Lab01 – Combinatorial circuits – Correction

1 Single output logic function: model & test-bench

1.1 Make your own model

```
library ieee;
use ieee.std_logic_1164.all;

entity myLogicFunction is port(
   a   : in std_logic;
   b   : in std_logic;
   c   : in std_logic;
   d   : in std_logic;
   e   : in std_logic;
   o   : out std_logic  -- Note that the last one doesn't have a semicolon at the end
);end myLogicFunction;

architecture arch of myLogicFunction is
begin
   o <= a and b and c and d and e; -- Concurrent assignment (here only one)
end arch;
```

1.2 Make your own model

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.math_real.all;

entity myLogicFunction_tb is
end entity;

architecture ben2 of myLogicFunction_tb is
component myLogicFunction is port(
    a, b, c, d, e     : in  std_logic;
    o  : out std_logic);
end component;
-- These are the internal wires
signal a, b, c, d, e, o: std_logic;
begin
    uut : myLogicFunction port map(a => a, b => b, c => c, d => d, e => e, o => o);
    stim : process
    variable inputs : std_logic_vector(2 downto 0);
    begin
      for i in 0 to 2**inputs'length - 1 loop
         inputs := std_logic_vector(to_unsigned(i, inputs'length));
         (a, b, c, d, e) <= inputs;
      wait for 10 ns;
      end loop;
    end process;
end ben2;
```

2 Logic functions with multiple outputs

```
library ieee;
use ieee.std_logic_1164.all;

entity myLogicFunction_v2 is port (
    a : in std_logic;
    b : in std_logic;
    c : in std_logic;
    o1 : out std_logic;
    o2 : out std_logic;
    o3 : out std_logic
); end myLogicFunction_v2;

architecture arch of myLogicFunction_v2 is
begin
    o1 <= a and b and c;
    o2 <= (a and b) or c;
    o3 <= a;
end arch;
```

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.math_real.all;

entity myLogicFunction_v2_tb is
end entity;

architecture ben3 of myLogicFunction_v2_tb is
component myLogicFunction_v2 is port(
    a,b,c     : in  std_logic;
    o1,o2,o3  : out std_logic);
end component;

signal a,b,c,o1,o2,o3 : std_logic;
begin
    uut : myLogicFunction_v2 port map(a => a, b => b, c => c, o1 => o1, o2 => o2, o3 => o3);
    stim : process
    variable inputs : std_logic_vector(2 downto 0);
    begin
      for i in 0 to 2**inputs'length - 1 loop
         inputs := std_logic_vector(to_unsigned(i, inputs'length));
         (a, b, c) <= inputs;
      wait for 10 ns;
      end loop;
    end process;
end ben3;
```

3 Design hierarchy (structured designs)

```
library ieee;
use ieee.std_logic_1164.all;

entity topLevel is port (
    a : in std_logic;
    b : in std_logic;
    c : in std_logic;
    o1 : out std_logic;
    o2 : out std_logic;
    o3 : out std_logic
); end topLevel;

architecture arch of topLevel is
    component myLogicFunction_v2 is port(
      a,b,c     : in  std_logic;
      o1,o2,o3  : out std_logic);
    end component;
signal m1,m2,m3 : std_logic;
begin
uut1 : myLogicFunction_v2 port map(a => a, b => b, c => c, o1 => m1, o2 => m2, o3 => m3);
uut2 : myLogicFunction_v2 port map(a => m1, b => m2, c => m3, o1 => o1, o2 => o2, o3 => o3);
end arch;
```

```
library ieee;
use ieee.std_logic_1164.all;

entity topLevel_tb is
end entity;

architecture ben of topLevel_tb is
component topLevel is port(
    a,b,c     : in  std_logic;
    o1,o2,o3  : out std_logic);
end component;


signal a,b,c,o1,o2,o3 : std_logic;
begin
    uut : topLevel port map(a => a, b => b, c => c, o1 => o1, o2 => o2, o3 => o3);
    stim : process
    begin

        a <= '0'; b <= '0'; c <= '0';
        wait for 10 ns;
        assert ((o1 = '0') and (o2 = '0') and (o3 = '0'))
        report "test failed for input combination 000" severity error;

        a <= '1'; b <= '1'; c <= '1';
        wait for 10 ns;
        assert ((o1 = '1') and (o2 = '1') and (o3 = '1'))
        report "test failed for input combination 111" severity error;

        a <= '0'; b <= '1'; c <= '0';
        wait for 10 ns;
        assert ((o1 = '0') and (o2 = '0') and (o3 = '0'))
        report "test failed for input combination 010" severity error;

    end process;
end ben;
```

4 Basic Half & Full adder circuits

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.math_real.all;

entity HalfAdder_tb is
end entity;

architecture ben of HalfAdder_tb is
component HalfAdder is port(
    a,b    : in  std_logic;
    s,c0   : out std_logic);
end component;
-- These are the internal wires
signal a,b,s,c0 : std_logic;
begin
    uut : HalfAdder port map(a => a, b => b, s => s, c0 => c0);
    stim : process
    variable inputs : std_logic_vector(1 downto 0);
    begin
      for i in 0 to 2**inputs'length - 1 loop
         inputs := std_logic_vector(to_unsigned(i, inputs'length));
         (a,b) <= inputs;
      wait for 10 ns;
      end loop;
    end process;
end ben;
```

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.math_real.all;

entity FullAdder_tb is
end entity;

architecture ben of FullAdder_tb is
component FullAdder is port(
    a,b,ci    : in  std_logic;
    s,c0   : out std_logic);
end component;
-- These are the internal wires
signal a,b,ci,s,c0 : std_logic;
begin
    uut : FullAdder port map(a => a, b => b, s => s, c0 => c0,ci => ci);
    stim : process
    variable inputs : std_logic_vector(2 downto 0);
    begin
      for i in 0 to 2**inputs'length - 1 loop
         inputs := std_logic_vector(to_unsigned(i, inputs'length));
         (a ,b, ci ) <= inputs;
      wait for 10 ns;
      end loop;
    end process;
end ben;
```

5 Ripple carry adders

```
library ieee;
use ieee.std_logic_1164.all;

entity AdderRC is generic(N : integer);
port (
    a : in std_logic_vector(N-1 downto 0);
    b : in std_logic_vector(N-1 downto 0);
    o : out std_logic_vector(N-1 downto 0);
    c0 : out std_logic
); end AdderRC;


architecture arch of AdderRC is
    component HalfAdder is port(
    a, b       : in std_logic;
    s, c0      : out std_logic);
  end component;
    component FullAdder is port(
    a, b, ci   : in std_logic;
    s, c0      : out std_logic);
    end component;
signal m1 : std_logic_vector(N-1 downto 0);
begin
uut : HalfAdder port map(a => a(0), b => b(0), s => o(0), c0 => m1(0));
loop : for i in 1 to N-1 generate
      uut2 :FullAdder port map (a => a(i), b => b(i), ci => m1(i-1) ,s => o(i), c0 => m1(i));
      end generate;
c0 <= m1(N-1);
end arch;
```

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.math_real.all;

entity AdderRC_tb is
end entity;

architecture ben of AdderRC_tb is
component AdderRC is generic (N : integer); port(
    a : in std_logic_vector(N-1 downto 0);
    b : in std_logic_vector(N-1 downto 0);
    o : out std_logic_vector(N-1 downto 0);
    c0 : out std_logic);
end component;



signal a,b,o : std_logic_vector(7 downto 0);
signal c0 : std_logic;
begin
    AdderRC_8 : AdderRC
    generic map(N => 8) port map(a => a, b => b, o => o, c0 => c0);
    stim : process
    variable inputs : std_logic_vector(7 downto 0);
    variable inputs2 : std_logic_vector(7 downto 0);
    begin
      for i in 0 to 2**inputs'length - 1 loop
           for j in 0 to 2**inputs'length - 1 loop
                inputs := std_logic_vector(to_unsigned(i, inputs'length));
                inputs2 := std_logic_vector(to_unsigned(j, inputs'length));
                a <= inputs; b <= inputs2 ;
                wait for 10 ns;
            end loop;
      end loop;
    end process;
end ben;
```