Template code for READ/WRITE from ROM/RAM in VHDL

1 Basic visualization of memory

ROM or RAM can be simply visualized as a table with index<->value pairs. Index is the memory location and value is the data.

Table 1: RAM/ROM in table form

Index (decimal address)	Value (8 bit decimal data)
0	255
1	80
10	16
100	4
.	.
.	.
16383	0

2 Test bench to read from ROM

Following code iterate over block ROM (8-bit and size 16384) by varying the **rdaddress** signal and displaying the memory content onto **data** signal. Block ROM input signal are **clka** (mapped to **clock** signal), **addra** (mapped to **rdaddress** signal) and output signal **douta** (mapped to data signal).

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use ieee.numeric_std.all;


ENTITY tb_ROM_block IS
END tb_ROM_block;
ARCHITECTURE behavior OF tb_ROM_block IS
    COMPONENT blk_mem_gen_0
    PORT(
        clka : IN STD_LOGIC;
        addra : IN STD_LOGIC_VECTOR(13 DOWNTO 0);
        douta : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
        );
    END COMPONENT;
  --Inputs
   signal clock : std_logic := '0';
   signal rdaddress : std_logic_vector(13 downto 0) := (others => '0');
```

```
  --Outputs
   signal data : std_logic_vector(7 downto 0) := (others => '0');

   -- Clock period definitions
   constant clock_period : time := 10 ns;

   signal i: integer;
BEGIN
 -- Read image in VHDL
   uut: blk_mem_gen_0  PORT MAP (
          clka => clock,
          douta => data,
          addra => rdaddress
        );

   -- Clock process definitions
   clock_process :process
   begin
      clock <= '0';
      wait for clock_period/2;
      clock <= '1';
      wait for clock_period/2;
   end process;
   -- Stimulus process
   stim_proc: process
   begin
      for i in 0 to 16383 loop
        rdaddress <= std_logic_vector(to_unsigned(i, 14));
      wait for 20 ns;
      end loop;
      wait;
   end process;

END;
```

3 Test bench to write into the RAM

Following code write some value into the block RAM (8-bit and size 16384) by varying the **wraddress** signal and writing memory content using **data_in** signal. Block RAM input signal are **clka**, **addra**, **dina**, **write_enable** and output signal **douta**.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use ieee.numeric_std.all;


ENTITY tb_RAM_block IS
END tb_RAM_block;
```

```
ARCHITECTURE behavior OF tb_RAM_block IS
    COMPONENT blk_mem_gen_1
    PORT(
        clka : IN STD_LOGIC;
        wea : IN STD_LOGIC_VECTOR(0 DOWNTO 0);
        addra : IN STD_LOGIC_VECTOR(13 DOWNTO 0);
        dina : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
        douta : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
        );
    END COMPONENT;
  --Inputs
   signal clock : std_logic := '0';
   signal wraddress : std_logic_vector(13 downto 0) := (others => '0');
   signal data_in : std_logic_vector(7 downto 0) := (others => '0');
   signal wr: STD_LOGIC_VECTOR(0 DOWNTO 0) := (others => '0');
  --Outputs
   signal data_out : std_logic_vector(7 downto 0) := (others => '0');

   -- Clock period definitions
   constant clock_period : time := 10 ns;

   signal i: integer;
BEGIN
 -- Read image in VHDL
   uut: blk_mem_gen_1  PORT MAP (
          clka => clock,
          dina => data_in,
          wea => wr,
          douta => data_out,
          addra => wraddress
        );

   -- Clock process definitions
   clock_process :process
   begin
      clock <= '0';
      wait for clock_period/2;
      clock <= '1';
      wait for clock_period/2;
   end process;
   -- Stimulus process
   stim_proc: process
   begin
      for i in 0 to 16383 loop
        -- Write address for the RAM
        wraddress <= std_logic_vector(to_unsigned(i, 14));
        -- enabling write pin of the RAM
        wr <= "1";
        -- Input data for the memory location
        data_in <= std_logic_vector(to_unsigned(i mod 255, 8));
      -- waiting for 2 clock cycles to hold the RAM input signals
```

```
    wait for 20 ns;
    end loop;
    wait;
  end process;
```