**EEE102**
**Sample Problems for Fall 2009**
**7-4-2008**
**Introduction to VHDL problems**

## Dataflow examples:

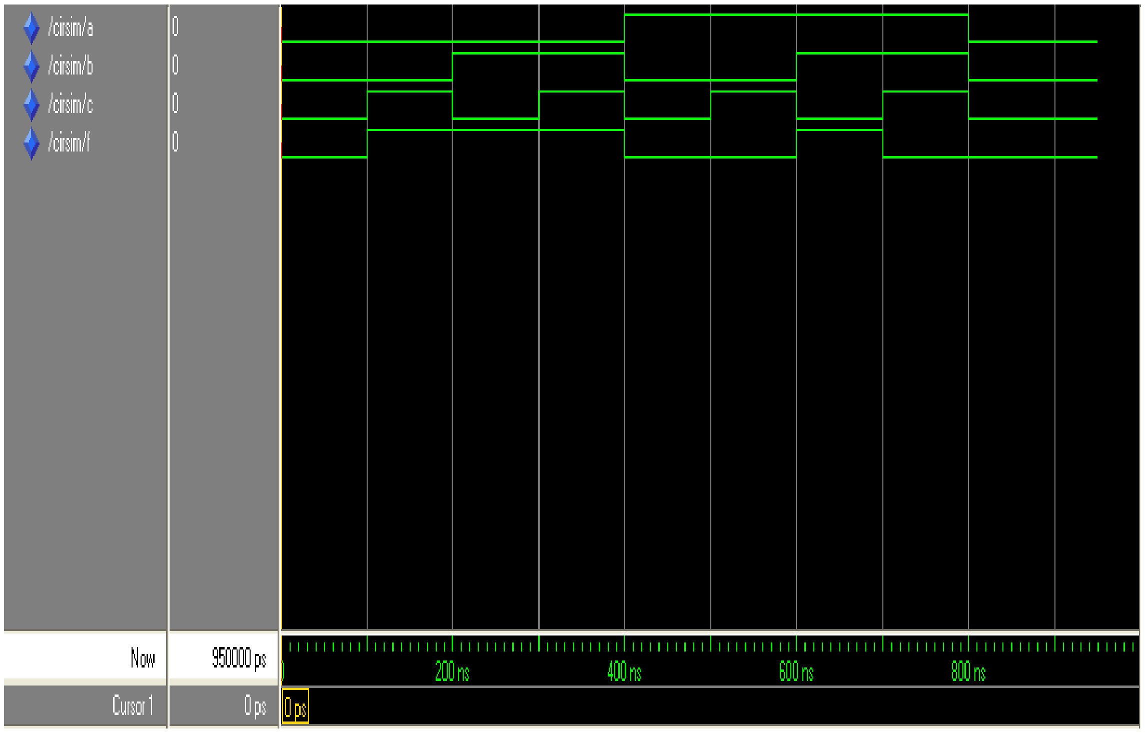
### Q1.

**Write VHDL code to describe the logic function F given by the following truth table. Do not attempt to make a design yourself. Make a functional simulation so that the waveform for F has logic values which appear with the same order they appear in the TT, from top to bottom. Copy and paste the VHDL code and also the simulation output graph into your homework solution.**

| A | B | C | F |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

### Solution:

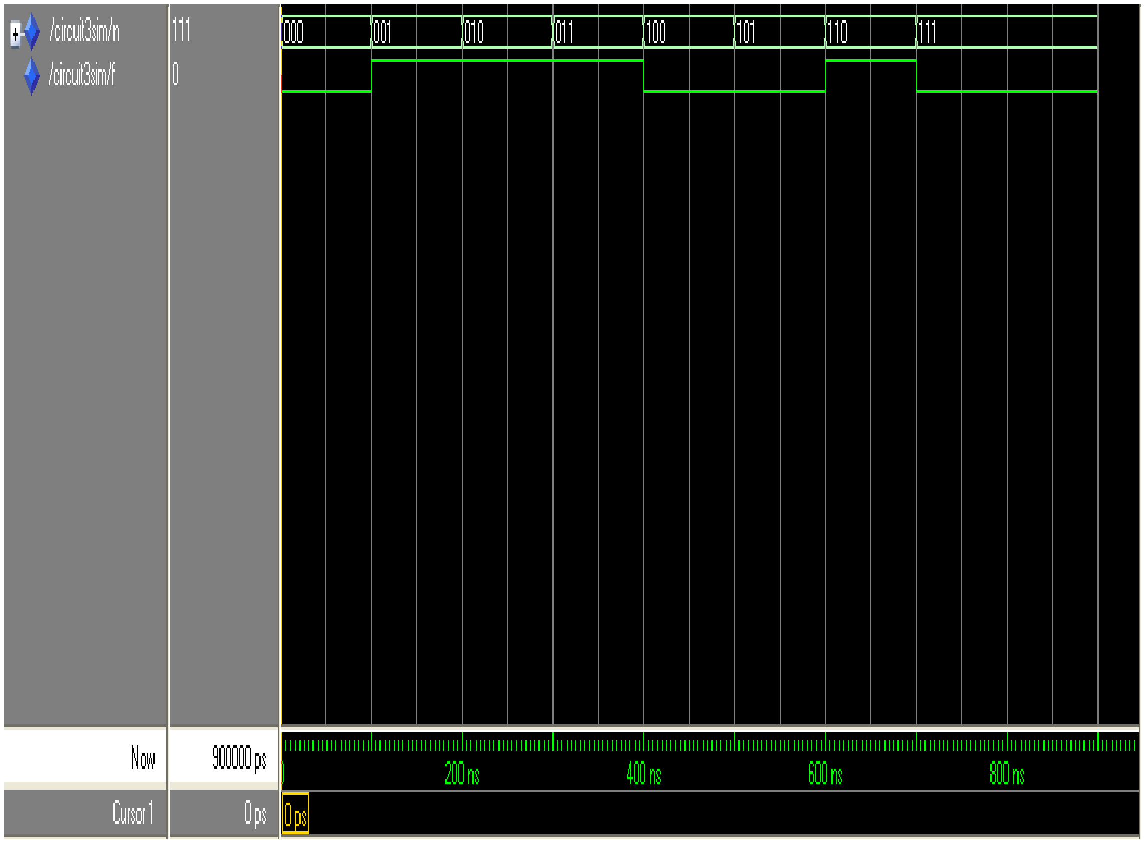
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity circuit is
   Port ( A : in std_logic;
          B : in std_logic;
          C : in std_logic;
          F : out std_logic);
end circuit;

architecture Behavioral of circuit is

begin
 F <= '1' when A='0' and B='0' and C='1' else
     '1' when A='0' and B='1' and C='0' else
          '1' when A='0' and B='1' and C='1' else
          '1' when A='1' and B='1' and C='0' else
          '0';
end Behavioral;
```



**<u>You can also define the inputs as a 3-bit binary number N = ABC</u>**

**The following solution also illustrates the use of functions.**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity circuit3 is
   Port ( N : in std_logic_vector(2 downto 0);
          F : out std_logic);
end circuit3;

architecture Behavioral of circuit3 is

begin

with CONV_INTEGER(N) select
         F<='1' when 1|2|3|6,
              '0' when others;

end Behavioral;
```



## Behavioral examples:

### Q2.

**Write VHDL code to desribe the logic funtion F given by the following truth table.**

| A | B | C | F |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

**In writing your VHDL code view the inputs A,B,C as a 3-bit vector signal. Use only a single process statement in your code.**

**Solution:**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity circuit2 is
```

```
   Port ( N : in std_logic_vector(2 downto 0);
          F : out std_logic);
end circuit2;

architecture Behavioral of circuit2 is

begin
        process(N)
        variable NI:integer;
        begin
                NI:=CONV_INTEGER(N);
                if NI=1 or NI=2 or NI=3 or NI=6 then F<='1'; else F<='0'; end if;
        end process;
end Behavioral;
```

## Q3.

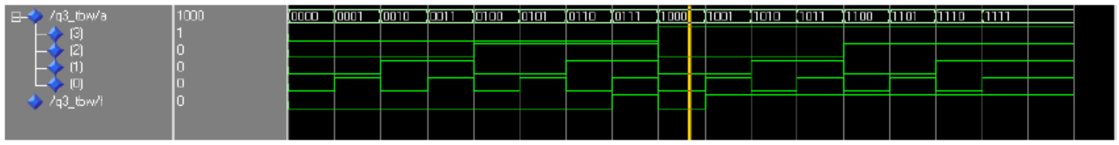
**Write a VHDL program for an electronic voting device for 4 people that performs the following function:**

- **When any one of the members A, B, C votes, it is counted as one vote**
- **When member D votes, it is counted as 2 votes.**
- **To vote for YES, the members press their buttons. If they do not press, it is assumed to be NO.**
- **The output, F, is 1 when there are at least 3 YES votes.**

**For example, when members A and D press their buttons, since D has two votes, total number of YES votes are 3 and the output should be 1.**

**Solution:**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity q3 is
Port ( A : in std_logic_vector(3 downto 0);
--A3 has 2 votes
F : out std_logic);
end q3;
architecture Behavioral of q3 is
begin
process (A)
begin
if A(3)='1' then F<= A(2) OR A(1) OR A(0);
else F<= A(2) AND A(1) AND A(0);
end if;
end process;
end Behavioral;
```

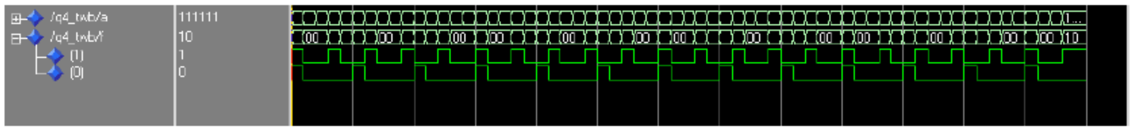


## Q4.

**(Exercise 5.24 from Wakerly 4th Edition, ignoring the last part of the question regarding the required resources) Write a VHDL program for a combinational logic function with six input bits A5–A0 representing an integer between 0 and 63, and two outputs F1 and F0 that indicate whether the integer is a multiple of 3 and 5, respectively.          .**

**Solution:**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity q4 is
Port ( A : in std_logic_vector(5 downto 0);
F : out std_logic_vector(1 downto 0));
end q4;
architecture Behavioral of q4 is
signal X,Y3,Y5: INTEGER;
begin
process (A,X,Y3,Y5)
begin
        X<=CONV_INTEGER (A);
        Y3<=X rem 3;
        Y5<=X rem 5;
        if Y3=0 then F(1)<='1';
        else F(1)<='0';
        end if;
        if Y5=0 then F(0)<='1';
        else F(0)<='0';
        end if;
end process;
end Behavioral;
```



## Structural examples:

### Q5.

a) **Find what the circuit "nice_module" described by the following VHDL code does. Draw its pin diagram.**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity nice_module is
   Port ( w : in std_logic_vector(1 downto 0);
          en : in std_logic;
          y : out std_logic_vector(0 to 3));
end nice_module;

architecture description of nice_module is
signal enw:std_logic_vector(2 downto 0);
begin
enw<=en&w;
with enw select
           y<="1000" when "100",
              "0100" when "101",
              "0010" when "110",
              "0001" when "111",
              "0000" when others;
end description;
```

**Note: & is the concatenation (linking together) operator.**
**For example "10" & "100" = "10100".**

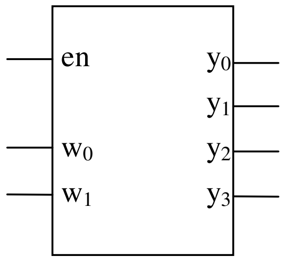
b) **Find what the circuit "big_circuit" described by the following VHDL code does. Draw its inside circuit using the "nice_module" as a block (component).**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity big_circuit is
   Port ( x : in std_logic_vector(2 downto 0);
          c : in std_logic;
          z : out std_logic_vector(0 to 7));
end big_circuit;

architecture Behavioral of big_circuit is
component nice_module
   Port ( w : in std_logic_vector(1 downto 0);
          en : in std_logic;
```

```
        y : out std_logic_vector(0 to 3));
end component;

signal A,B:std_logic;

begin
A<=c and not x(2);
B<=c and x(2);
label1:nice_module port map(x(1 downto 0),A,z(0 to 3));
label2:nice_module port map(x(1 downto 0),B,z(4 to 7));
end Behavioral;
```

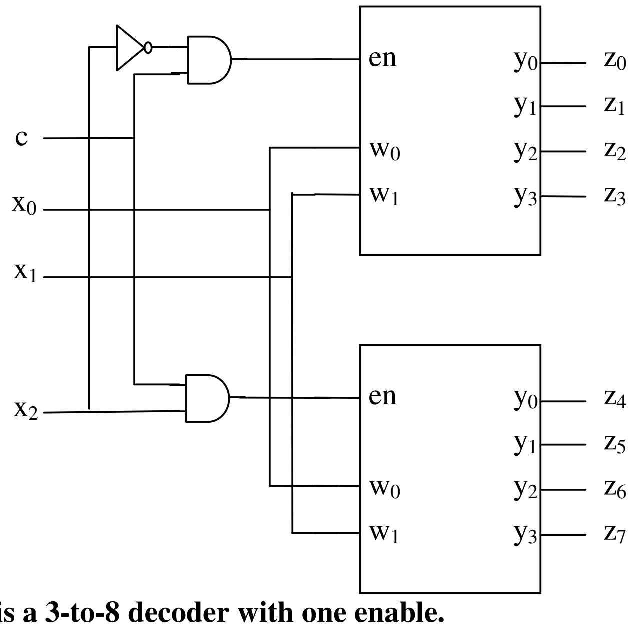
**Solution:**

**a)**



**"nice_module" is a 2-to-4 decoder with one enable.**

**b)**



**"big_circuit" is a 3-to-8 decoder with one enable.**

## Q6.

**An odd-parity module (entity) "odd_parity_8bit" for 8-bit binary numbers is already written using VHDL and saved.**

**Write VHDL code for the entity named "odd_parity_16bit" which describes an odd_parity circuit for 16-bit binary numbers. Use "odd_parity_8bit" module in your code.**
**The entity declaration of "odd_parity_8bit" is as follows:**

```
entity odd_parity_8bit is
        port (A:in std_logic_vector (7 downto 0);
              F: out std_logic);
end odd_parity_8bit;
```

**Notes: 1) What "odd_parity_8bit" does: If the number of 1 bits in A is odd then F = 1, else F = 0.**
**2) If X is declared as std_logic_vector (7 downto 0) then X(5 downto 1) is a 5-bit number picking the appropriate subrange of X.**
**3) It is not important for this question's purposes to include all necessary libraries.**
**4) Do not write "odd_parity_16bit" from scratch but make use of the previously written and saved "odd_parity_8bit".**

**Solution:**

```
entity odd_parity_16bit is
        port (A:in std_logic_vector (15 downto 0);
        F: out std_logic);
end odd_parity_16bit;

architecture structural of odd_parity_16bit is
component odd_parity_8bit
        port (A:in std_logic_vector (15 downto 0);
        F: out std_logic);
end component;
signal F8H, F8L: std_logic;
begin
        label1: odd_parity_8bit port map(A(15 downto 8), F8H);
        label2: odd_parity_8bit port map(A(7 downto 0), F8L);
        F <= F8H xor F8L;
End structural;
```

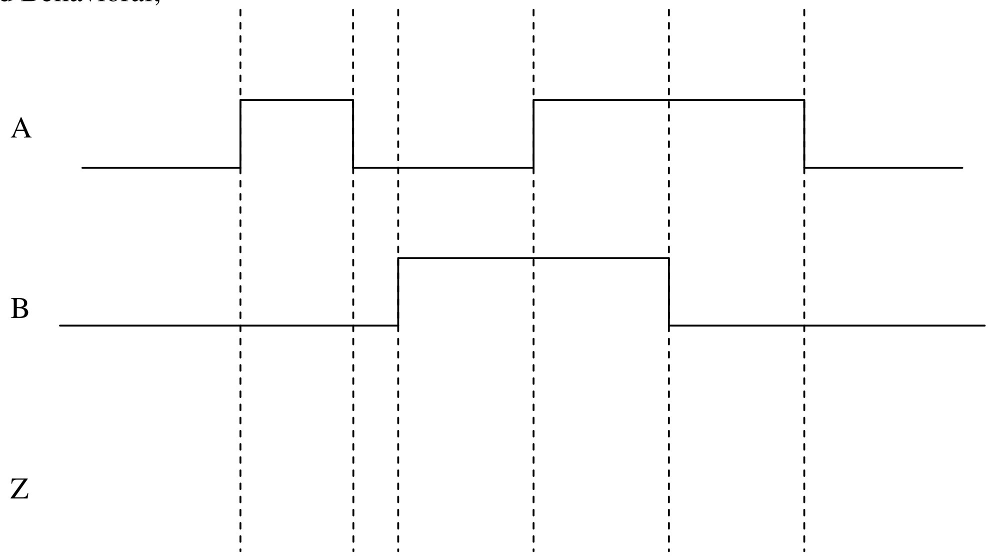
## Timing examples:

### Q7.

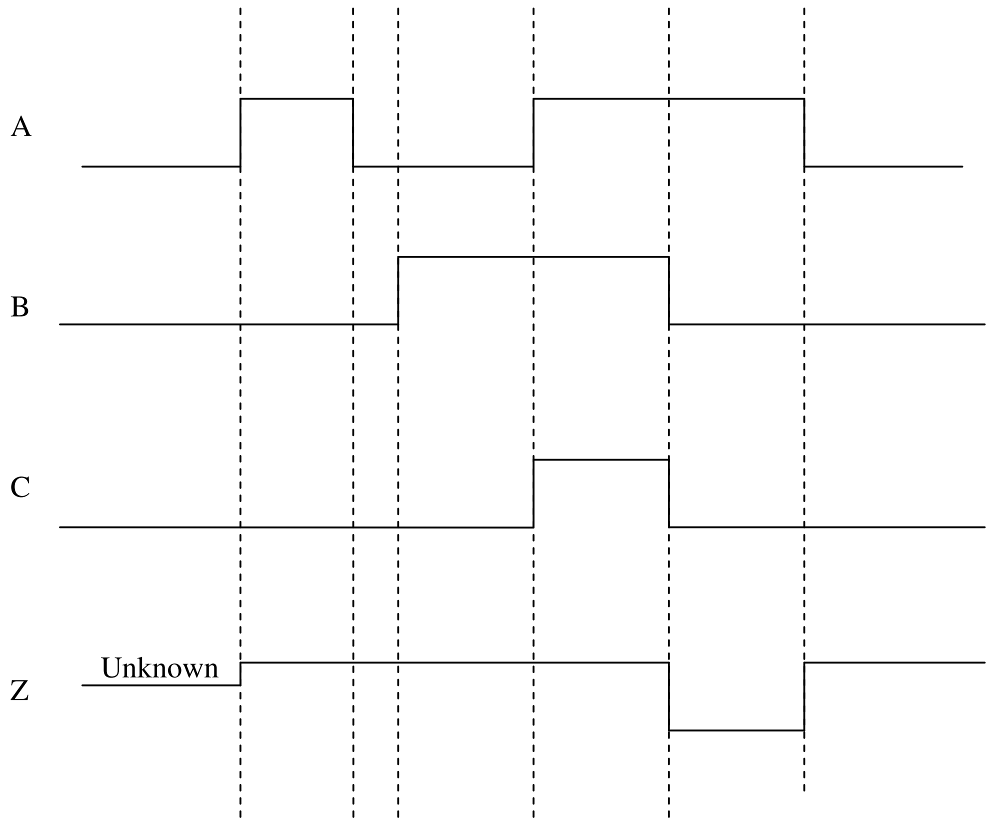
**For the following code draw the simulation waveform of Z for the below given waveforms of A and B.**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

```
entity circuit is
   Port ( A : in  STD_LOGIC;
          B : in  STD_LOGIC;
          Z : out  STD_LOGIC);
end circuit;

architecture Behavioral of circuit is
signal C: std_logic;
begin
        process(A,B)
        begin
                C <= A and B;
                Z <= not C;
        end process;
end Behavioral;
```



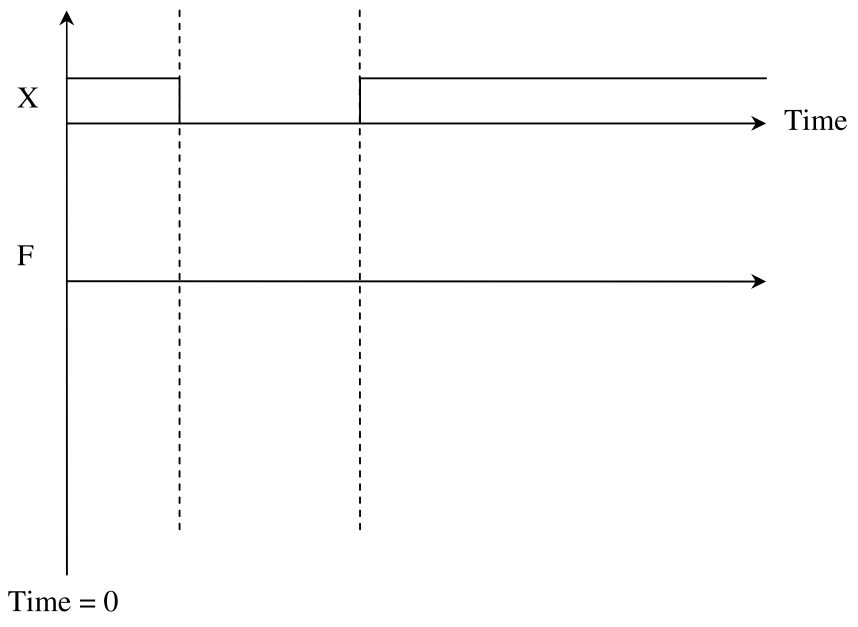
**Solution:**



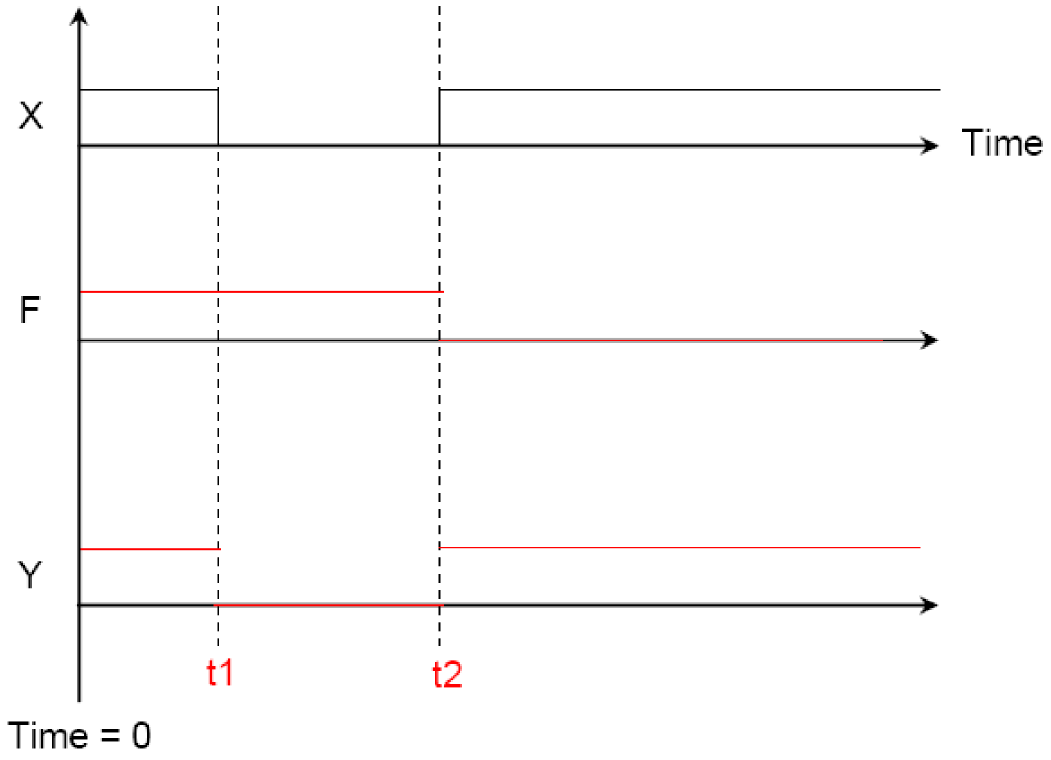
## Q8.

**Draw the timing diagram for the circuit described by the following VHDL code for the input signal waveform given below.**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity simple is
        port(X:in std_logic;
                F:out std_logic);
end simple;

architecture Behavioral of simple is
signal Y:std_logic :='1';

begin
process(X)
begin
        Y<=X;
        F<=Y;
end process;

end Behavioral;
```



**Solution:**



**Note that Y is not in the sensitivity list of the process. The process will be executed once at t = 0, and then whenever the value of X changes, namely at t = 1 and t = t2.**

**Remember that the new values of the assigned signals become effective after the process call terminates. For this reason, the statement F <= Y will store the old value of Y to F each time it executes. At t = 0, the old value of Y is 1, due to the initialization statement. At t = t1, the old value of Y is 1. At t = t2, the old value of Y is 0.**

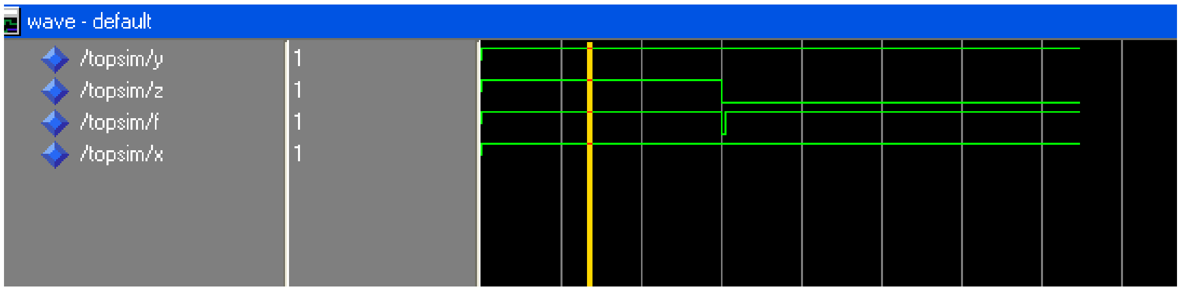
## Q9.

**a) Consider the circuit in Figure 4.38 of Wakerly 4th edition. This circuit has static-1 hazard. Write VHDL code for this circuit ( not using the schematic editor). Write code such that the inverter has 5 ns delay (both H to L, and L to H) but the other gates do not have delays. Make timing simulation to illustrate the static-1 hazard.**
**b) Alter your circuit to remove static-1 hazard, like in Figure 4.41. Repeat the same simulation to show that the static-1 hazard disappears.**
**For both parts a) and b) copy-paste your code and simulation waveforms.**

**Solution:**
**a)**

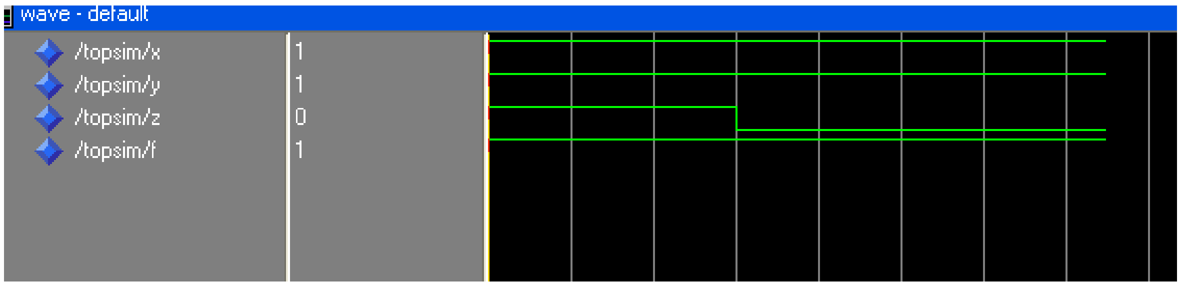
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

```
entity top is
    Port ( X : in std_logic;
           Y : in std_logic;
           Z : in std_logic;
           F : out std_logic);
end top;

architecture Behavioral of top is
 signal ZP,XZP,YZ:std_logic;
begin
 ZP<=not z after 5 ns;
 XZP<=X and ZP;
 YZ<=Y and Z;
 F<=XZP or YZ;

end Behavioral;
```



**b)**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity top is
    Port ( X : in std_logic;
           Y : in std_logic;
           Z : in std_logic;
           F : out std_logic);
end top;

architecture Behavioral of top is
 signal ZP,XZP,YZ,XY:std_logic;
begin
 ZP<=not z after 5 ns;
 XZP<=X and ZP;
 YZ<=Y and Z;
 F<=XZP or YZ or XY;
 XY<=X and Y;
end Behavioral;
```



```
/topsim/x   1
/topsim/y   1
/topsim/z   0
/topsim/f   1
```