Part 1:

Objectives of the Lab/Program. (1 point)

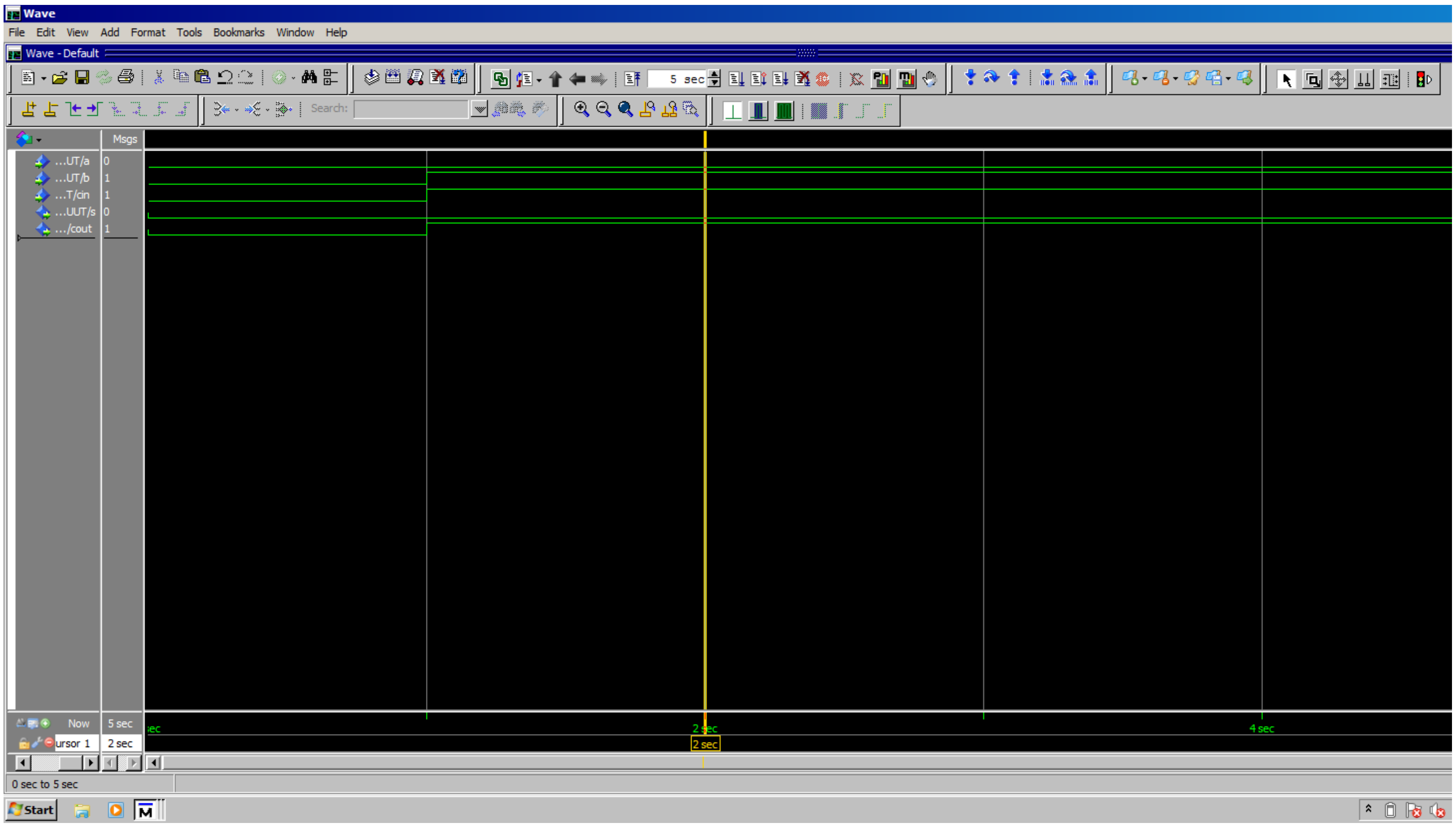
The objectives of the first laboratory assignment is to design, implement, test, and simulate a behavioral model for a full adder using a standardized Hardware Description Language.

Verilog/VHDL Source Codes (4 points)

```
--Lab 5:
--Write a VHDL/Verilog program and simulate the following modules:
--a) Full adder (behavioral)
--Due: 7/17 8:00 AM; lab report by 23:59 7/17
-------------------------------------------------------------------------------
--Author: Peter Dranishnikov    --Partner: NOT APPLICABLE
--"Veryhighhardwaredl"

library ieee;
use ieee.std_logic_1164.all;

entity fulladder is
    port(a, b, cin : in STD_LOGIC; s, cout : out STD_LOGIC);
end entity fulladder;

architecture behavioral of fulladder is
begin
    s <= a xor b xor cin;
    cout <= (a and b) or (a and cin) or (b and cin);
end architecture behavioral;

--TESTBENCH--TESTBENCH--TESTBENCH--TESTBENCH--TESTBENCH--TESTBENCH--

library ieee;
use ieee.std_logic_1164.all;

entity t_fulladder is
end entity t_fulladder;

architecture behavioral of t_fulladder is
    signal t_a : std_logic := '0';
    signal t_b : std_logic := '0';
    signal t_cin : std_logic := '0';
    signal t_s : std_logic;
    signal t_cout : std_logic;
    component fulladder is
        port(a, b, cin : in STD_LOGIC; s, cout : out STD_LOGIC);
    end component fulladder;
begin
    UUT : fulladder port map
```

```
        (
            a => t_a,
            b => t_b,
            cin => t_cin,
            s => t_s,
            cout => t_cout
        );

        process begin
            wait for 1 sec;
            t_a <= '0';
            t_b <= '1';
            t_cin <= '1';
            wait;
        end process;
end architecture behavioral;
```

Screen shot of the simulation (waveforms). (4 points)



Note: Image is cropped to fit page.

Conclusion and References. (1 point)

The example applied for testing is setting the "b" and "cin" inputs for the full adder to std_logic '1', and the rest of the inputs to '0', respectively. The simulation works as expected: the sum is '0' and the carryout is '1'. Therefore, it is possible to model combinational digital logic using a standardized HDL.

Dranishnikov, Peter A. "Experiment 5: Design and Implementation of Half Adder and Full Adder Circuit(s)"
Mano, M. Morris. & Ciletti, Michael D. *Digital Design* 6th edition
Harris, David Money & Harris, Sarah L. *Digital Design and Computer Architecture* 2nd edition

Part 2:

Objective of the Lab/Program. (1 point)

The objectives of the second part of the first laboratory assignment is the structural definition and implementation of a 4-bit PIPO register with an enable line in a standardized Hardware Description Language.

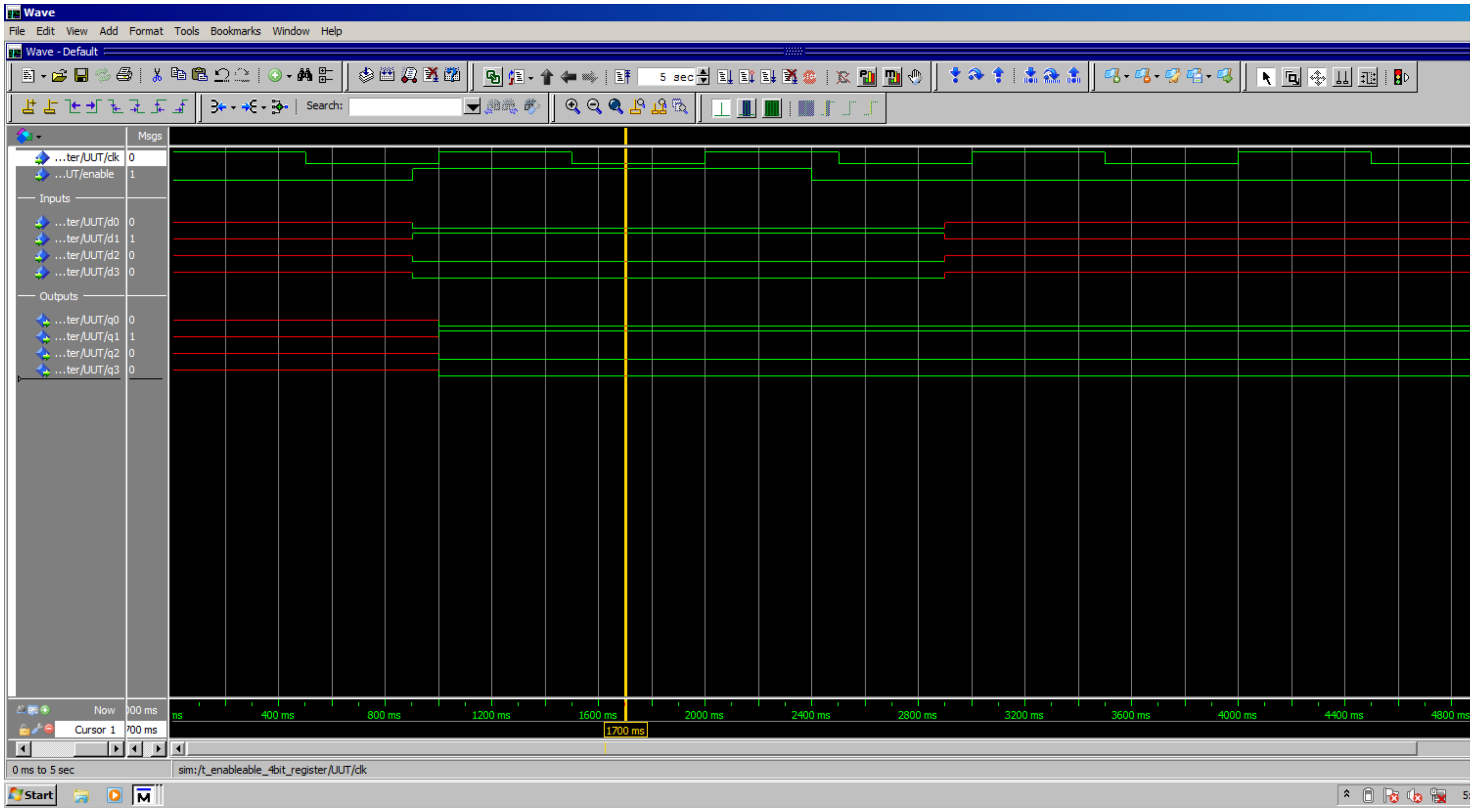
Verilog/VHDL Source Codes (4 points)

```
--Lab 5:
--Write a VHDL/Verilog program and simulate the following modules:
--b) 4 bit register with enable (structural)
--Due: 7/17 08:00; lab report by 23:59 7/17
------------------------------------------------------------------------------
--Author: Peter Dranishnikov    --Partner: NOT APPLICABLE
--"Veryhighhardwaredl"

library ieee;
use ieee.std_logic_1164.all;
entity d_ff is
    port(d, clk : in std_logic; q : out std_logic);
end entity d_ff;
architecture behavioral of d_ff is
begin
    d_flip_flop: process(clk) begin
        if(rising_edge(clk)) then
            q <= d;
        end if;
    end process;
end architecture behavioral;

library ieee;
use ieee.std_logic_1164.all;
entity and_gate is
```

```
    port(a, b : in std_logic; c : out std_logic);
end entity and_gate;
architecture behavioral of and_gate is
begin
    enable_line: c <= a and b;
end architecture behavioral;

library ieee;
use ieee.std_logic_1164.all;
entity enableable_4bit_register is
    port(clk, enable, d0, d1, d2, d3 : in std_logic; q0, q1, q2, q3 :
out std_logic);
end entity enableable_4bit_register;

architecture structural of enableable_4bit_register is
    component d_ff is
        port(d, clk : in std_logic; q : out std_logic);
    end component d_ff;

    component and_gate is
        port(a, b : in std_logic; c : out std_logic);
    end component and_gate;
    signal m_clk : std_logic;
begin

    clk_enable: and_gate port map
        (
            a => clk,
            b => enable,
            c => m_clk
        );

    DR0: d_ff port map
        (
            d => d0,
            clk => m_clk,
            q => q0
        );

    DR1: d_ff port map
        (
            d => d1,
            clk => m_clk,
            q => q1
        );

    DR2: d_ff port map
        (
            d => d2,
            clk => m_clk,
            q => q2
        );
    DR3: d_ff port map
        (
            d => d3,
            clk => m_clk,
            q => q3
        );
end architecture structural;

--TESTBENCH--TESTBENCH--TESTBENCH--TESTBENCH--TESTBENCH--TESTBENCH--

library ieee;
use ieee.std_logic_1164.all;

entity t_enableable_4bit_register is
end entity t_enableable_4bit_register;

architecture behavioral of t_enableable_4bit_register is
    constant CLKCYCLE : time := 1 sec;
    signal t_clk : std_logic := '1';
    signal t_enable : std_logic := '0';
    signal t_d0, t_d1, t_d2, t_d3 : std_logic;
    signal t_q0, t_q1, t_q2, t_q3 : std_logic;

    component enableable_4bit_register is
        port(clk, enable, d0, d1, d2, d3 : in std_logic; q0, q1, q2,
q3 : out std_logic);
    end component enableable_4bit_register;
begin
    UUT : enableable_4bit_register port map
    (
        clk => t_clk,
        enable => t_enable,
        d0 => t_d0,
        d1 => t_d1,
        d2 => t_d2,
        d3 => t_d3,
        q0 => t_q0,
        q1 => t_q1,
        q2 => t_q2,
        q3 => t_q3
    );

    CLK_GEN : process is
    begin
        wait for CLKCYCLE / 2;
        t_clk <= not t_clk;
    end process CLK_GEN;

    process begin
        wait for 900 ms;
        t_enable <= '1';
        t_d0 <= '0';
        t_d1 <= '1';
```

```
                t_d2 <= '0';
                t_d3 <= '0';
                wait for 1500 ms;
                t_enable <= '0';
                wait for 500 ms;
                t_d0 <= 'X';
                t_d1 <= 'X';
                t_d2 <= 'X';
                t_d3 <= 'X';
                wait;
            end process;
        end architecture behavioral;
```

Screen shot of the simulation (waveforms). (4 points)



Note: image is cropped to fit page.

Conclusion and References. (1 point)

To test the structural implementation of the 4-bit PIPO register, the enable line was set to std_logic ‘1’ and the input of binary 0100 was written in parallel (d0 being the MSB). The enable line was then set to ‘0’ and the input set to ‘X’ to verify that the register can hold data when not writing, i.e. readable. The register functioned as intended, therefore it is possible to design and simulate a sequential circuit using a standardized HDL.

Dranishnikov, Peter A. "Experiment 9: Design and Implement a Synchronous BCD Counter with J-K flip-flops"

Mano, M. Morris. & Ciletti, Michael D. *Digital Design* 6th edition

Harris, David Money & Harris, Sarah L. *Digital Design and Computer Architecture* 2nd edition