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 10*a*, 10*b* ... 17*a*, and 17*b*. Note that identity 17, called *consensus*, is not in the first edition of the textbook. It is:

$xy + yz + \bar{x}z = xy + \bar{x}z$ 17*a*

$(x + y)(y + z)(\bar{x} + z) = (x + y)(\bar{x} + z)$ 17*b*

An example problem and solution is given below. Note that

Example:

(e.g.) Prove the following Boolean relation using algebraic manipulation in two steps, using exactly two identities. 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 (vi) 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**	
$= \overline{(w \oplus x)} + \bar{y}z$	(12*b*)	1 mark

(ii) Prove the following Boolean relation using algebraic manipulation in two steps, using exactly two identities. 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			
$= (x + y)z + x + y$	(12*a*)	or	$= x + yz + y$	(13*a*) 1 mark
$= x + y$	(13*a*)	or	$= x + y$	(13*a*) 1 mark

(iii) Prove the following Boolean relation using algebraic manipulation in two steps, using exactly two identities. 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**
$= \overline{x} + \overline{(y+z)}$	(15*a*) 1 mark
$= \overline{x} + \overline{y} \cdot \overline{z}$	(15*b*) 1 mark

(iv) Prove the following Boolean relation using algebraic manipulation in three steps, using exactly three identities. 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**
$= (w+x)y + yz + (\overline{w} \cdot \overline{x})z$	(12*a*) 1 mark
$= (w+x)y + yz + \overline{(w+x)}z$	(15*a*) 1 mark
$= (w+x)y + \overline{(w+x)}z$	(17*a*) 2 marks

(v) Prove the following Boolean relation using algebraic manipulation in three steps, using exactly three identities. 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**
$= (\overline{x} \cdot \overline{y} + z) \cdot (\overline{(x+y)} + \overline{z})$	(15*a*) 1 mark
$= (\overline{x} \cdot \overline{y} + z) \cdot (\overline{x} \cdot \overline{y} + \overline{z})$	(15*b*) 1 mark
$= \overline{x} \cdot \overline{y}$	(14*b*) 1 mark

[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.

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

You are to derive a number of covers for f from the above 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 no cost.

(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 solution yourself.

1) $f = \overline{w} \cdot \overline{x} \cdot \overline{y} + w \cdot x \cdot \overline{y} + y\overline{z} + \overline{w}xz + w\overline{x}z$ (2 marks; -1 for each wrong term, or extra term)

2) $f = \overline{w} \cdot \overline{x} \cdot \overline{z} + \overline{w} \cdot x \cdot y + \overline{y}z + w\overline{x}y + wx\overline{z}$ (2 marks; -1 for each wrong term, or extra term)

(You can use the K-map 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

3) $f = (w + \bar{x} + y + z)(\bar{w} + x + y + z)(w + x + \bar{y} + \bar{z})(\bar{w} + \bar{x} + \bar{y} + \bar{z})$ (2 marks; -1 for each wrong term, or extra term)

(You can use the K-map 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) $f = w \cdot x \cdot \bar{z} + \bar{x} \cdot y \cdot \bar{z} + w\bar{x}z + x\bar{y}z + \bar{w}xy + \bar{w}\bar{x}\bar{y}$ (2 marks; -1 for each wrong term, or extra term)

2) $f = \bar{w} \cdot y \cdot \bar{z} + w \cdot y \cdot \bar{z} + w\bar{x}z + wx\bar{y} + \bar{w}\bar{x}\bar{y} + \bar{w}xz$ (2 marks; -1 for each wrong term, or extra term)

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

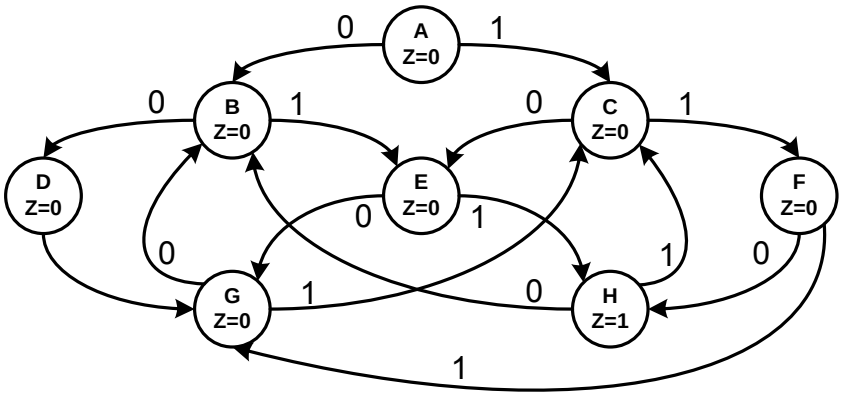
K-MAPS for ROUGH WORK

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

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

[10] **Q3**. A system checks an incoming serial bit stream, *x*. It examines the value of *x* in three successive clock cycles, and if exactly two "1" bits are detected, then the system has to set an 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:



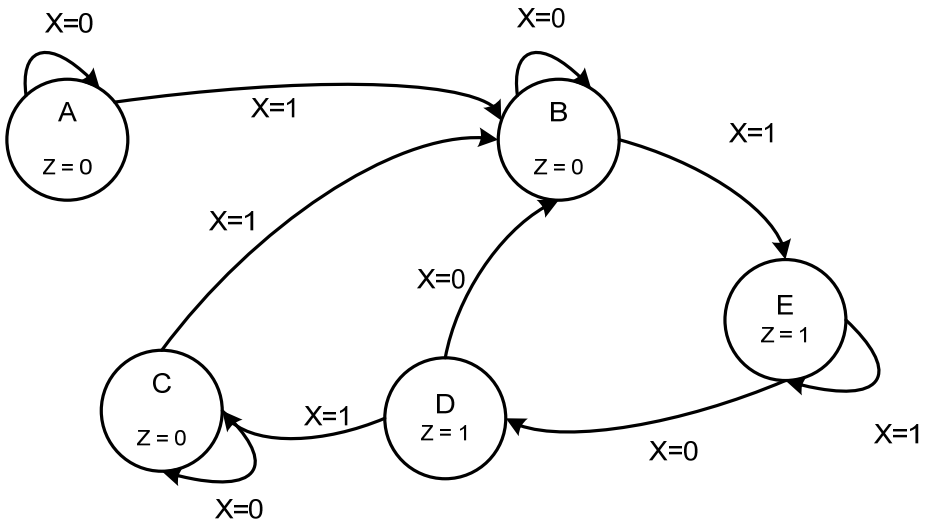
-1 mark for each extra state
-1 mark for each wrong link between states

[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:



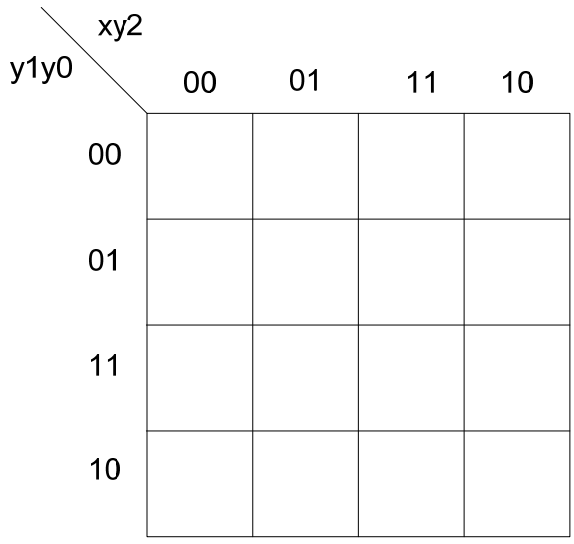
Marking Scheme: Worth 4 marks total; -0.5 for every error to max of 4. An error can be a missing arc, wrongly labeled arc, missing output, or wrongly labeled state. Exceptions – if forget the outputs entirely, just take off a maximum of 2.

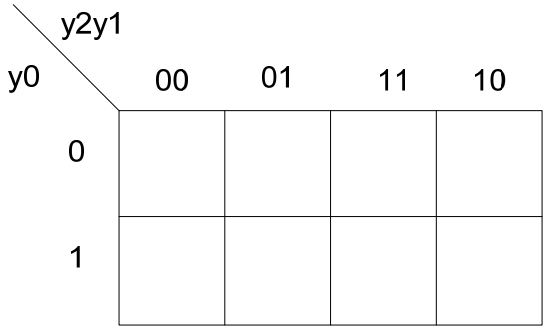
Q4, continued.

(ii) You are to determine an optimized *portion* of the gate-level design of this machine: the output logic function, and the next state logic *for only* state bit y0. 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 y0.

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:

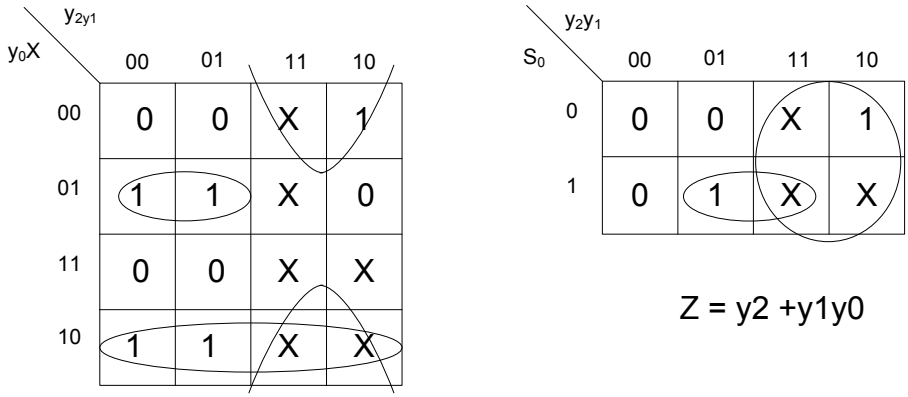




Optimized Logic Expression for the Output Logic, Z:

Optimized Logic Expression for the State Bit y_0:

SOLUTION:



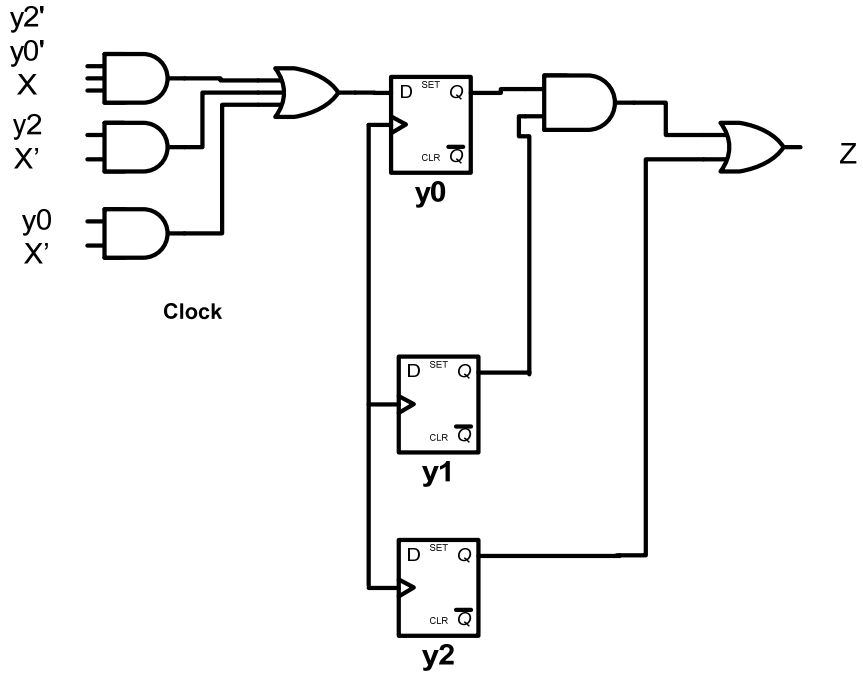
Y0 = y0X' + y2'X' + y2'y0'X

Z = y2 +y1y0

Total marks – 5 - 3 for y0 and 2 for Z. Note this table has different labels than the one in the question.

(iii) Give the Circuit Diagram for this FSM that shows all three state flip-flops, but just the next state logic for state bit S0 and the output logic.

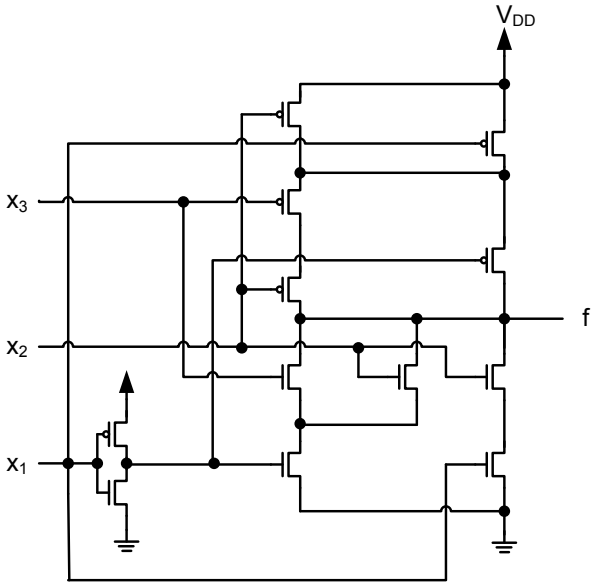
ANSWER:
SOLUTION:



Worth 3 marks – 1 for the three flip-flops, 1 for the next state logic, 1 for output logic
-0.5 for each mistake in each of the three areas.

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

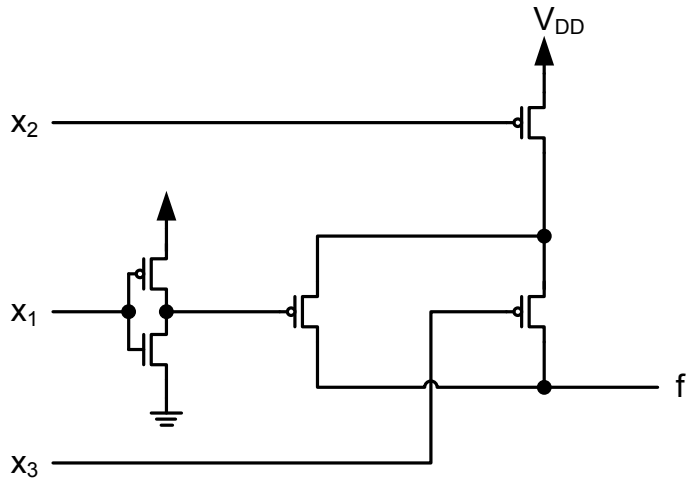
(i) For the CMOS circuit shown below, write the logic expression for the output function, f.

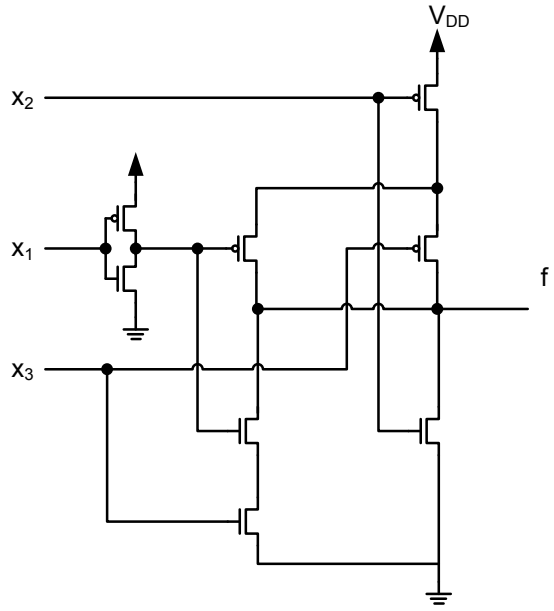


[5] $f = \overline{x_1 x_2 + \overline{x_1}(x_2 + x_3)}$

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.

Solutions:





[3] 1 marks for each correctly drawn transistor

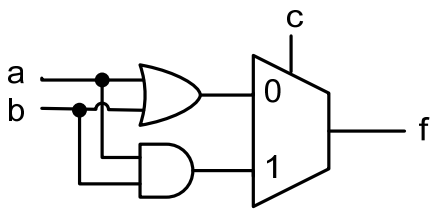
[8] **Q6.** Consider the logic function f (a, b, c) = ∑m(2, 4, 6, 7)

(i) Implement the function *f* using **one** 2-to-1 multiplexer, **one** 2-input AND gate, and **one** 2-input OR gate.

Solutions: f (a, b, c) = ∑m(2, 4, 6, 7) $= \bar{a} \bullet b \bullet \bar{c} + a \bullet \bar{b} \bullet \bar{c} + a \bullet b \bullet \bar{c} + a \bullet b \bullet c$

$= \bar{c}\left(\bar{a}b + a\bar{b} + ab\right) + c(ab)$

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

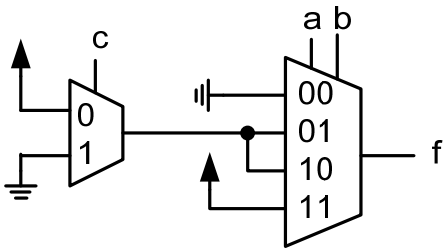


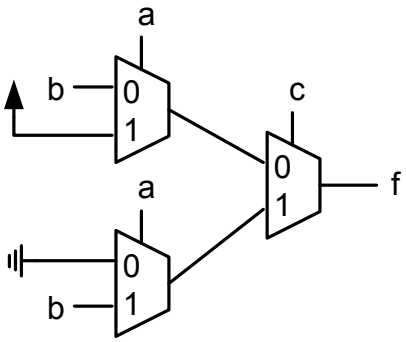
[3]

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

Solutions: f (a, b, c) = ∑m(2, 4, 6, 7) $= \bar{a} \bullet b \bullet \bar{c} + a \bullet \bar{b} \bullet \bar{c} + a \bullet b \bullet \bar{c} + a \bullet b \bullet c$

$= \bar{a} \bullet \bar{b}(0) + \bar{a} \bullet b(\bar{c}) + a \bullet \bar{b}(\bar{c}) + a \bullet b(1)$





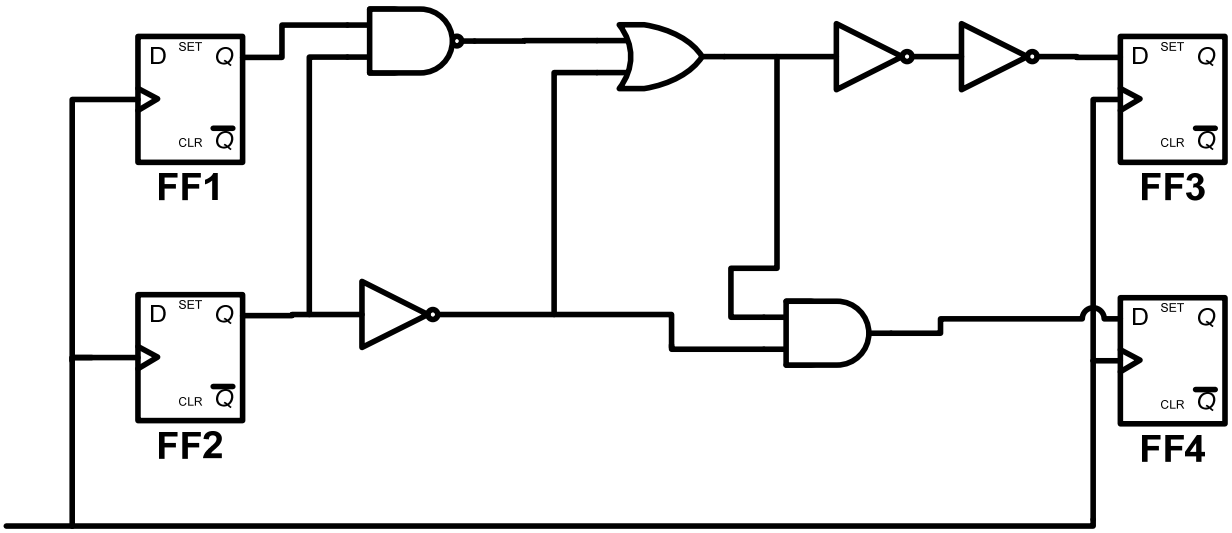
[5]

4 marks

-1 mark for each extra mux used
-1 mark for each logic gate used

[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_{HOLD}) 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:
Solution:
Must look at path to both FF3 and FF4

FF3 min = T(clock-to-q) + max(Tinv, Tnand) + Tor + 2 x Tinv + Tsetup
= 2 + max(2,1) + 4 + 2x1 + 3
= 2 + 2 + 4 + 2 + 3
= 13 ns

FF4 min = T(clock-to-q) + max(Tinv, max(Tinv, Tnand) + Tor) + Tand+ Tsetup
= 2 + max(1, max(1,2) +4) + 3 + 3
= 2 + 6 + 3 + 3

= 14ns Which is greater than 13ns, so this is the answer.

Marking scheme – 3 marks for correct answer, 1 more if calculation showing DA is given. Part marks if some of the correct path is shown.

Q7, continued

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

ANSWER:

SOLUTION:

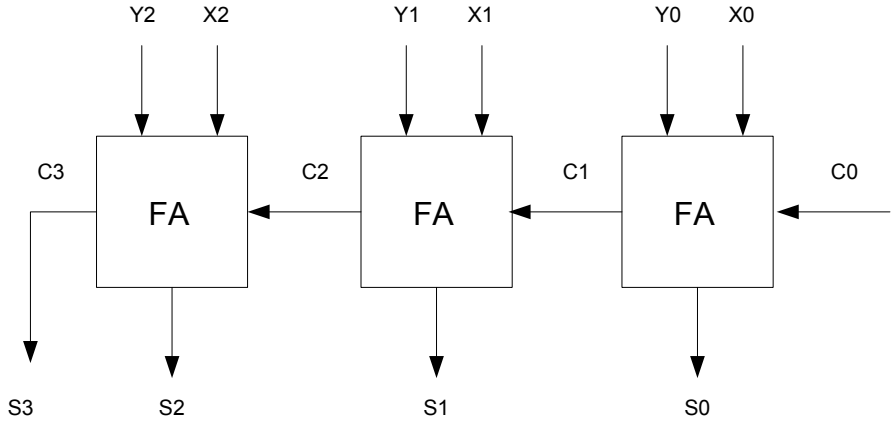
FF4 min change = min of Tclock-to-Q + Tinv + AND
= 1 + 0.5 + 1.5 = 2.5ns >2ns yes

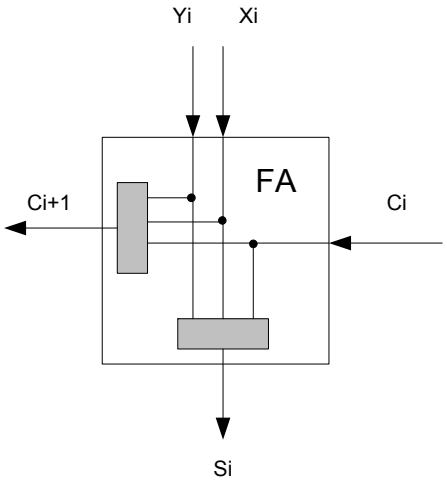
FF3 min change = min of Tclock-to-Q + min(Tinv,Tnand) + Tor + Tinv + Tinv + AND
= 1 + min(0.5,1) + 2 + 0.5 +0.5
= 1 + 0.5 + 2 + 0.5 +0.5
= 4.5ns > 2ns YES

Both FF's hold times are **not** violated.

Total marks – 4, worth 2 each – 1 for answer and 1 for work.

[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 computer the 4-bit sum? Show how you arrive at your answer, and be sure to identify *all* critical paths.

ANSWER:
SOLUTION: Both S3 and S2 have a 3 LUT delay x 3ns = 9ns; 1 for correct answer, 1 for identifying S3 and S2 as critical.

Q8, continued.

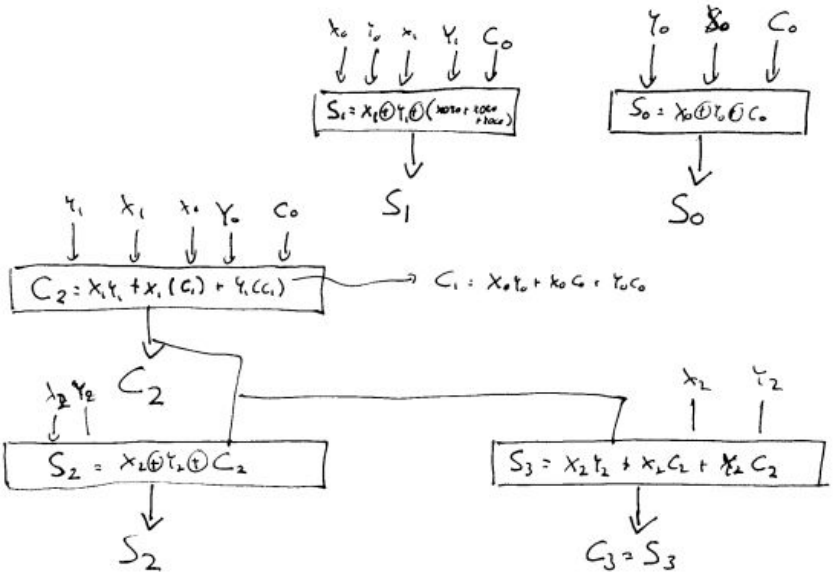
(ii) Give the design of the fastest possible 3-bit adder that can be implemented using **5-input** LUTs. Assume that all inputs arrive at the same time, and that the propagation delay of a 5-input LUT is 4ns. You should also try to use as few 5-input LUTs as possible.

Draw a schematic show the labels of the inputs and outputs of the 5-LUTs, and then indicate the function of the LUT by writing a logic expression. Your expression can employ both ANDs, ORs and X-OR symbols.

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

ANSWER:
SOLUTION:

Q3 (b) Solution



Total delay = 2 5LUTs = 2 × 4 = 8ns.

Totall # 5-LUTs = 5

6 marks 4 for right delay + circuit
2 for # 5-LUTs right

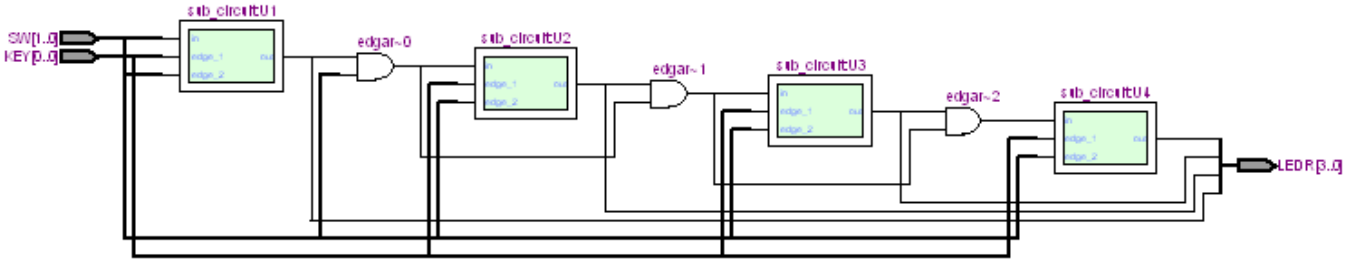
[10] **Q9**. Draw a schematic diagram using only D flip-flops, AND, OR, and NOT gates that corresponds to the Verilog code given below. Show the inputs SW and outputs LEDR on your schematic.

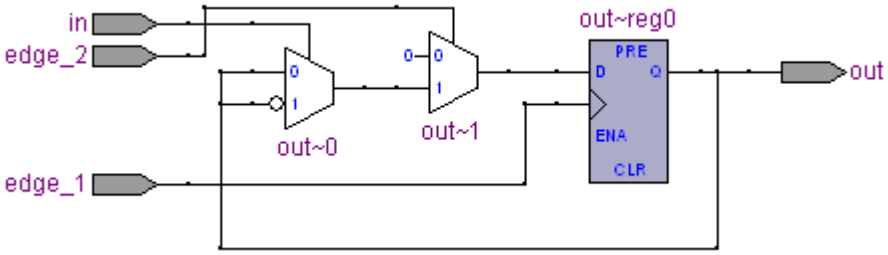
```
module some_thing (SW, KEY, LEDR);
   input [1:0] SW ;
   input [0:0] KEY ;
   output [3:0] LEDR;

   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.

SOLUTIONS SOLUTIONS SOLUTIONS SOLUTIONS SOLUTIONS SOLUTIONS

SHOW SCHEMATIC HERE:





5 marks total for the D flip-flop details:

- 1 mark for D FF itself
- 2 marks for synchronous clear (AND gate is fine too)
- 2 marks for mux that loads the FF with either Q or !Q

5 marks for overall circuit structure that connects the four FFs together:

- 1 mark for the AND gates used as enables between FFs
- 1 mark for showing that SW[0] is used as the reset input
- 1 mark for showing that SW[1] is used as the counter enable
- 1 mark for showing that KEY[0] is used as the clock
- 1 mark for showing that LEDR is connected to the FF outputs