ECE 303
HW #1
2006.10.05

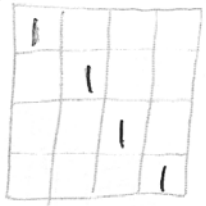
1.) $f(a,b)$ $\quad a=(a_1,a_0)$ $\quad b=(b_1,b_0)$

a.) $f(a,b)=\bar{a}_1\bar{a}_0\bar{b}_1\bar{b}_0+\bar{a}_1 a_0\bar{b}_1 b_0+a_1\bar{a}_0 b_1\bar{b}_0+a_1 a_0 b_1 b_0$

b.)

a_1	a_0	b_1	b_0	$f(a,b)$
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

c.)



d.) 2^n and gates are needed because when n increases by 1, the number of combinations where $a=b$ doubles. Thus, for a two level circuit, each combination needs an and gate and there are 2^n possible combinations.

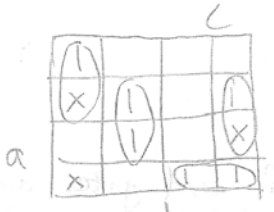
2.) a.)

a	b	c	d	m	
0	0	0	0	0	1
0	0	0	1	1	0
0	0	1	0	2	0
0	0	1	1	3	0
0	1	0	0	4	1
0	1	0	1	5	1
0	1	1	0	6	0
0	1	1	1	7	0
1	0	0	0	8	1
1	0	0	1	9	0
1	0	1	0	10	x
1	0	1	1	11	0
1	1	0	0	12	1
1	1	0	1	13	1
1	1	1	0	14	x
1	1	1	1	15	0

$\bar{a}\bar{b}\bar{c}\bar{d} + (\bar{a}b\bar{c}\bar{d}) + \bar{a}bc\bar{d} + [a\bar{b}\bar{c}\bar{d}] + [\bar{a}\bar{b}cd] + (a\bar{b}\bar{c}\bar{d})$

$\bar{a}\bar{b}\bar{c}\bar{d} + b\bar{c}\bar{d} + \bar{a}bc\bar{d} + a\bar{b}c \quad : [\, ab + a\bar{b} = a \,]$

$f = \bar{a}\bar{b}\bar{c}\bar{d} + b\bar{c}\bar{d} + \bar{a}bc\bar{d} + a\bar{b}c$



$f = \bar{a}\bar{c}\bar{d} + b\bar{c}d + a\bar{b}c + bc\bar{d}$

d = ~~0100~~ ; ~~1000~~ ; ~~1110~~

PI = ~~0000~~ ; ~~0101~~ ; ~~0110~~ ; ~~1010~~ ; ~~1011~~ ; ~~1101~~

$\Sigma = 0$: 0000 ✓

$\Sigma = 1$: 0100 ~ , 1000 ✓

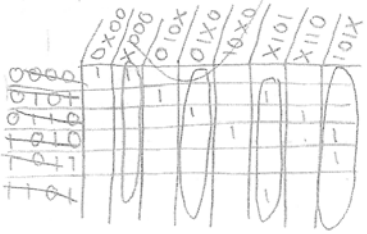
$\Sigma = 2$: 0101 ✓ , 0110 ✓ , 1010 ~

$\Sigma = 3$: 1011 ✓ , 1101 ✓ , 1110 ✓

0x00 , x000

010x , 01x0 , 10x0

x101 , x110 , 101x



$f = a\bar{b}c + b\bar{c}d + \bar{a}b\bar{d} + \bar{b}\bar{c}\bar{d}$

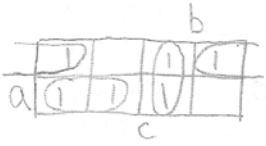
b.)

a	b	c	m	f
0	0	0	0	1
0	0	1	1	-
0	1	0	2	1
0	1	1	3	1
1	0	0	4	1
1	0	1	5	1
1	1	0	6	0
1	1	1	7	1

$\bar{a}\bar{b}\bar{c} + \bar{a}b\bar{c} + \bar{a}bc + a\bar{b}\bar{c} + a\bar{b}c + abc$

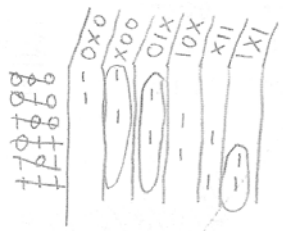
$\bar{a}\bar{c} + bc + a\bar{b}$ $(\bar{a}b + ab = b)$

$\bar{a}\bar{c} + bc + a\bar{b}$



$\bar{a}\bar{c} + bc + a\bar{b}$

$\Sigma = 0$	000	0X0
		X00
$\Sigma = 1$	010	01X
	100	10X
$\Sigma = 2$	011	X11
	101	1X1
$\Sigma = 3$	111	



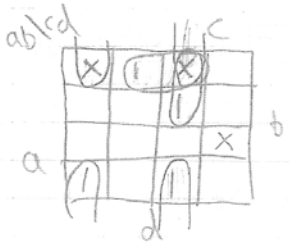
$\bar{b}\bar{c} + \bar{a}b + ac$

This problem is a cyclic core. This affects minimization because there are multiple correctly reduced choices.

c.)

a	b	c	d	f(a,b,c,d)
0	0	0	0	X
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	X
1	1	0	0	1
1	1	0	1	0
1	1	1	0	X
1	1	1	1	0

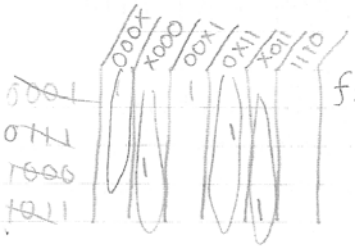
$f = \bar{a}\bar{b}\bar{c}d + \bar{a}bcd + a\bar{b}\bar{c}\bar{d} + a\bar{b}cd$



$f = \bar{b}\bar{c}\bar{d} + \bar{a}\bar{b}d + \bar{a}cd + \bar{b}cd$

min	d
~~0001~~	~~0000~~
0111	~~0011~~
~~1000~~	1110
1011	

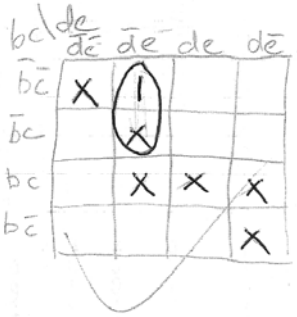
$\Sigma=0$	0000 ✓	000X
$\Sigma=1$	0001 ✓	X000
	1000 ✓	00X1
$\Sigma=2$	0011 ✓	0X11
$\Sigma=3$	0111 ✓	X011
	1011 ✓	1110
	1110	



$f = \bar{a}\bar{b}\bar{c} + \bar{b}\bar{c}\bar{d} + \bar{a}cd + \bar{b}cd$

3.) $a = 0$

$a = 1$



$\bar{a}\bar{b}\bar{e} + \bar{a}be + \bar{b}\bar{d}e + \bar{a}b\bar{e}$

Problem 4: **(5 points)** Minimize the following following functions using K-maps. Give your answer in POS form. In this notation, the zeros and don't-cares are specified instead of the ones and don't-cares. You can use either of the techniques presented in class.

1. $f = \prod(1,4,5,9,11,14) + d(0,2,7,8,13)$
2. $f = \prod(2,7,10,11,14) + d(1,4,6)$

1. $f = (a+c)(\overline{a}+\overline{b}+\overline{c}+d)(\overline{a}+b+\overline{d})$
2. $f = (a+\overline{b}+\overline{c})(\overline{a}+b+\overline{c})(\overline{c}+d)$

Problem 5: **(5 points)** Define the following terms:

1. Observability Don't-care
2. Satisfiability Don't-care

1. Observability Don't-care

   For some input patterns, the outputs of the system won't change if one output bit of some subsystem changes. So that bit of that subsystem is a don't-care for such input patterns.

2. Satisfiability Don't-care

   Some input patterns never appear. The outputs of the system are don't-cares for such input patterns.

Problem 6: **(10 points)** What techniques are available for two-level logic minimization? Briefly (a word or two for each), what are the advantages and disadvantages of each?

1. Algebraic

   Not a systematic method. Hard to reach optimal.

2. K-map

   Fast for human beings with no more than 4 variables. Hard to design an algorithm based on it. Hard for human beings when the size of the problem increases.

3. Quine-McCluskey

   A systematic method to convert the logic minimization problem into the unate covering problem. However, consume a lot of space and time.

4. Expresso

   Fast heuristic algorihtms that can be built into software. Only give near-optimal solutions.

Problem 7: **(5 points)** Cube $f(a,b) = a$. Cube $g(a,b) = a\overline{b}$. Does f cover g? Does g cover f?

The cube f covers g but g does not cover f.

Problem 8: **(5 points)** In two or three sentences, describe how Espresso searches for good implicants? Does it find all primes?

Expresso searches for good implicants by repeating *reduce*, *irredundant cover*, and *expand* to find alternative prime implicant. A cost function is used to determine if the new covers are improved. Heuristics are applied to pick the cubes that would be processed next. It won't find all primes.

Problem 9: **(5 points)** Cofactor $f(a,b,c,d) = \overline{a}\,b + ac + ab\overline{d}$ by the cube $a\overline{b}$.

$$\begin{aligned} f_a &= (f(a,b,c,d))_a \\ &= (\overline{a}\,b + ac + ab\overline{d})_a \\ &= 0 \cdot b + 1 \cdot c + 1 \cdot b\overline{d} \\ &= c + b\overline{d} \end{aligned}$$

$$\begin{aligned} f_{a\overline{b}} &= (f_a)_{\overline{b}} \\ &= (c + b\overline{d})_{\overline{b}} \\ &= c + 0 \cdot \overline{d} \\ &= c \end{aligned}$$

Problem 10: **(10 points)** Consider the following set of cubes:

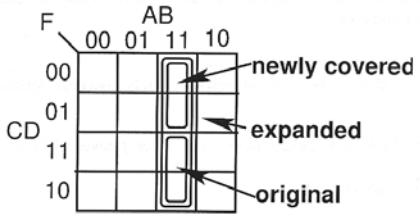
0	0	X	0	1
X	1	0	X	1
1	X	0	1	1
1	1	1	X	1

Tautology checking can be used for expansion checking. Espresso uses tautology checking to solve a number of problems, for example, irredundant cover. However, Espresso doesn't use tautology checking for expansion checking.

You would like to expand the 111X|1 cube to 11XX|1. However, you need to confirm that this expansion is valid.

1. Write down new cube that must be checked for containment.
   Treating cubes like sets of minterms, that's the expanded cube minus the original cube.

$$11XX|1 - 111X|1 = 110X|1$$



2. Write down the set of cubes within which it must be contained.

$$\begin{array}{cccc|c} 0 & 0 & X & 0 & 1 \\ X & 1 & 0 & X & 1 \\ 1 & X & 0 & 1 & 1 \end{array}$$

By the way, it will work just fine even if $111X|1$ is left in. If this doesn't make sense, cofactor the cover with $111X|1$ left in and see what happens.

3. Cofactor the set of cubes by the new cube and use the result to determine whether the cube is contained.

$$\left(\begin{array}{cccc|c} 0 & 0 & X & 0 & 1 \\ X & 1 & 0 & X & 1 \\ 1 & X & 0 & 1 & 1 \end{array} \right)_{110X} = \begin{array}{cccc|c} X & X & X & X & 1 \\ X & X & X & 1 & 1 \end{array}$$

4. Repeatedly cofactor on unate variables of the cover until it is clear that the expansion is valid or invalid.

We determine that the cover is unate, even before cofactoring. Therefore we need only scan the rows for a 1 ($XXXX|1$).

5. Explain why cofactoring on unate variables (instead of non-unate variables) makes validity checking (or, in general, tautology checking) faster.

If we cofactor on variable in which the cover is unate, then we know that the variable's column does not contain both zeros and ones. Thus, in a SOP expression for the cover, the variable appears only in its complemented or uncomplemented form. Without loss of generality, assume that the variable v_0 appears only in its uncomplemented form. We can clearly write the n-variable function in the following form:

$$f(v_0, v_1, \ldots, v_n) = v_0 \cdot f_1(v_1, v_2, \ldots, v_n) + f_2(v_1, v_2, \ldots, v_n)$$

Cofactoring by the uncomplemented form, v_0, we get

$$f(v_0, v_1, \ldots, v_n)_{v_0} = f_1(v_1, v_2, \ldots, v_n) + f_2(v_1, v_2, \ldots, v_n)$$

Cofactoring by the complemented form, $\overline{v_0}$, we get

$$f(v_0, v_1, \ldots, v_n)_{\overline{v_0}} = f_2(v_1, v_2, \ldots, v_n)$$

For $f(v_0, v_1, \ldots, v_n)$ to be tautological, both $f(v_0, v_1, \ldots, v_n)_{v_0}$ and $f(v_0, v_1, \ldots, v_n)_{\overline{v_0}}$ must be tautological.

$$f_2(v_1, v_2, \ldots, v_n) \leq f_1(v_1, v_2, \ldots, v_n) + f_2(v_1, v_2, \ldots, v_n)$$

...therefore, we need only evaluate $f_2(v_1, v_2, \ldots, v_n)$, cutting the problem size in half

In summary, we cofactor a set of cubes by another cube. If the resulting cover is tautological, then the set of cubes covers the cube. We need to do the tautology check but this could be slow. We use two tricks to accelerate it. If a cover is unate, we check for a tautological cube. If we find one, it's a tautology. If we don't find one, it is not a tautology. If the cover isn't unate, but it is unate in one or more variables, we decompose on that variable. This way, we only have to check one half of the decomposition. If there are no unate variables in the cover, we decompose on another variable. We may need to check both halves of the decomposition. However, if we hit a leaf that is not tautological, we can stop because we know the whole cover cannot be tautological.

I hope it's really clear now. *R.D.*