Name: Meeyad Mohd. Shabab
Studentnr# 485066

NTNU
Department of electronics and Telecommunications
2016-08-28

TFE4141 Design of digital systems 1

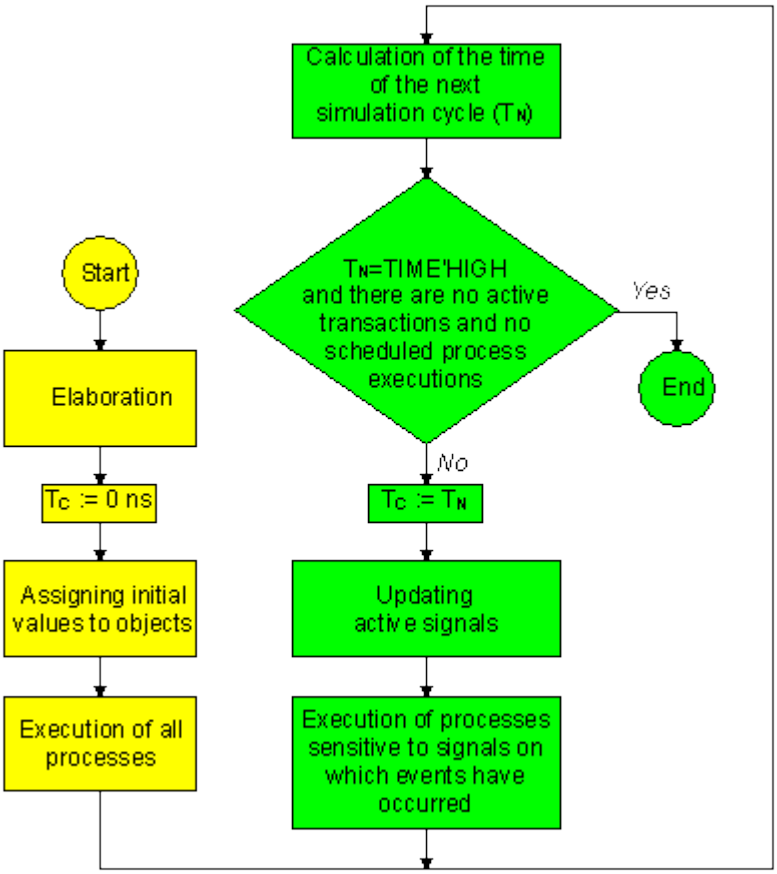
Assignment 2: Understanding the Delta-delay

Task 1

1. **Read the paper: Time and delta-delay in VHDL (found under Lectures/VHDL on it's learning).**
2. **Explain in your own words what happens in a simulation cycle. If you want you may illustrate your explanation with a flow diagram. Keep your answer short.**

Answer:

To help understand the concept of event-driven simulation, the algorithm of the VHDL simulation cycle is presented below:



The moment we choose the **Initialize Simulation** command, the following operations are performed:

- The simulation model is built in the memory (elaboration).
- The current simulation time, **T**C, is set to 0 ns.

- Each object in the model (a signal, variable or constant) acquires its initial value.
- Each process in the model is executed until it suspends.

If the initialization is successful, the time of the next simulation cycle, $\mathbf{T_N}$, is determined according to the rules discussed at the end of the simulation cycle description. The operations making up the actual loop of the simulation cycle are shown in the figure as green-colored blocks.

The simulation cycle comprises the following phases:

1. The current simulation time $\mathbf{T_C}$ is set equal to $\mathbf{T_N}$.
2. All active signals are updated. If a signal changes its value, an *event* is said to occur on it.
3. Processes which meet one of the following conditions are executed until they suspend:
    - A process is sensitive to a signal on which an event has occurred.
    - A process has been scheduled to execute at the current simulation time (applies to processes with **wait** statements with timeout clauses).

The time of the next simulation cycle, $\mathbf{T_N}$, is determined by setting it to the earliest of:

- TIME'HIGH
- The next time at which a driver becomes active.
- The next time at which the execution of a process is scheduled. A process may resume execution without any events occurring on signals the process is sensitive to if it contains wait statements with timeout clauses.

Simulation is completed when $\mathbf{T_N}$ = TIME'HIGH and no transactions nor process executions are scheduled.

If $\mathbf{T_N} = \mathbf{T_C}$, then the next simulation cycle (if any) will be a *delta cycle*.

3. Explain what happens to signals that are assigned values with and without explicit delays.

Answer:

When signals are assigned without explicit delays, the value of the signal does not immediately change! Instead, the assigned value is remembered and will be committed as the actual signal value later (in preparation for the next delta cycle, which is effectively the next quantum of time). Since the next delta cycle will not begin until all processes from the previous delta cycle have completed, signal values will only change when no process is running. Once all signals have changed, the next delta cycle begins and any process sensitive to one of the changed signals will be executed.

When the signals are assigned with explicit delays, the assignments to the output signals occur instantaneously. If those assignment are based on input signals then the input signals are sampled at the instant the process runs. One thing to know is the process commands are executed sequentially, from top to bottom. This is important since a signal could be given multiple assignments within the same process. When that happens the last assignment "wins" and becomes the signal assignment that occurs during that execution of the process.

Task 2

Given the following two concurrent (samtidige) processes in an architecture:

```
Q <= A nor QN;
QN <= B nor Q;
```

Given the following stimuli:

```
A <= '1' ; B <= '0' ;
wait for 10 ns ;
A <= '0' ;
wait for 10 ns ;
B <= '1' ;
wait for 10 ns ;
B <= '0' ;
wait for 10 ns ;
B <= '1' ; A <= '1' ;
```

1. Use the type **std_ulogic** as defined on the next page. Make a timing diagram for A, B, Q, and QN. Make sure that you include all delta-cycles.

Answer:

Time	A	B	Q	QN
0	U	U	U	U
0+Δ	1	0		
0+2Δ			X	1
0+3Δ			0	0
0+4Δ			0	
10ns				
10+Δ	0			
10+2Δ			1	
20ns				
20+Δ		1		
20+2Δ				0
20+3Δ				1
20+4Δ				0
0+nΔ				
30		0		

2. Does the simulation stop after all stimuli have been applied and the two processes Q and QN are suspended, or do you experience oscillations?

Answer:

There are no oscillations for Q and QN after all stimuli have been applied and the two processes Q and QN are suspended. The Signals for A and B is high but there are no output signal/oscillation for Q and QN.

3. Write VHDL-code with entity and architecture that implements the processes Q and QN. Write a testbench which applies the given stimuli to this entity.

Answer:

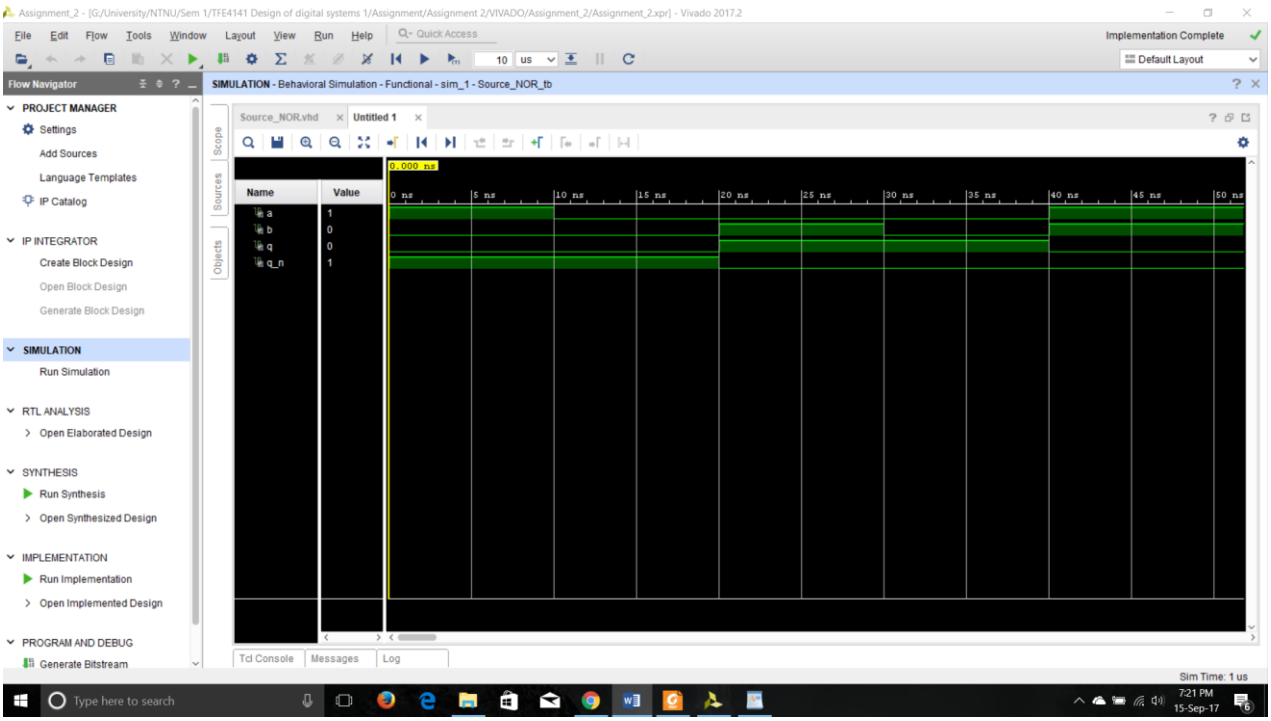
Source Code:

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Source_NOR is
    Port ( a : in STD_LOGIC;
           b : in STD_LOGIC;
           q : inout STD_LOGIC;
           q_n : inout STD_LOGIC);
end Source_NOR;

architecture Behavioral of Source_NOR is

begin

q <= a nor q_n;
q_n <= b nor q;

end Behavioral;
```

Testbench Code:

```
--import std_logic from the IEEE library
library IEEE;
use IEEE.Std_logic_1164.all;
use IEEE.Numeric_Std.all;

--ENTITY DECLARATION: no inputs, no outputs
entity Source_NOR_tb is
end Source_NOR_tb;

-- Describe how to test the NOR Gate
architecture bench of Source_NOR_tb is

--pass norGate entity to the testbench as component
  component Source_NOR
      Port ( a : in STD_LOGIC;
             b : in STD_LOGIC;
             q : inout STD_LOGIC;
             q_n : inout STD_LOGIC);
  end component;

  signal a: STD_LOGIC;
  signal b: STD_LOGIC;
  signal q: STD_LOGIC;
  signal q_n: STD_LOGIC;

begin

--map the testbench signals to the ports of the norGate
  uut: Source_NOR port map ( a   => a,
                             b   => b,
                             q   => q,
                             q_n => q_n );

  stimulus: process
  begin

  a <='1';
  b <='0';
  wait for 10 ns ;
  a <='0';
  wait for 10 ns ;
  b <='1';
  wait for 10 ns ;
  b <='0';
  wait for 10 ns ;
  b <= '1';
  a <= '1';

    wait;
  end process;


end bench;
```

Simulation Result (Oscillation Output):



Use the type std_ulogic which is defined as follows:

```
--
TYPE std_ulogic IS ( 'U',  -- Uninitialized
                     'X',  -- Forcing Unknown
                     '0',   -- Forcing 0
                     '1',   -- Forcing 1
                     'Z',   -- High Impedance
                     'W', -- Weak Unknown
                     'L',   -- Weak 0
                     'H',  -- Weak 1
                     '-',   -- Don't care );
--
```

Note: To make this type available you must include the following in you code:

library *ieee ;*
use *ieee.std_logic_1164.**all** ;*

These code lines must be written above the entity and architecture declaration.
When a simulation starts, a signal not given a specific initialization value, will be assigned the first value in the list, in this case '**U**' – (**U**ninitialized). Later on the signal value will typically change into '**X**', '**1**' or '**0**'.