Homework 6 – DSD2

**Exercise 1**

A picture containing timeline

Description automatically generated

library ieee;

use ieee.std\_logic\_1164.all;

entity Hw6Q1PriorityEncoder is

port (

w : in std\_logic\_vector (3 downto 0);

y : out std\_logic\_vector (1 downto 0)

);

end entity Hw6Q1PriorityEncoder;

architecture obehavior of Hw6Q1PriorityEncoder is

begin

if\_proc : process(w) is begin

if w = "0001" then

y <= "00";

elsif w = "0010" then

y <= "01";

elsif w = "0100" then

y <= "10";

else

y <= "11";

end if;

end process;

end obehavior;

**Exercise 2**

Graphical user interface

Description automatically generated

library ieee;

use ieee.std\_logic\_1164.all;

entity Hw6Q2PriorityEncoder is

port (

w : in std\_logic\_vector(7 downto 0);

y : out std\_logic\_vector(2 downto 0)

);

end Hw6Q2PriorityEncoder;

architecture Behavioral of Hw6Q2PriorityEncoder is

begin

case\_proc : process(w) is begin

case w is

when "00000001" => y <= "000";

when "00000010" => y <= "001";

when "00000100" => y <= "010";

when "00001000" => y <= "011";

when "00010000" => y <= "100";

when "00100000" => y <= "101";

when "01000000" => y <= "110";

when others => y <= "111";

end case;

end process case\_proc;

end Behavioral;

library ieee;

use ieee.std\_logic\_1164.all;

use IEEE.NUMERIC\_STD.ALL;

entity Hw6Q2PriorityEncoderTB is

end Hw6Q2PriorityEncoderTB;

architecture tb of Hw6Q2PriorityEncoderTB is

signal w : std\_logic\_vector(7 downto 0);

signal y : std\_logic\_vector(2 downto 0);

begin

dut : entity work.Hw6Q2PriorityEncoder

port map (

w => w, y => y

);

stimuli : process

-- use the unsigned type when doing math, requires numericstd library

variable test\_vector: unsigned(7 downto 0) := "00000001";

variable test\_result: unsigned(2 downto 0) := "000";

begin

for i in 0 to 7 loop

-- variables can be assigned to signals

-- unsigned is simply converted to std\_logic\_vector

w <= std\_logic\_vector(test\_vector);

wait for 10 ns;

test\_vector := shift\_left(test\_vector, 1);

assert y = std\_logic\_vector(test\_result)

report "error, expected y = " & to\_hstring(test\_result) &

"actual value is = " & to\_hstring(y);

test\_result := test\_result + 1;

Diagram

Description automatically generated wait for 10 ns;

end loop;

wait;

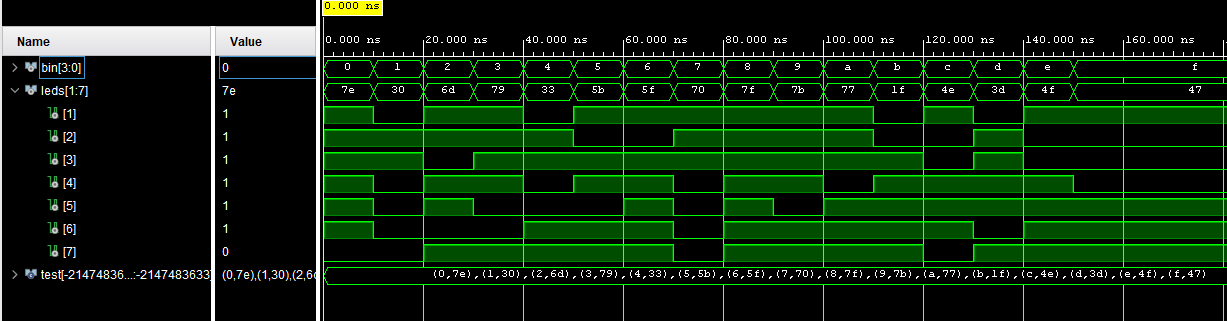
end process;

end tb;

**Exercise 3**

Diagram, schematic

Description automatically generated

**Exercise 4**

library ieee;

use ieee.std\_logic\_1164.all;

entity Hw6Q4SevenSegSolnTB is

end Hw6Q4SevenSegSolnTB;

architecture tb of Hw6Q4SevenSegSolnTB is

signal bin : std\_logic\_vector (3 downto 0) := (others => '0');

signal leds : std\_logic\_vector (1 to 7);

-- using unconstrained vectors in record

-- sized determined in constant

-- VHDL 2008 required

type test\_rec is record

bin : std\_logic\_vector;

leds : std\_logic\_vector;

end record test\_rec;

type test\_rec\_array is array (integer range <>) of test\_rec;

constant test : test\_rec\_array := (

("0000", 7UX"7E"),

("0001", 7UX"30"),

("0010", 7UX"6D"),

("0011", 7UX"79"),

("0100", 7UX"33"),

("0101", 7UX"5B"),

("0110", 7UX"5F"),

("0111", 7UX"70"),

("1000", 7UX"7F"),

("1001", 7UX"7B"),

("1010", 7UX"77"),

("1011", 7UX"1F"),

("1100", 7UX"4E"),

("1101", 7UX"3D"),

("1110", 7UX"4F"),

("1111", 7UX"47")

);

begin

dut : entity work.seven\_segment\_decoder

port map (bin => bin, leds => leds);

stimuli : process

begin

for i in test'range loop

bin <= test(i).bin;

wait for 10 ns;

assert leds = test(i).leds report "bin = " & to\_hstring(bin) & " leds = " & to\_hstring(leds) severity note;

end loop;

wait;

end process;

end tb;

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity seven\_segment\_decoder is

port(

bin : in std\_logic\_vector(3 downto 0);

leds : out std\_logic\_vector(6 downto 0)

);

end seven\_segment\_decoder;

architecture Behavioral of seven\_segment\_decoder is

begin

leds <= "1111110" when bin = "0000" else

"0110000" when bin = "0001" else

"1101101" when bin = "0010" else

"1111001" when bin = "0011" else

"0110011" when bin = "0100" else

"1011011" when bin = "0101" else

"1011111" when bin = "0110" else

"1110000" when bin = "0111" else

"1111111" when bin = "1000" else

"1111011" when bin = "1001" else

"1110111" when bin = "1010" else

"0011111" when bin = "1011" else

"1001110" when bin = "1100" else

"0111101" when bin = "1101" else

"1001111" when bin = "1110" else

"1000111" when bin = "1111" else

"1111111";

end Behavioral;

**Exercise 5**

Graphical user interface

Description automatically generated

library ieee;

use ieee.std\_logic\_1164.ALL;

use ieee.numeric\_std.all;

entity Hw6Q5Mux is

port (

shift\_amt: in std\_logic\_vector(2 downto 0);

y: out std\_logic\_vector(15 downto 0)

);

end Hw6Q5Mux;

architecture oh\_behavior of Hw6Q5Mux is

-- create an array of vectors to hold each of the shifters

type shifty\_array is array (7 downto 0) of std\_logic\_vector(15 downto 0);

signal y\_array: shifty\_array := (x"89AB", x"789A", x"6789", x"5678", x"4567", x"3456", x"2345", x"1234");

begin

y <= y\_array(to\_integer(unsigned(shift\_amt)));

end oh\_behavior;

library ieee;

use ieee.std\_logic\_1164.all;

entity Hw6Q5MuxTB is

end Hw6Q5MuxTB;

architecture tb of Hw6Q5MuxTB is

signal shift\_amt : std\_logic\_vector(2 downto 0) := (others => '0');

signal y : std\_logic\_vector(15 downto 0);

-- this record is constrained

-- the index ranges are given

type test\_rec is record

shift\_amt : std\_logic\_vector(2 downto 0);

y : std\_logic\_vector(15 downto 0);

end record test\_rec;

type test\_rec\_array is array(7 downto 0) of test\_rec;

constant test : test\_rec\_array := (

("000", X"1234"),

("001", X"2345"),

("010", X"3456"),

("011", X"4567"),

("100", X"5678"),

("101", X"6789"),

("110", X"789A"),

("111", X"89AB")

);

begin

dut : entity work.Hw6Q5Mux

port map (shift\_amt => shift\_amt, y => y);

stimuli : process

begin

for i in test'range loop

shift\_amt <= test(i).shift\_amt;

wait for 10 ns;

assert (test(i).y = y) report "error in result. y is " &

to\_hstring(y) & " and should be = " &

to\_hstring(test(i).y) & " when the shift amount is = " &

to\_hstring(test(i).shift\_amt);

wait for 10 ns;

end loop;

wait;

end process;

end tb;