University of Toronto

Department of Electrical and Computer Engineering

Faculty of Applied Science and Engineering

Final Examination - December 8, 1998

Second Year - Programs 7 and 9

ECE241 - Digital Systems

Examiners: S.D. Brown and J.S. Rose

EXAM Type: D (no aids)

Last Name: ____________________

First Name: ____________________

Student Number: ____________________

Duration: **2.5 Hours**

You should answer ALL questions.
All answers should be on these sheets.
No aids permitted.
See Instructions on page 2.

EXAMINER's REPORT

1	______	/25
2	______	/20
3	______	/15
4	______	/15
5	______	/15
6	______	/10
TOTAL:	______	/100

Instructions

- READ ALL QUESTIONS CAREFULLY BEFORE ANSWERING.
- Attempt all questions.
- If you need to make any assumptions, state them clearly with your answers.
- Write your answers neatly. Messy work is very hard to read and may cause you to lose marks.
- For questions that specify minterms using the notation in the example below

$$f(x_1, x_2, x_3) = \sum m(1, 3, 5)$$

  the minterms are to be interpreted as used in class. Specifically, minterm m_1 represents $x_1x_2x_3 = 001$, minterm m_2 represents $x_1x_2x_3 = 010$, and so on. Answers that fail to interpret the minterms in this way will be considered incorrect.

Question 1 — Short Answers **[25 marks]**

a. Use algebraic manipulation to derive the minimal sum-of-products form for the following expression. Be sure to show all of your steps.

i. $f = \bar{x}_1 x_2 + x_2 x_4 + x_1 x_3 \bar{x}_4 + x_2 x_3$

b. The function $f(x_1, x_2, x_3) = \sum m(1, 4, 5, 7)$ can be implemented using **one** 2-to-1 multiplexer, **one** 2-input AND gate, and **one** 2-input OR gate (be sure to see Page 2 for interpretation of minterms). Design the circuit using exactly *and only* these gates.

c. Design a 3-bit ripple counter using D-type flip-flops and any other needed gates. Show the schematic diagram of your circuit.

d. Design a 1-bit full adder using 2-input NAND gates only. Show the schematic diagram of your circuit.

e. For each of the following n-bit binary numbers, show the value, **in base 10**, if the number is considered *unsigned* or *signed* (2's complement).

i. 110110

unsigned value: ________________________

signed value: ________________________

ii. 1000

unsigned value: ________________________

signed value: _______________________

f. Perform the following operations assuming that signed numbers are used. Indicate whether or not overflow occurs.

i.
```
   1000
 + 0111
 ---------
```
Overflow (yes, no): _________________

ii.
```
   1000
 + 1111
 ----------
```
Overflow (yes, no): _________________

iii.
```
   0111
 + 1111
 ----------
```
Overflow (yes, no): _________________

iv.
```
   0101
 + 0011
 ----------
```
Overflow (yes, no): _________________

Question 2 — Word problem/VHDL code **[20 marks]**

Below is given VHDL code for a simple 4-bit up-counter. This code is shown for your reference only.

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.std_logic_unsigned.all ;

ENTITY upcount IS
    PORT (      Clock       : IN            STD_LOGIC ;
                Q           : BUFFER        STD_LOGIC_VECTOR(1 DOWNTO 0) ) ;
END upcount ;

ARCHITECTURE Behavior OF upcount IS
BEGIN
    PROCESS ( Clock )
    BEGIN
        IF (Clock’EVENT AND Clock = ‘1’) THEN
            Q <= Q + ‘1’ ;
        END IF ;
    END PROCESS;

END Behavior ;
```

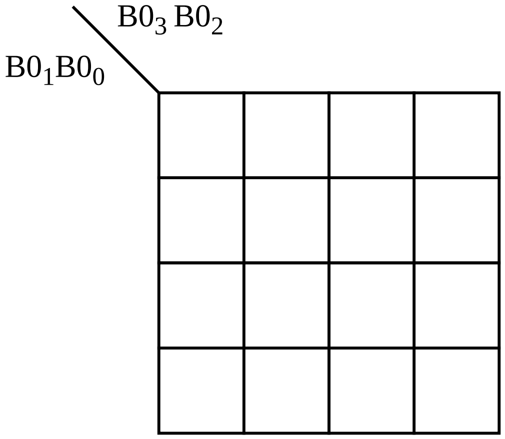
Consider a 2-digit binary-coded decimal (BCD) counter. Each (decimal) digit in this counter consists of four bits. Let the most-significant digit be called *B1* and the least-significant digit be called *B0*. Each digit can have the values from 0000 to 1001, corresponding to the decimal values 0 to 9. Thus, the counter outputs, *B1 B0*, can be viewed as the decimal numbers from 00 to 99.

a. Write complete VHDL code for the BCD counter. Your code must describe the required counting sequence. Assume that there is a *Clock* input, and a *Reset* input. When *Reset* = 1 the counter should be set to 00 asynchronous of the *Clock* input. Put your answer in the space provided on the next page.

ANSWER for part a)

b. For this part, assume that the four bits in *B0* are used as inputs to another circuit that produces the output *g*. This output should be 1 if *B0* is odd; otherwise *g* should be 0. Fill in the K-map given below and derive a minimal **product-of-sums** expression for *g*.

K-map for *g*:



Minimum-cost product-of-sums expression for *g:*

c. For this part, assume that both B1 and B0 are used as the inputs to another circuit that produces the output *h*. This output should be 1 if the two-digit decimal number represented by B1 B0 is between 0 and 90 and is an even multiple of 9. In your design you can use any type of logic gates needed, and you can also use standard subcircuits that you have seen in class, such as full-adder, n-bit ripple-carry adder, flip-flops, registers, counters, and so on. Higher marks will be given for simple circuits that work.

Put your circuit in the space on the next page.

ANSWER to part c)

Question 3 — Finite state machines [15 marks]

The following is a moore model of a finite state machine, described in VHDL:

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY fsm IS
    PORT (          Clock   : IN            STD_LOGIC ;
                    w       : IN            STD_LOGIC ;
                    z       : OUT           STD_LOGIC ) ;
END fsm ;

ARCHITECTURE fsm_arch OF fsm IS
    TYPE state_names IS (SA,SB,SC,SD,SE,SF) ;
    SIGNAL state : state_names ;
BEGIN
    PROCESS ( Clock )
    BEGIN
        IF ( Clock’EVENT AND Clock= ‘1’ ) THEN
            CASE state IS
                WHEN SA =>
                                    IF w = ‘0’ THEN state <= SB ;
                                    ELSE state <= SA END IF ;
                WHEN SB =>
                                    IF w = ‘0’ THEN state <= SB ;
                                    ELSE state <= SC END IF ;
                WHEN SC =>
                                    IF w = ‘0’ THEN state <= SD ;
                                    ELSE state <= SA END IF ;
                WHEN SD =>
                                    IF w = ‘0’ THEN state <= SB ;
                                    ELSE state <= SE END IF ;
                WHEN SE =>
                                    IF w = ‘0’ THEN state <= SF ;
                                    ELSE state <= SA END IF ;
                WHEN SF =>
                                    IF w = ‘0’ THEN state <= SB ;
                                    ELSE state <= SE END IF ;
            END CASE ;
        END IF ;
    END PROCESS ;

    z <= ‘1’ WHEN state = SF ELSE ‘0’ ;

END fsm_arch ;
```

a. Draw the corresponding state diagram for this state machine.

State diagram:

b. What does this finite state machine do?

ANSWER: ____

____

____

____

____

c. Indicate how the VHDL code can be revised to add a synchronous active-low reset signal, *Resetn*, that causes the state machine to enter state *SA*. Shown only the parts of the code that need to be changed.

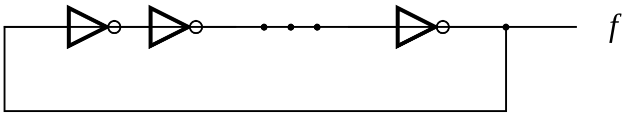
d. Draw the state table that corresponds to the VHDL code given above. Derive minimum sum-of-products equations for the state variables and the output, z, of this circuit. Assume that the state variables are named *y3*, *y2*, *y1* and that the codes for the states are assigned in numeric sequence. Thus, for *SA* the code used is *y3 y2 y1* = 000, for *SB* the code is *y3 y2 y1* = 001, and so on, and for *SF* the code is *y3 y2 y1* = 101. NOTE: be careful in deriving the state table, because no marks will be given for K-maps that correspond to an incorrect state table!!

ANSWER space is continued on next page

ANSWER space for part d) continued

Question 4 — Sequential circuits [15 marks]

A ring oscillator is a circuit that has an *odd* number, *n*, of inverters connected in a ring-like structure, as shown in the figure below. The output of each inverter is a periodic signal with a certain period.

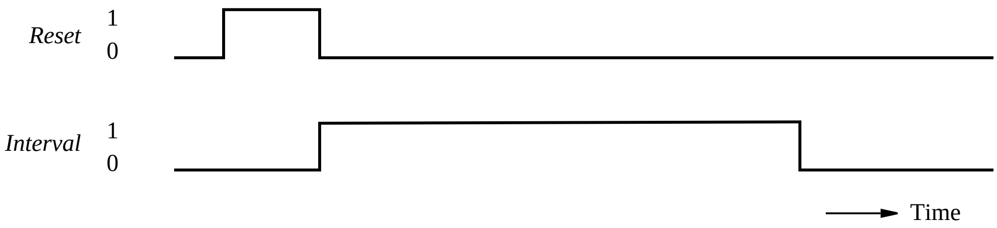


Ring oscillator

a. Assume that all of the inverters are identical, hence they all have the same delay, called t_p. Call the output of one of the inverters *f*. Give an equation that expresses the period of the signal *f* in terms of *n* and t_p.

ANSWER:

b. For this part you are to design a circuit that can be used to experimentally measure the delay, t_p, through one of the inverters in the ring oscillator. Assume that there exists an input called *Reset* and another called *Interval*. The timing of these two signals is shown below:



The length of time for which Interval has the value 1 is known. Assume that this length of time is 100 ns. Design a circuit that uses the *Reset* and *Interval* signals, and the signal *f* from part a) to experimentally measure t_p. In your design you may use logic gates and standard sub-circuits such as adders, flip-flops, counters, registers, and so on. Higher marks will be given for simple circuits that work.

Answer for the circuit for part b)

Describe below in one paragraph how your circuit works:

____

____

____

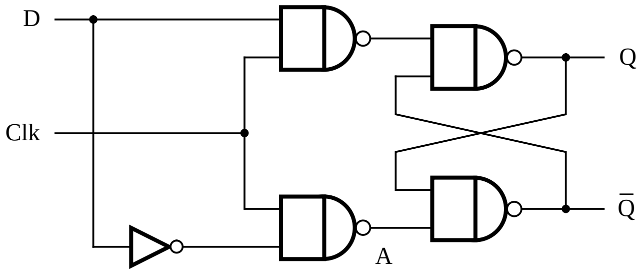
____

____

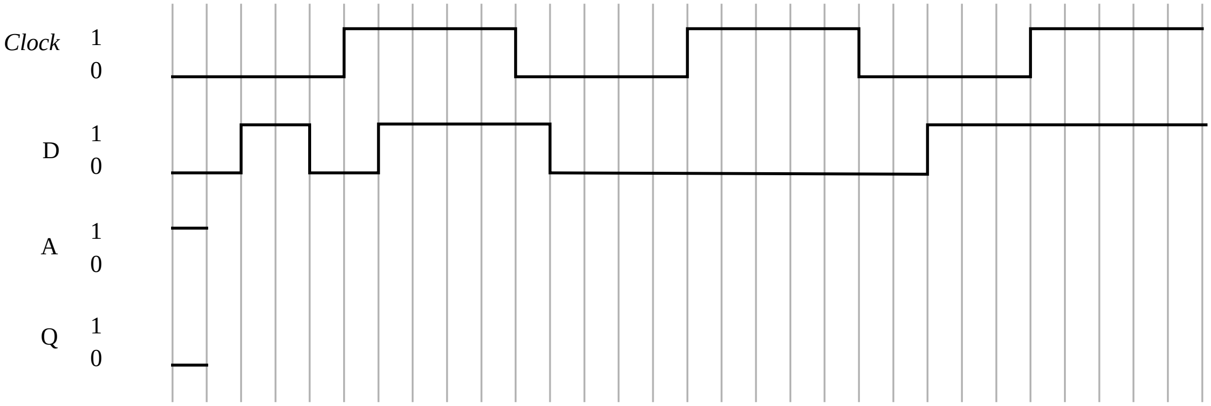
____

Question 5 — Timing **[15 marks]**

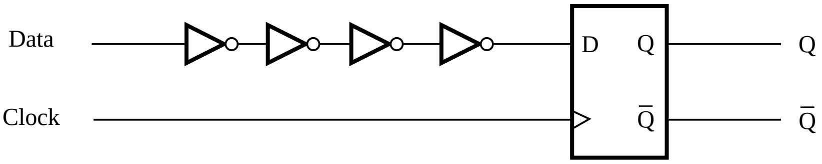
a. A circuit is given below for a gated D latch.



Assume that the propagation delay through a NAND gate is equal to 2 ns and the delay through an inverter is 1 ns. Fill in the timing diagram shown below, which shows the signal values with 1 ns resolution.



b. Consider a D flip-flop for which the setup time is 3 ns, the hold time is 2 ns, and the clock-to-Q time (read time) is 1.5 ns. This flip-flop is used in the circuit below:



In the above circuit, you do not know the exact delay of an inverter. But, you do know that the minimum inverter delay is 0.5 ns, and the maximum delay is 1.5 ns. Using this information, what are the setup, hold, and read times *with respect to the* Data and Clock inputs to the circuit?

ANSWERS:

Setup time: ______________________________

Hold time: _______________________________

Clock-to-Q (read time): _______________________________

c. Consider an 8-bit ripple-carry adder built using eight full-adders. Each full adder is built using a two-level sum-of-products circuit for the carry-out and a 3-input XOR gate for the sum. In this adder assume that the delay through a gate is:

Gate delay = 1 + 0.1 x (#*Inputs*-1) ns,

where #*Inputs* is the number of inputs to the gate. Using this equation, the delay through an inverter is equal to 1 ns, the delay through a 2-input gate (any type) is 1.1 ns, and so on.

You can use the extra pages at the back of the test paper for your rough work.

i. What is the critical path delay in the ripple-carry adder? (Recall: the critical path is the longest (slowest) path in the circuit.) Give your answer in ns.

ANSWER: ________________________

ii. Consider next a carry lookahead adder, as discussed in class. If you can use gates of any size (number of inputs), what is the critical path delay in the 8-bit carry lookahead adder?

ANSWER: ________________________

iii. Now assume that you can use gates with a fanin of no more than four inputs. What is the critical path delay for the carry lookahead adder in this case?

ANSWER: ________________________

Question 6 — Fast adder **[10 marks]**

In class you learned about a hierarchical carry lookahead adder. In this question you are to design a complete 4-bit version of such an adder. Your adder must be built using two 2-bit carry lookahead adders. The carry-in to the most-significant adder block is generated by a *second-level* carry lookahead circuit. Draw the complete hierarchical carry lookahead adder in the space below. Do not show a simple block diagram; draw the logic gates used in the circuit.

EXTRA SPACE — USE ONLY IF NEEDED OR FOR SCRAP PAPER

EXTRA SPACE — USE ONLY IF NEEDED OR FOR SCRAP PAPER

EXTRA SPACE — USE ONLY IF NEEDED OR FOR SCRAP PAPER

EXTRA SPACE — USE ONLY IF NEEDED OR FOR SCRAP PAPER

EXTRA SPACE — USE ONLY IF NEEDED OR FOR SCRAP PAPER