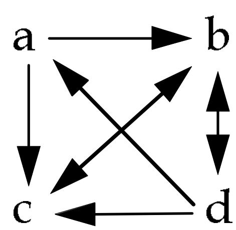
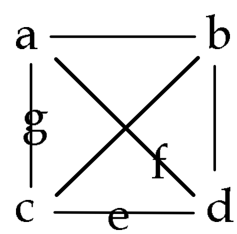
(4) (b) Obtain a **race-free state assignment** for the merged flow table shown on the right (*not* related to part 4(a)) using **as few additional states as possible. List all adjacency constraints. Show all steps and your final state-codes.** (Hint: three state variables are required.)

x1 x2			
00	**01**	**11**	**10**
1	2	3	4
7	6	3	5
7	6	8	4
7	2	3	5

4. (b) (10pts)





	x1 x2			
	00	**01**	**11**	**10**
a	**a**	**a**	**b**	**g**
b	**c**	**b**	**b**	**d**
c	**c**	**b**	**c**	**c**
d	**e**	**f**	**b**	**d**
e	**c**	**-**	**-**	**-**
f	**-**	**a**	**-**	**-**
g	**-**	**-**	**-**	**c**

Q3 \ Q1 Q2	00	01	11	10
0	b 0	c 2	g 6	a 4
1	d 1	e 3	- 7	f 5

If you noted that in fact, under the fundamental mode assumption, the 2 in the final row is really a don't-care, and that as a result we only need two additional states really (i.e. we don't need state f above), you were given bonus points.

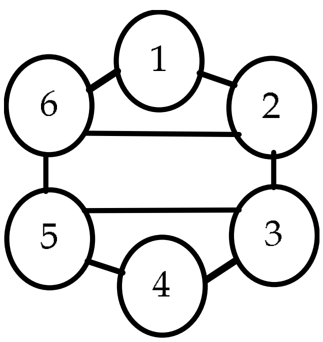
Additional space for Problem 4(b)

(4) (a) For the two-input primitive flow table shown on the right, obtain the **merged flow table** by merging states to produce a table with the **minimum possible number of rows**. Show your merger diagram (graph). Assume all outputs are the same value. ***Do not* attempt to reduce the table** by eliminating equivalent states first; all states *must* appear in your final answer.

x1 x2			
00	**01**	**11**	**10**
1	5	6	2
1	-	-	2
-	5	3	2
4	5	3	-
-	5	6	2
4	5	-	-

4. (a) (10pts)

Merger diagram and minimum-row merged flow table



Merger Diagram

x1 x2			
00	**01**	**11**	**10**
1	5	6	2
4	5	3	2

Minimum-row merged flow table

Additional space for Problem 4(a)

(c) Add the following numbers in base-(-2). Pay close attention to the carry-in and carry-out aspects because they are important in Part (d)

3(c) (2pts)

$001110_{-2} + 001111_{-2}$ = ____110101__ $_{-2}$

(d) Draw a truth table for a one-bit bit slice of a base-(-2) adder. Assume one bit of carry-in. You must determine the specification for the carry-out.

3(d) (4pts)

C_i	A_i	B_i	S_i	C_{i+1}	C_{i+2}
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	1	0	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	0	1	1
1	1	0	0	1	1
1	1	1	1	1	1

Additional space for Problem 3

(3) This problem concerns a base-(-2) (base-(minus-two)) adder. In CS150, we only considered number systems with positive bases (in particular, the binary, or base-2, system). For example, in a base-3 system, the number $ABCD_3$ is equal to:

$ABCD_3 = A*3^3 + B*3^2 + C*3^1 + D*3^0$.

So for a base-(-2) system, $EFGH_{-2}$ would be computed using:

$EFGH_{-2} = E*(-2)^3 + F*(-2)^2 + G*(-2)^1 + H*(-2)^0$

(a) Convert the following decimal numbers to base-(-2). Use as many bits as you need.

3(a) (8pts)

5_{10} = _101_ $_{-2}$ 6_{10} = _11010_ $_{-2}$ -7_{10} = _1001_ $_{-2}$ 11_{10} = _11111_ $_{-2}$

(b) For the purpose of this problem, we define a number system as *contiguous* if, given any two integer values *m* and *n* which can be represented in the number system, there exists no integer value *x*, with $m < x < n$, that cannot be represented in the system. For example, a 1-bit decimal system is contiguous. It can represent the counting numbers 0-9, and there are no numbers in between 0 and 9 which cannot be represented using one decimal digit. The binary number system (base 2) is also contiguous.

(i) Is a four-bit base-(-2) system contiguous?
(ii) What are the maximum and minimum values (in decimal) that can be represented by a four-bit base-(-2) system?
(iii) Is an n-bit base-(-2) system contiguous, where n is any positive integer? Justify your answer.

3(b) (6pts)

(i) __yes___

(ii) Maximum$_{10}$ = ___5____ **Minimum$_{10}$ =** __-10___

(iii) __yes___

For any negative bit, b_i, the associated positive value is found by adding the next bit, b_{i+1}. However, since $+2^n$ is contiguous, then -2^n must also be contiguous.

2(b) Considering the definition of setup time for a latch, is it possible for the setup time for a latch to be negative? **If so, under what conditions can the setup time for a latch be negative?** State all the conditions you can think of and illustrate with an example, where possible.

2(b) (10pts)

Katz defines setup time as "the minimum time interval preceding the clock event during which the input must be stable to be validly recognized." If the setup time were negative, that would mean that the input could change (need not be stable) for up to $|T_{su}|$ *after* the clock event. This would only be possible if the circuitry from the latch input to the clocked storage element was faster than the delay from the clock edge on the clock pin to the internal storing event in the latch. In fact, this is the case in a number of practical latches. For example, certain Altera PLD families have negative setup-time latches (see the spec. sheets). This is often achieved by using a very fast Schmitt trigger circuit on the input to the latch to "speed up" the input. Such latches are very effective in helping to speed up the overall performance of a circuit; something not possible with the more conservative Xilinx parts.

Another way of viewing the situation is to imagine a very slow clock input transition while having a fast transition on the inputs.

....

....

....

....

....

....

....

....

....

....

(2) (a) For the following state table, where x is the input and Z is the output:

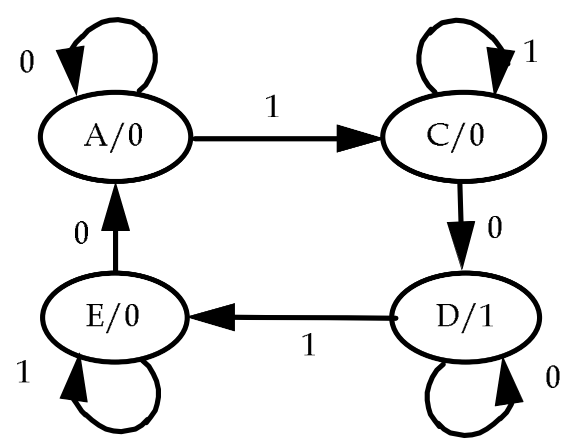
PS	NS		Z
	x=0	x=1	
A	B	C	0
B	A	C	0
C	D	C	0
D	D	E	1
E	A	F	0
F	B	G	0
G	A	E	0

(i) Use an **implication table** to identify and eliminate all redundant states. **Show your final implication table and the state table with no redundancy**.

(ii) Show a **state graph** for the final irredundant machine. Include all information on the graph, including inputs and outputs.

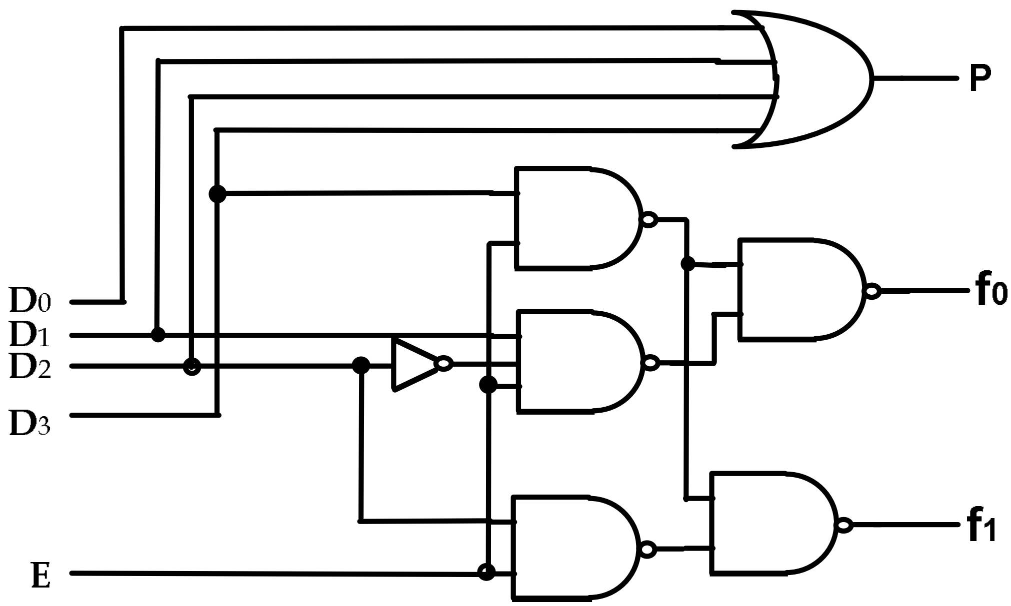
2(a) (10pts)

Equivalent states from implication table are (AB) → A, (C) → C, (D) → D, (EFG) → E



(b) Draw a **schematic diagram** for the entire priority encoder, including output P and input E in the schematic, using a **minimum number of NAND, OR, and inverter** gates only. Assume **complements are not available**.

1(b) (10pts)



Additional space for Problem1

UNIVERSITY OF CALIFORNIA AT BERKELEY

BERKELEY • DAVIS • IRVINE • LOS ANGELES • RIVERSIDE • SAN DIEGO • SAN FRANCISCO   SANTA BARBARA • SANTA CRUZ

Department of Electrical Engineering
and Computer Sciences

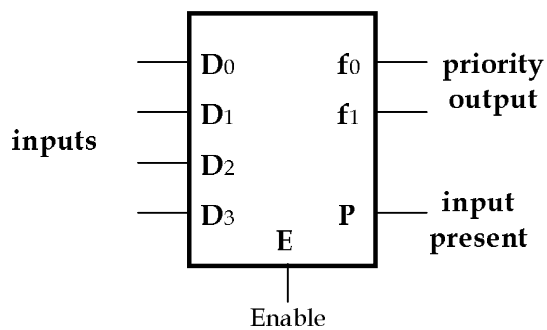
CS 150 - Spring 1996
Prof. A. R. Newton

Quiz 2 Solutions

Room 10 Evans Hall, 2:10pm Tuesday April 2
(Open Katz only, Calculators OK, 1hr 20mins)

Include all final answers in locations indicated. Use space provided for all working and state all assumptions. If necessary, attach additional sheets by staple at the end. BE SURE TO WRITE YOUR NAME ON EVERY SHEET.

(1) You are to design a **four-input priority encoder**, shown below, such that when two inputs, D_i and D_j, are high simultaneously, D_i has priority over D_j when i>j. The encoder produces a binary output code (f_1 f_0) corresponding to the input that has the highest priority. The circuit should also have an enable input, E, such that (f_1 f_0) are set to (00) when E is low, and an output P which indicates the presence of data (1's) on any of the inputs.



(a) Show a **truth-table** for the priority encoder, in terms of D_0 to D_3 only, and derive the **logic equations for f_0 and f_1.** Express the logic equations using a **minimum number of literals.**

1(a) (10pts)

D_0	D_1	D_2	D_3	f_1	f_0
0	0	0	0	-	-
1	0	0	0	0	0
-	1	0	0	0	1
-	-	1	0	1	0
-	-	-	1	1	1

You lost points if you did not indicate the output don't cares for the 0000 case.

f0 = ____ $D_3 + D_1 \bullet (D_2)'$____________   **f_1 =** ____ $D_2 + D_3$______________