Q1. Do question 7.1 (chapter 7).

7.1. Create an `i2std` (`integer` to `std_logic_vector`) function similar to the one for `i2bv` function.

The following code screenshot displays the function as well as its testbench.

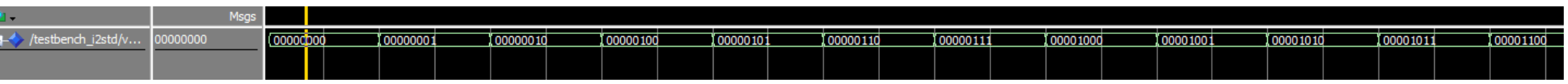
C:\Users\musil\OneDrive - Ohio University\First Year\Winter\Design of Digital Circuits\Module 6\homework\Question1\testBench_i2std.vhd - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

i2std_tester.vhd | testBench_i2std.vhd

```
1  -- q1 test bench from scratch
2
3  library ieee;
4  use ieee.std_logic_1164.all;
5
6  entity testBench_i2std is
7  end entity;
8
9  architecture my_behavior of testBench_i2std is
10
11 -- Declare the function to be tested
12 function i2std (myInteger, width : integer) return std_logic_vector is
13     variable result: std_logic_vector(width-1 downto 0):= (others => '0');
14     variable bits: integer:= width;
15     begin
16         for i in 0 to bits-1 loop
17             if ((myInteger/(2**i)) mod 2 = 1) then
18                 result(i) := '1';
19             end if;
20         end loop;
21     return (result);
22     end i2std;
23 -- Declare signals which will hold the resultant std_logic_vector
24
25 constant my_width : integer := 8;
26
27 signal varStorage : std_logic_vector(7 downto 0);
28
29 begin
30     varStorage <= i2std(0, my_width), i2std(1, my_width) after 20ns,
31                   i2std(2, my_width) after 40 ns, i2std(4, my_width) after 60 ns,
32                   i2std(5, my_width) after 80 ns, i2std(6, my_width) after 100 ns,
33                   i2std(7, my_width) after 120 ns, i2std(8, my_width) after 140 ns,
34                   i2std(9, my_width) after 160 ns, i2std(10, my_width) after 180 ns,
35                   i2std(11, my_width) after 200 ns, i2std(12, my_width) after 220 ns;
36 end my_behavior;
```

The function was tested using ModelSim

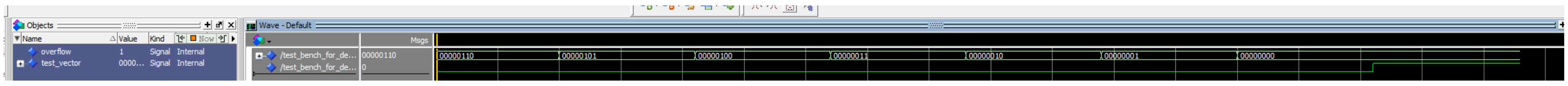


Q2. Do question 7.5 (chapter 7).

7.5. Create a procedure for decrementing bit_vectors. Also, create an underflow output for the procedure.

```
library ieee;
use ieee.numeric_bit.all;

entity test_bench_for_decrementer is
end test_bench_for_decrementer;

architecture behavioral of test_bench_for_decrementer is

    signal test_vector: bit_vector(7 downto 0) := X"06";
    signal overflow: bit:= '0';


    procedure decrement_a_bit_vector (
        signal bit_vector_in : in bit_vector(7 downto 0);
        signal bit_vector_out: out bit_vector(7 downto 0);
        signal underflow: out bit := '0'
        ) is

    begin

        if bit_vector_in = X"00" then
            underflow <= '1';
        else
            bit_vector_out <= bit_vector(unsigned(bit_vector_in) - 1);
        end if;
        wait for 1 ns;
    end decrement_a_bit_vector;

begin

    process is
    begin
        wait for 10 ns;
        decrement_a_bit_vector(test_vector, test_vector, overflow);
        wait for 10 ns;
        decrement_a_bit_vector(test_vector, test_vector, overflow);
        wait for 10 ns;
        decrement_a_bit_vector(test_vector, test_vector, overflow);
        wait for 10 ns;
        decrement_a_bit_vector(test_vector, test_vector, overflow);
        wait for 10 ns;
        decrement_a_bit_vector(test_vector, test_vector, overflow);
        wait for 10 ns;
        decrement_a_bit_vector(test_vector, test_vector, overflow);
        wait for 10 ns;
        decrement_a_bit_vector(test_vector, test_vector, overflow);
        wait for 10 ns;
        decrement_a_bit_vector(test_vector, test_vector, overflow);
        wait;
    end process;

end behavioral;
```

Tested in ModelSim



Q3. Write a procedure to perform 8-bit even parity check.

eight_bit_even_parity_checker.vhd

```vhdl
-- Description: Cascaded XOR operations implement an 8 parity check

-- Imports
library ieee;
use ieee.std_logic_1164.all;


entity test_bench_for_parity_checker is
end test_bench_for_parity_checker;

architecture behavioral of test_bench_for_parity_checker is
-- Signal Declarations

    -- External signals
    signal even_vector: std_logic_vector(7 downto 0) := b"01010101";
    signal odd_vector: std_logic_vector(7 downto 0) := b"01010111";
    signal parity_result: std_logic; -- Indicates even parity



-- Perform parity check

procedure eight_bit_parity_checker (
    signal vector_under_test : in std_logic_vector(7 downto 0);
    signal result: out std_logic := '0'
    ) is

    -- Internal signals
    variable a1, a2, a3, b1, b2, b3, f1: std_logic := '0';

    begin

    a1 := vector_under_test(0) xor vector_under_test(1);
    a2 := vector_under_test(2) xor vector_under_test(3);
    a3 := a2 xor a1;

    b1 := vector_under_test(4) xor vector_under_test(5);
    b2 := vector_under_test(6) xor vector_under_test(7);
    b3 := b2 xor b1;

    f1 := not(b3 xor a3);

    result <= f1;



    end eight_bit_parity_checker;

-- Test bench
begin

    process is
    begin
        wait for 10 ns;
        eight_bit_parity_checker(even_vector, parity_result);
        wait for 10 ns;
        eight_bit_parity_checker(odd_vector, parity_result);

    end process;

end behavioral;
```