# Department of Electrical & Computer Engineering
# Dalhousie University

## ECED 4260
## IC Design and Fabrication
## Midterm Examination

| | |
|---|---|
| Instructor: | Y. Ma |
| Exam date: | Oct. 26, 2006 |
| Exam duration: | 80 minutes |
| Aids permitted: | This is a closed book exam. No books or notes are to be consulted. Two double-sided 8.5″x11″ formula sheets are allowed. |

Instructions:
1. Provide your name printed, signature and I.D. number on this page.
2. Verify that this booklet contains 8 pages (including the cover page).
3. Neatly enter your answers in the spaces provided.
4. Use the reverse sides of the pages for rough work.

Student name: _____

Signature: _____

Student ID: _____

Reference Solution

| Question | Time (min) | Worth | Mark | Subject |
|---|---|---|---|---|
| 1 | 15 | 20 | | Boolean algebra |
| 2 | 10 | 20 | | Combinational logic design |
| 3 | 20 | 20 | | VHDL |
| 4 | 20 | 20 | | Logic Simulation |
| 5 | 15 | 20 | | Finite State Machine |
| Total | 80 min | 100 | | |

Question 1. (20 marks)

1.1 (15 marks) Sometimes the XOR gates in a logic design need to be converted into their corresponding NAND implementation. For example, an ECED 4260 student and his/her lab partner may want to reduce the types of gates used in their design in order to reduce the number of chips in their lab circuit implementation. In practice, testing engineers may do the conversion to improve the testability of digital circuits. The Boolean expression given below describes the XOR function of variables a and b. Use Boolean algebra to manipulate the expression, so that *f* contains 2-input NAND logic only. No inversion or any other logic operations are allowed.

$f = a'b + ab'$

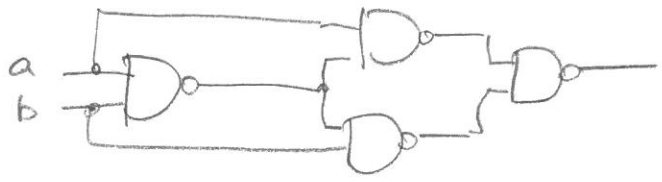
$= \bar{a}b + a\bar{b} + a\bar{a} + b\bar{b}$

$= a(\bar{a}+\bar{b}) + b(\bar{a}+\bar{b})$

$= \overline{\overline{a \cdot \overline{ab}} + b \cdot \overline{ab}}$

$= \overline{\overline{a \cdot \overline{ab}} \cdot \overline{b \cdot \overline{ab}}}$

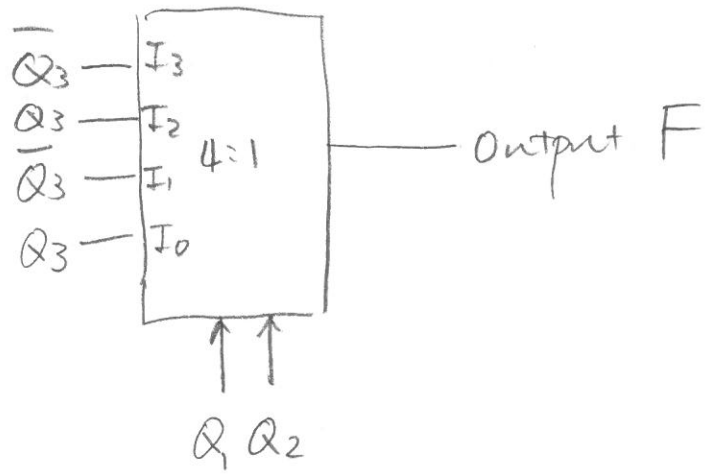
1.2 (5 marks) Draw the schematic diagram corresponding to your new *f* function.



Question 2. (20 marks)

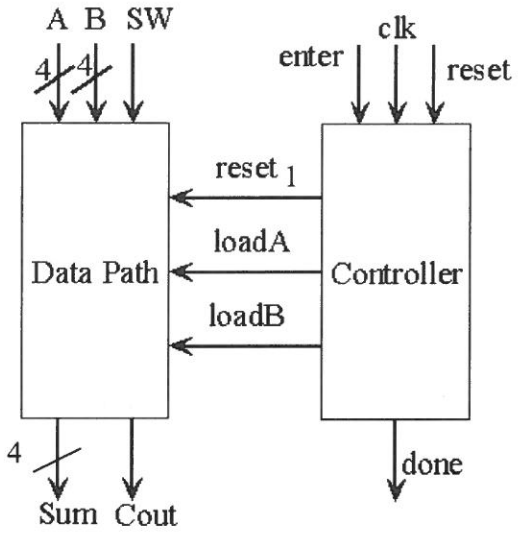
2. Multiplexers are important constructs for RTL circuits. If you only have a 4-input 1-bit multiplexer available, can you implement the function, F, defined by the truth table given below? Draw the Mux and indicate the inputs.

| Q3 \ Q1Q2 | | 00 | 01 | 11 | 10 |
|---|---|---|---|---|---|
| | 0 | 0 | 1 | 1 | 0 |
| | 1 | 1 | 0 | 0 | 1 |



Question 3. (20 marks)

3. The figure below shows the final design of the system. Assume that the architecture of data path and controller are available; write out the entities of Data Path and Controller and your final design entity and architecture.



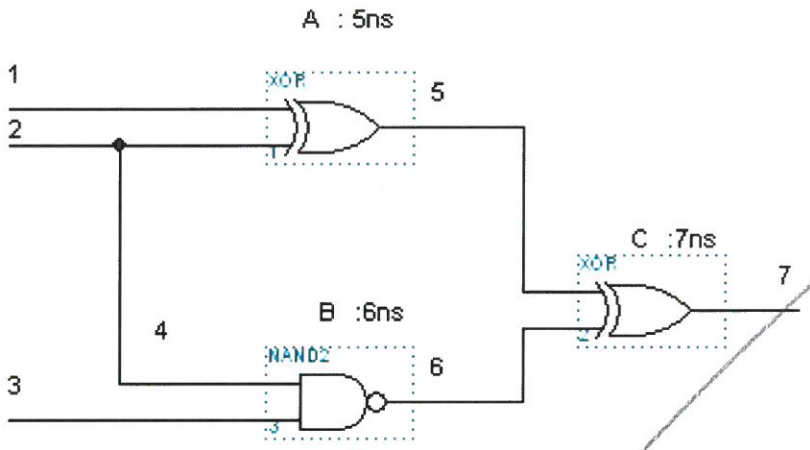

```
library ieee;
use ieee.std_logic_1164.all;

entity final-design is
    port( A, B: in std_logic_vector(3 downto 0);
          sw, enter, clk, reset : in std_logic;
          sum : out std_logic_vector(3 downto 0);
          cout, done : out std_logic);
end final-design;

Architecture structural of final_design is
  component Data_Path
      port ( A, B : in std_logic_vector(3 downto 0);
             sw, reset1, loadA, loadB: in std_logic;
             sum : out std_logic_vector(3 downto 0);
             cout : out std_logic);
  end component;
```

Question 3 (cont'd)

```
component Controller
    port (enter, clk, reset: in std_logic;
          reset1, loadA, loadB, done: out std_logic);
end component;

signal reset1, loadA, loadB: std_logic;

Begin
    myDatapath: component Data-Path
        port map( A, B, SW, reset1, loadA, loadB, sum, cout);

    myController: component Controller
        port map (enter, clk, reset, reset1, loadA, loadB, done);

End structural;
```

Question 4. (20 marks)

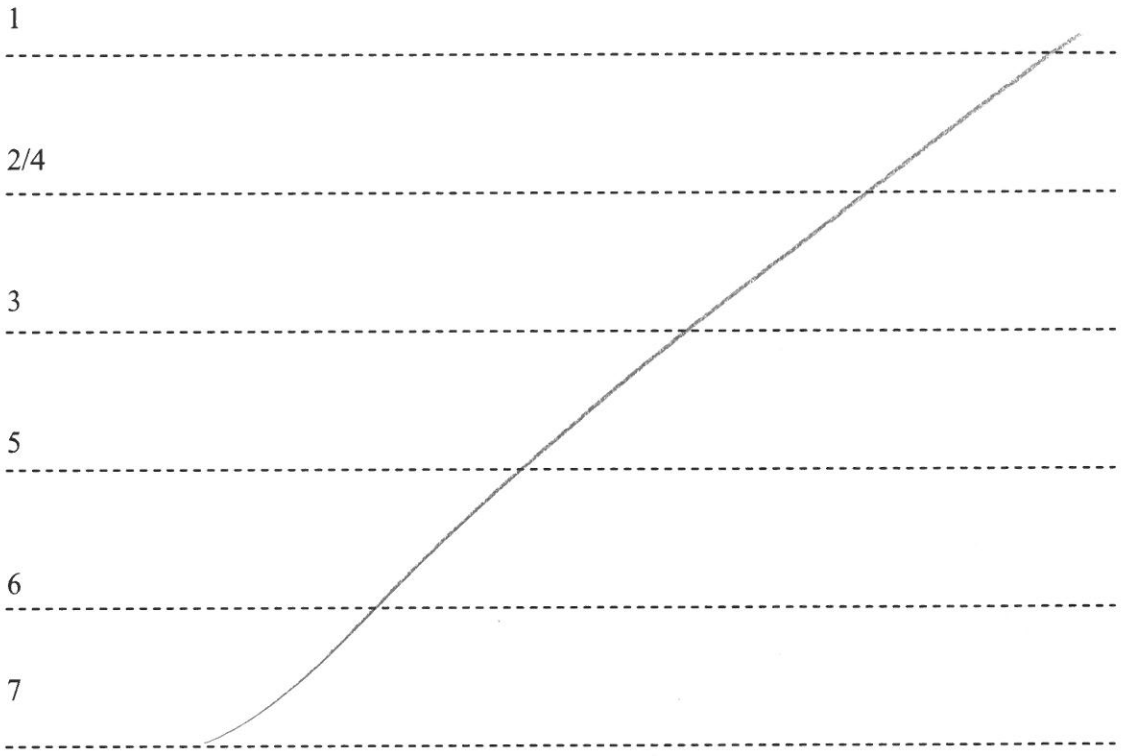
4. The circuit given below has been initialized by input x1x2x3=010 and all the circuit nodes are in known stable states. Perform the event-driven simulation of the circuit for two input vectors x1x2x3=000 at t=5 and x1x2x3=011 at t=20, following the initialization.




4.1 Complete the simulation table below (15 marks)

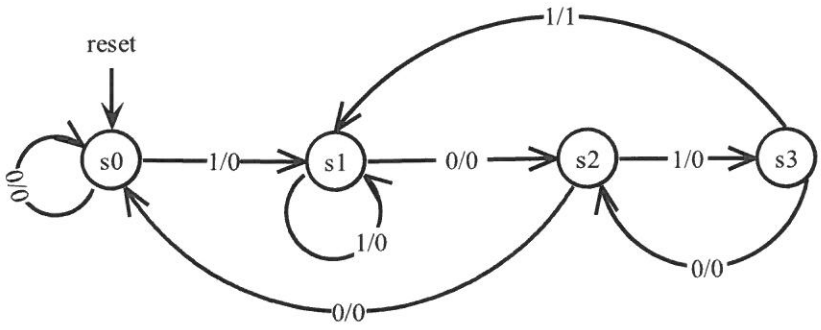
| | Line values | | | | | | | Gates affected | | | Scheduled events |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Time | 1 | 2 | 3 | 4 | 5 | 6 | 7 | A | B | C | |
| 0ns | 0 | 1 | 0 | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |

4.2 Derive the wave-form timing diagram of all the circuits nodes. (5 marks)




Question 5. (20 marks)

5.1 A finite state machine has one input x and one output z. The state transition diagram is shown below. Explain what the machine is supposed to do. (6 marks)




Pls. see Sample midterm 3: Q3

5.2 The VHDL code given below describes the above FSM with an extra output showing the binary assignment of the states.
(a) (7 marks) By writing directly in the code, show how to add Mealey output z to the state machine. Label these changes with (a).
(b) (7 marks) By writing directly in the code, indicate how an asynchronous reset can be added that causes transitions to the state s0. Label these changes with (b).

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

entity example is
        port (clk: in std_logic;
                x: in std_logic;
                stateout: out std_logic_vector (1 downto 0));

end entity example;

architecture state_machine of example is
         type statetype is (s0, s1, s2, s3);
        signal present_state, next_state: statetype;
begin
        comb_logic: process (present_state, x) begin
                Case present_state is
                                when s0 => stateout<="00";
                                        If (x='1') then next_state<=s1;
                                        else next_state<=s0;
                                        end if;
                                when s1 => stateout<="01";
                                        If (x='0') then next_state<=s2;
                                        else next_state<=s1;
                                        end if;
                                when s2 => stateout<="10";
                                        If (x='1') then next_state<=s3;
                                        else next_state<=s0;
                                        end if;
                                when s3 => stateout<="11";
                                        If (x='1') then next_state<=s1;
                                        else next_state<=s2;
                                        end if;
                end case;
        end process comb_logic;

        memory_element: process (clk ) begin

                                        If (clk'event and clk='1') then
                                        present_state <= next_state;
                                        end if;
        end process memory_element;

end architecture state_machine;
```