University of Toronto
Faculty of Applied Science and Engineering
Edward S. Rogers Sr. Department of Electrical and Computer Engineering

Final Examination
ECE 241F - Digital Systems

Examiners: S. Brown, J. Rose, and B. Wang
December 2007

Duration: 2.5 Hours

ANSWER ALL QUESTIONS. USE ONLY THESE SHEETS, USING THE BACKS IF NECESSARY.

- Exam Type D, these specific aids allowed:
  i. Original Versions (no photocopies) of the course text, **Fundamentals of Digital Logic with Verilog Design (1st or 2nd edition),** by Brown & Vranesic, ISBN 0-07-282315-1.
  ii. One 8.5 x 11" two-sided aid sheet.

- The amount of marks available for each question is given in square brackets [].

LAST NAME: _____

FIRST NAME: _____

STUDENT NUMBER: _____

Question	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Total**
Maximum Mark	**12**	**10**	**10**	**12**	**8**	**8**	**8**	**8**	**10**	**86**
Mark Obtained										

[12] **Q1.** For this question you are to use algebraic manipulation to prove some Boolean relations using the identities in the textbook. The identities are in Chapter 2 of the book and are labeled from 10*a* to17*b*. Note that identity 17, called *consensus*, is listed in the second edition of the textbook, but is not shown in the first edition. It is:

$$xy + yz + \bar{x}z = xy + \bar{x}z \qquad 17a$$
$$(x + y)(y + z)(\bar{x} + z) = (x + y)(\bar{x} + z) \qquad 17b$$

An example problem and solution is given below. Note that

Example:

(e.g.) The following Boolean relation can be proved using algebraic manipulation in two steps, using one identity in each step. Specify which identity is used in each of your two steps.

Relation: $(x + xy)z + x\bar{z} = x$

Proof:	$(x + xy)z + x\bar{z}$	**Identity Used**
	$= (x)z + x\bar{z}$	(13*a*)
	$= x$	(14*a*)

ANSWER PARTS (i) to (v) below.

(i) The following Boolean relation can be proved using algebraic manipulation in one step, using exactly one identity. Specify which identity can be used.

Relation: $(\overline{(w \oplus x)} + \bar{y}) \cdot (\overline{(w \oplus x)} + z) = \overline{(w \oplus x)} + \bar{y}z$

Proof:	$(\overline{(w \oplus x)} + \bar{y}) \cdot (\overline{(w \oplus x)} + z)$	**Identity Used**
	=	( )

(ii) The following Boolean relation can be proved using algebraic manipulation in two steps, using one identity in each step. Specify which identity is used in each of your two steps.

Relation: $xz + yz + x + y = x + y$

Proof:	$xz + yz + x + y$	**Identity Used**
	=	( )
	=	( )

(iii) The following Boolean relation can be proved using algebraic manipulation in two steps, using one identity in each step. Specify which identity is used in each of your two steps.

Relation: $\overline{x(y+z)} = \overline{x} + \overline{y} \cdot \overline{z}$

Proof: $\overline{x(y+z)}$	**Identity Used**
=	( )
=	( )

(iv) The following Boolean relation can be proved using algebraic manipulation in three steps, using one identity in each step. Specify which identity is used in each of your three steps.

Relation: $wy + xy + yz + (\overline{w} \cdot \overline{x})z = (w+x) \cdot y + \overline{(w+x)} \cdot z$

Proof: $wy + xy + yz + (\overline{w} \cdot \overline{x})z$	**Identity Used**
=	( )
=	( )
=	( )

(v) The following Boolean relation can be proved using algebraic manipulation in three steps, using one identity in each step. Specify which identity is used in each of your three steps.

Relation: $(\overline{x} \cdot \overline{y} + z) \cdot \overline{((x+y)z)} = \overline{x} \cdot \overline{y}$

Proof: $(\overline{x} \cdot \overline{y} + z) \cdot \overline{((x+y)z)}$	**Identity Used**
=	( )
=	( )
=	( )

[10] **Q2**. Consider the logic $f(w,x,y,z) = \sum m(0,1,2,5,6,7,9,10,11,12,13,14)$. A Karnaugh map for *f* is shown below.

w x / y z	0 0	0 1	1 1	1 0
0 0	1	0	1	0
0 1	1	1	1	1
1 1	0	1	0	1
1 0	1	1	1	1

You are to derive a number of covers for *f* from its K-map, but not necessarily of the lowest cost. All covers are required to be either sum-of-products or product-of-sums. The cost is calculated as (the number of gates) + (the number of inputs to gates). Inversion of an input variable has a cost of zero.

(i) Derive three expressions for *f* that each has a cost of **25**. For the first two answers below we give part of the solution and you have to fill in the rest. For the third answer (on the next page), derive the whole solution yourself.

1) $f = \overline{w} \cdot \overline{x} \cdot \overline{y} + w \cdot x \cdot \overline{y} +$ _____

2) $f = \overline{w} \cdot \overline{x} \cdot \overline{z} + \overline{w} \cdot x \cdot y +$ _____

(You can use the K-map given at the top of this page and the one below for rough work)

w x / y z	0 0	0 1	1 1	1 0
0 0	1	0	1	0
0 1	1	1	1	1
1 1	0	1	0	1
1 0	1	1	1	1

3) $f =$ _____

_____

(You can use the K-map given below for rough work)

y z \ w x	0 0	0 1	1 1	1 0
0 0	1	0	1	0
0 1	1	1	1	1
1 1	0	1	0	1
1 0	1	1	1	1

(ii) Derive two expressions for *f* that each has a cost of 31. For the answers below we give part of the solution and you have to fill in the rest.

1) $f = w \cdot x \cdot \bar{z} + \bar{x} \cdot y \cdot \bar{z} +$ _____

_____

2) $f = \overline{w} \cdot y \cdot \bar{z} + w \cdot y \cdot \bar{z} +$ _____

_____

(You can use the two K-maps given on the next page for rough work)

K-MAPS for ROUGH WORK

w x / y z	0 0	0 1	1 1	1 0
0 0	1	0	1	0
0 1	1	1	1	1
1 1	0	1	0	1
1 0	1	1	1	1

w x / y z	0 0	0 1	1 1	1 0
0 0	1	0	1	0
0 1	1	1	1	1
1 1	0	1	0	1
1 0	1	1	1	1

[10] **Q3**. Consider a sequential digital system with an input *x* and a *clock*, and the output *z*. Starting from an initial reset state, this system examines the value of *x* in three successive clock cycles, and if exactly two "1" bits are detected out of these three bits, then the system has to set the output *z* to 1 in the following clock cycle. Otherwise *z* has to be 0. The system then resumes checking the next 3-bits of the data stream *x*. You are to give a Moore-type state diagram for this FSM. You can assume that the system is reset to start the process. Use as few states as possible in your design.

ANSWER:

[12] **Q4.** Consider the Moore-type finite state machine, with input *x* and output *z*, specified by the state assigned table given below:

Current State		Next State		Output, z
		x = 0	x = 1	
Name	$y_2\ y_1\ y_0$	$Y_2\ Y_1\ Y_0$	$Y_2\ Y_1\ Y_0$	
A	000	000	001	0
B	001	001	100	0
C	010	010	001	0
D	011	001	010	1
E	100	011	100	1

(i) Draw the State Diagram that corresponds to this table.

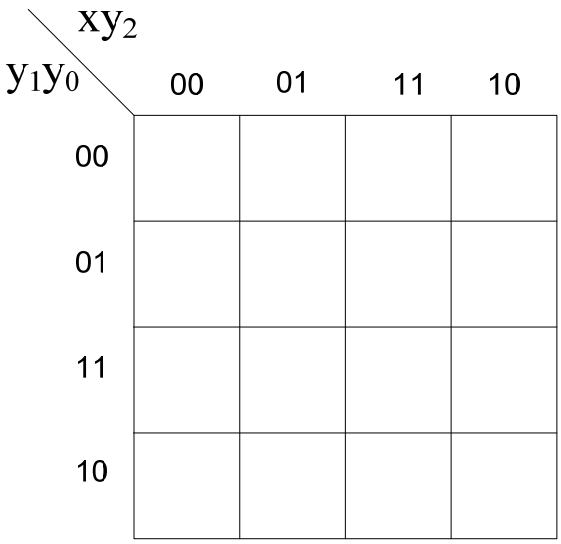
ANSWER:

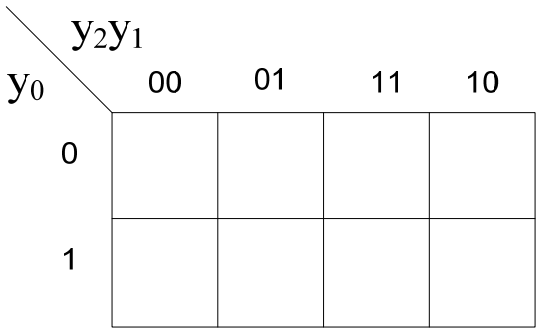
Q4, continued.

(ii) You are to derive an optimized *portion* of the gate-level design of this machine: the output logic function, and the next state logic *for only* state bit Y_0. Using another copy of the state assigned table given below and the two Karnaugh maps determine the optimized sum-of-products expression for the output logic and the next-state logic for state bit Y_0.

Current State		Next State		Output, z
		x = 0	x = 1	
Name	$y_2\ y_1\ y_0$	$Y_2\ Y_1\ Y_0$	$Y_2\ Y_1\ Y_0$	
A	000	000	001	0
B	001	001	100	0
C	010	010	001	0
D	011	001	010	1
E	100	011	100	1

ANSWER:





Optimized logic expression for the output logic, z:

z = _____

Optimized logic expression for the next state bit Y_0:

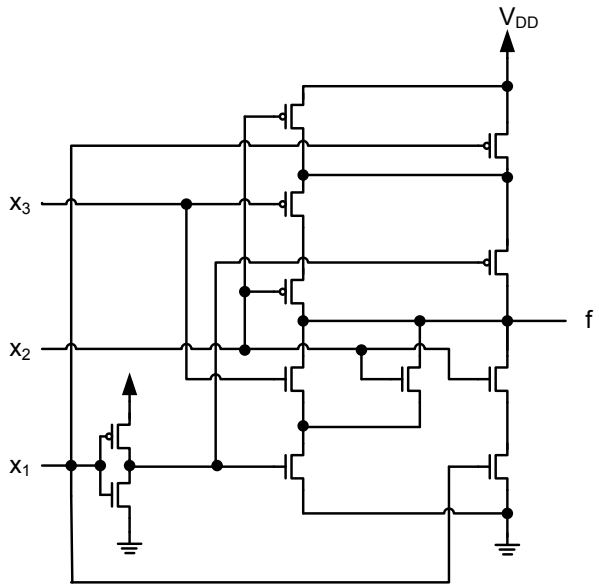
Y_0 = _____

(iii) Give a circuit diagram for this FSM that shows all three state flip-flops, but just the next-state logic for state flip-flop y_0 and the output logic.

ANSWER:

[8] **Q5**. Consider the CMOS circuits below.

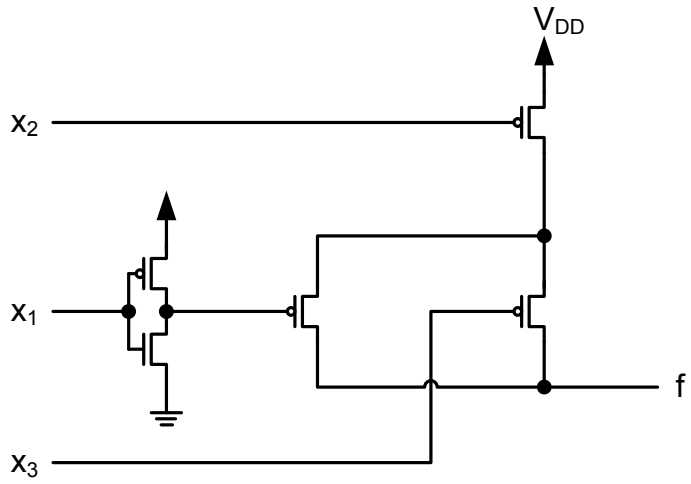
(i) For the CMOS circuit shown below, write a logic expression for the output *f*.



ANSWER: f = _____

_____

ii) The pull-up network (PUN) of a logic function is given below. Complete the CMOS logic network by drawing the correct pull-down network (PDN) in the same diagram.



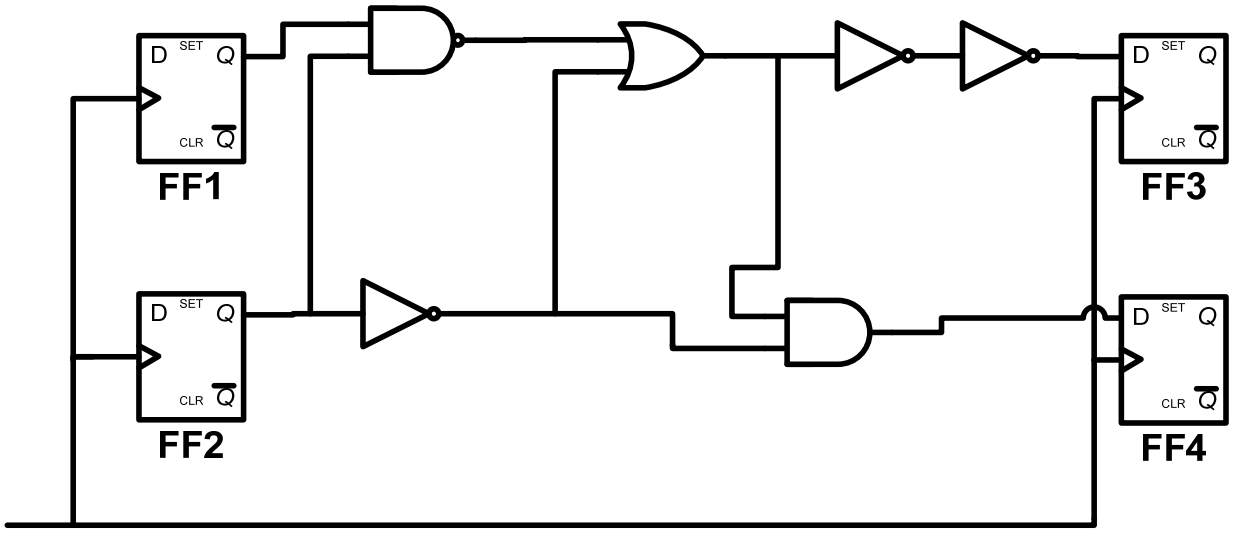
[8] **Q6.** Consider the logic function f (a, b, c) = ∑m(2, 4, 6, 7)

(i) Draw a circuit that implements the function *f* using only: **one** 2-to-1 multiplexer, **one** 2-input AND gate, and **one** 2-input OR gate.

(ii) Draw a circuit that implements the function *f* using only 2-to-1 multiplexer(s) and/or 4-to-1 multiplexer(s) and use as few as possible.

[8] **Q7**. Consider the circuit below for which the maximum and minimum propagation delay parameters of the gates and flip-flops are given in the table, and for which the setup time (t_{su}) of the flip-flops is 3 ns; the hold time (t_h) is 2 ns.

Timing Parameter	Propagation Delay (ns)	
	Max	Min
t_{INV}	1	0.5
t_{NAND}	2	1
t_{AND}	3	1.5
t_{OR}	4	2
$t_{Clock\text{-}to\text{-}Q}$	2	1



(i) What is the minimum period that the clock in the above circuit can have at which this circuit will operate correctly? Show how you arrive at your answer.

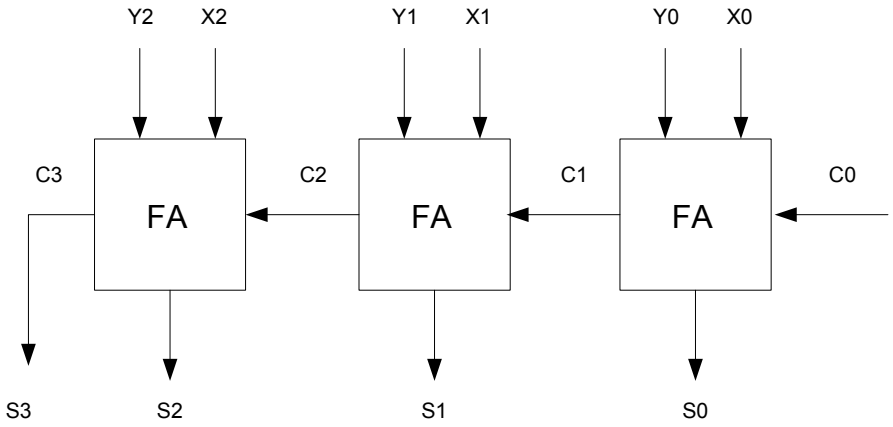
ANSWER:

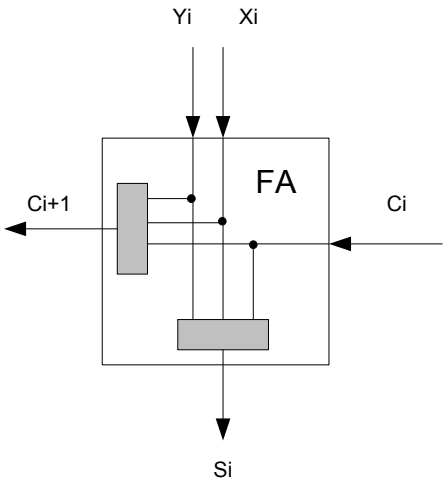
Q7, continued

(ii) Are the hold time requirements for flip-flops FF3 and FF4 met? Show how you arrive at your answer.

ANSWER:

[8] **Q8**. A 3-bit ripple-carry adder is built using 3-input lookup tables (LUTs) as illustrated below. The gray rectangles in the detailed FA block represent the 3-input LUTs.





(i) Assume that the propagation delay through a 3-input LUT is 3ns and that all inputs to this adder (X_i, Y_i and C_0) arrive at the same time. How long does it take to compute the 4-bit sum? Show how you arrive at your answer, and be sure to identify *all* critical paths.

ANSWER:

Q8, continued.

(ii) Give the design of the fastest possible 3-bit adder that can be implemented using **5-input** LUTs. Assume that all inputs arrive at the same time, and that the propagation delay of a 5-input LUT is 4 ns. You should also try to use as few 5-input LUTs as possible.

Draw a schematic of your circuit. Label the inputs and outputs of the 5-LUTs, and indicate the function of each LUT by writing a logic expression. Your expression can use AND, OR, NOT, and XOR operations.

Indicate clearly: (1) How long it takes (in ns) for your adder to compute the sum.
(2) How many 5-input LUTs you used (i.e. give the count).

ANSWER:

[10] **Q9**. Draw a schematic diagram using only D flip-flops, AND, OR, and NOT gates that corresponds to the Verilog code given below. Show the inputs SW and KEY, and the outputs LEDR on your schematic.

```
module some_thing (SW, KEY, LEDR);
   input [1:0] SW ;
   input [0:0] KEY ;
   output [3:0] LEDR;

   wire thing_1 = KEY[0];
   wire thing_2 = SW[0];

   wire [3:0] Z;
   wire [3:0] edgar;

   assign edgar[0] = SW[1];
   sub_circuit U1 (edgar[0], thing_1, thing_2, Z[0]);
   assign edgar[1] = Z[0] & edgar[0];
   sub_circuit U2 (edgar[1], thing_1, thing_2, Z[1]);
   assign edgar[2] = Z[1] & edgar[1];
   sub_circuit U3 (edgar[2], thing_1, thing_2, Z[2]);
   assign edgar[3] = Z[2] & edgar[2];
   sub_circuit U4 (edgar[3], thing_1, thing_2, Z[3]);

   assign LEDR = Z;

endmodule


module sub_circuit(in, edge_1, edge_2, out);
   input in, edge_1, edge_2;
   output reg out;

   always @(posedge edge_1)
      if (edge_2  == 1'b0)
         out <= 1'b0;
      else if (in)
         out <= ~out;

endmodule
```

PLEASE SHOW YOUR SCHEMATIC ON THE NEXT PAGE.

SHOW SCHEMATIC HERE: