EECS 281: **Test 2 (4 pages)** **Due: Tuesday, October 12, 2004**

Name: ______________________ Email: ______________________ Grade: _______ (100 points max)

1. (16 points) Using C++ data types for a **machine that uses a char of 8-bits**, convert the following into two's complement big-endian binary and if not, then show why not?:

Give unsigned char range:	0 to 255
Give signed char range:	−128 to +127
unsigned char x = 'C';	01000011
unsigned char x = 0xff;	11111111
unsigned char x = 129;	10000001
signed char x = −'C';	10111101
signed char x = −45;	11010011
signed char x = −129;	Range of signed char is −128 to +127, hence −129 is out of range

2. (16 points) Using C++ data types for a **machine that uses a char of 10-bits**, convert the following into two's complement big-endian binary and if not, then show why not?:

Give unsigned char range:	0 to 1023
Give signed char range:	−512 to +511
unsigned char x = 'C';	0001000011
unsigned char x = 0xff;	0011111111
unsigned char x = 129;	0010000001
signed char x = −'C';	1110111101
signed char x = −45;	1111010011
signed char x = −129;	1101111111

3. (5 points) Using C++ operator precedence, **add** the correct parenthesis (signed int a, b, c, d, e, w):

w = a * b + c | d ;	w = ((a * b) + c) | d ;
w = a & b | c + d * e;	w = ((a & b) | (c + (d * e)));

4. (5 points) Using C++ operator precedence, **remove** as many as possible parenthesis without changing the meaning:

w = ((a + b) * c);	w = (a + b) * c
= ((a * b) & (c | d));	w = a * b & (c | d)

5. (20 points) Using C++ convert the following into two's complement big-endian binary **that machine that uses a char of 4-bits**, where unsigned char u, a=0x4, b=0x7, c=0xf; signed char s, w=0x4, x=0x7, y=−1; For addition and subtraction indicate if overflow and/or carry has occurred. Show work.

u = (∼ b)+1;	u = 1000 + 1 = 1001
u = a & b;	u = 0100 & 0111 = 0100 (Bitwise AND)
u = a ˆ b;	u = 0100 ˆ 0111 = 0011 (Exclusive OR)
u = a + b;	u = 0100 + 0111 = 1011 (Addition, There is no overflow since both numbers are unsigned)
u = c >> 5;	u = 1111 >>5 = 0000 (Overflow during the fifth shift)
s = − x;	x = 0111, -x is going to be -7. s = 1001
s = w | x;	s = 0100 | 0111 = 0111
s = w + x;	s = 0100 + 0111. There is an overflow causing wrong results.
s = w − x;	s = 0100 − 0111 = -3 = 1101
s = y >> 5;	s = −1 = 1111. s >> 5 = 1111,overflow during the fifth shift. s is signed number

6. (5 points) Convert the 24-bit number 0x100457 to mime base64: _E_A_R_X______________

0x100457 = 0001 0000 0000 0100 0101 0111. For base64 conversion, 6 bits have to be combined at once.

Hence 0x100457 = 000100 000000 010001 010111 = 4 0 17 23

From the table 4 = E , 0 = A, 17 = R, 23 = X.

7. (5 points) Write a "single" C code statement of setting both bit d_3 and bit d_1 to 0 in the variable char a and all other bits unchanged. (Note: a big endian bit position of char is $d_7d_6d_5d_4d_3d_2d_1d_0$)

A quick solution is : "a = a & ¬(1<<3) & ¬(1<<1);" and the code could be cleaned up as follows: "a &= ¬(1<<3) & ¬(1<<1);" ⇒ "a &= 11110111_2 & 11111101_2 ;" ⇒ "a &= 11110101_2;" ⇒ "a &= 0xF5;"

8. (5 points) Write the "best" single C code statement of setting both bit d_3 and bit d_1 to 0 in the variable char a and all other bits unchanged. (Hint: a ?= 0x??;)

"a &= 0xF5;"

9. (10 points) Write the C code function to return 1 if an integer if odd parity and 0 otherwise: unsigned int odd(unsigned int a); (note: multiply and divide not allowed). Example: odd(0x1a) is 1.

<u>Best Code:</u>

int odd(unsigned int a) { return bcount(a) & 1; }

bcount(a) is from problem 8 of Homework 4 Solutions

10. (13 points) Give the n-cube, k-map, and SOP of the f(a,b,c) minterms for (3, 5, 6). Can this function be further minimized?

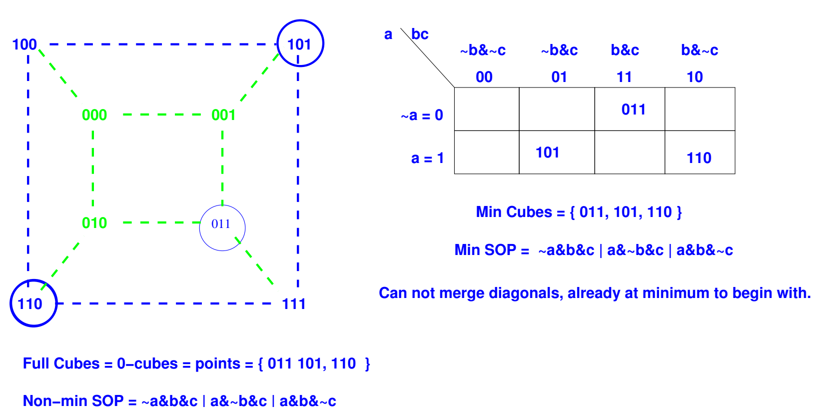

Figure 1: f(a,b,c) minterms for (3,5,6)

x1. (extra credit, 5 points) Minimize the f(a,b,c) minterms for (0,1,2,3). Show k-map, coverings on the k-map, and give minimized SOP.

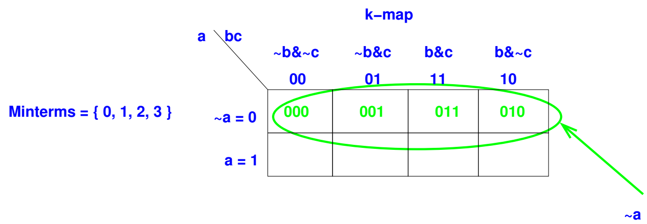

Figure 2: f(a,b,c) minterms for (3,5,6)

x2. (extra credit, 5 points) Minimize the f(a,b,c) minterms for (0,1,2,3) and a Don't Care minterm of (4,5,6,7). Show k-map, coverings on the k-map, and give minimized SOP.

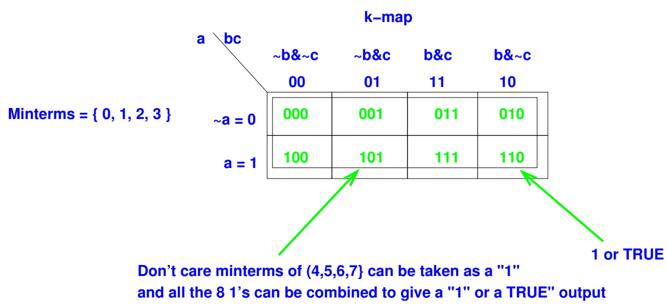

Figure 3: f(a,b,c) minterms for (3,5,6)

x3. (extra credit, 5 points) Minimize the f(a,b,c,d) minterms for (0,1,2,3) and a Don't Care minterm of (4,5,6,7). Show k-map, coverings on the k-map, and give minimized SOP.

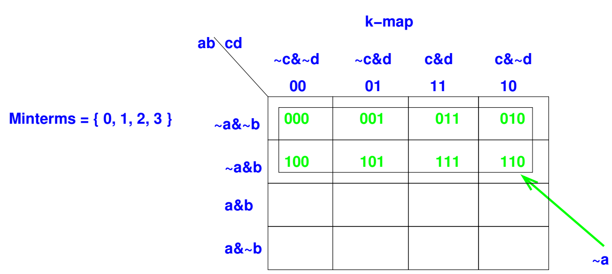

Figure 4: f(a,b,c) minterms for (3,5,6)