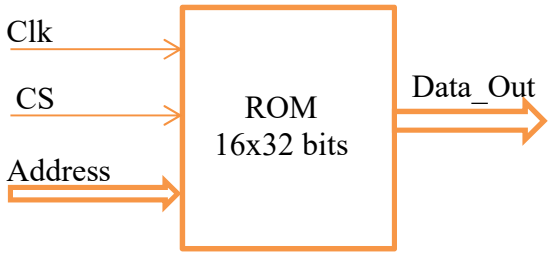
Lab #6: Memory Design

I. ROM Design

1. Requirement

Design ROM that has 16 locations each 32 bits wide. The 16 locations have the values like 0xDEADBEEF, 0xCAFEBABE, 0xDEADFEED and so on of your choice. There will be a *chipselect* (CS) input that activates the chip. The address input to the chip is a *vector*. The output would also be a *vector* that should send out the data already initialized at the active clock edge, depending on the address input to the chip.



The interface can be as below:

```
entity ROM_32Bits_Design is
      port(
            Clk:        in std_logic;
            CS:         in std_logic;
            Address:    in std_logic_vector(3 downto 0);
            Data_Out:   out std_logic_vector(31 downto 0)
      );
```

2. Pre-lab

- Study and analyze the working of a ROM
- Study how to do the type conversion in VHDL:
    - std_logic_Vector ⇔ signed/unsigned
    - signed/unsigned ⇔ integer
    - std_logic_vector ⇔ integer

3. Lab

- Write the VHDL code for this ROM design *(recommend using conversion functions located in **numeric_std** library for index conversion for array)*
- Simulate using Xilinx ISE simulator

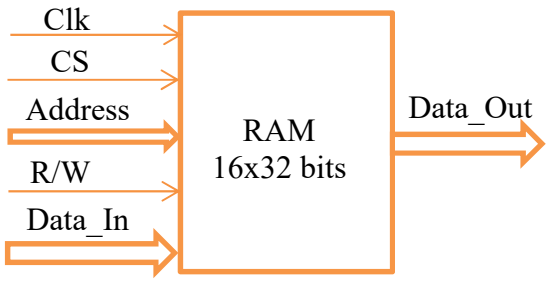
4. Deliverables

- VHDL program
- VHDL test bench with enough test cases
- Waveform with comments

II. RAM Design

1. Requirement

Write VHDL code for a RAM that has 16 locations each 32 bits wide. There will be a chipselect (CS) input that activates the chip. Another input to the circuit is an R/W which determines if the operation is a read or a write to the chip. The address input to the chip is a vector. The input and output would also be a vector(s) that should send and receive the data, depending on the address input to the chip.



The interface can be declared as below:

```
entity RAM_32Bits is
      port(
            Clk: in std_logic;
            CS: in std_logic;
            R_W: in std_logic;
            Address: in std_logic_vector(3 downto 0);
            Data_In: in std_logic_vector(31 downto 0);
            Data_Out: out std_logic_vector(31 downto 0)
      );
```

2. Pre-lab

- Study and analyze the working of a RAM

3. Lab

- Write the VHDL code for this RAM design design *(recommend using conversion functions located in **numeric_std** library for index conversion for array)*
- Simulate using Xilinx ISE simulator

4. Deliverables

- VHDL program
- VHDL test bench with enough test cases
- Waveform with comments

III. Bonus points (15 points): Optional

1. Requirement

Design a RAM with 32 locations each 32-bits wide by connecting 2 previous RAM designs together.

*Hint: use **port map** statements as introduced in lab#4, **MUX** and necessary logical gates to connect previous RAM design.*