

CSE 140 Homework Two

July 7, 2009

*Only **Problem Set Part B** will be graded. Turn in only **Problem Set Part B** which will be due on July 20, 2009 (Monday) at 3:00pm.*

1 Problem Set Part A

- textbook 2.24
- textbook 2.25
- textbook 2.29
- textbook 3.1(a)(d)
- textbook 3.3
- textbook 3.6(a)(c)
- textbook 3.9
- textbook 3.10(c)(d)
- textbook 3.12(a)(d)
- textbook 3.14(a)(b)
- textbook 4.4(d)
- Using the map method, determine the prime implicants of 4.4(d)
- textbook 4.5(a)(d)
- textbook 4.6(a)(d)
- textbook 4.7(b)
- textbook 4.9(b)
- textbook 4.17(a)
- textbook 4.18

2 Problem Set Part B

1 (Hamming Code)

A 7-bit Hamming code consists of 3 parity bits and 4 data bits, as defined in the table below. These 3 parity bits are introduced so as to achieve the capability of correcting any single bit error, regardless of whether the error occurs in the data or in the parity bits.

Position	7	6	5	4	3	2	1
Bit	d_3	d_2	d_1	p_2	d_0	p_1	p_0

$$p_2 = d_3 \oplus d_2 \oplus d_1$$

$$p_1 = d_3 \oplus d_2 \oplus d_0$$

$$p_0 = d_3 \oplus d_1 \oplus d_0$$

(Part A) The three parity equations present a way of generating the parity bits out of the data bits. Conversely, the data bits can be generated based on the values of the parity bits. Well, almost; since there are 4 data bits (to specify 16 distinct values) and only 3 parity bits, it takes a little bit more than the parity bits only to get the data bits. The relationship can be formalized into 4 extra *xor* equations derived from the three parity equations above. We have presented below an incompletely specified version of these 4 equations. Your job is to complete each equation by figuring out the missing variable (either a parity bit or a data bit) marked with the “___” symbol. The first equation has already been completed for you as an example.

$$d_3 = \underline{p_0} \oplus p_1 \oplus p_2$$

$$d_2 = \underline{\quad} \oplus p_0 \oplus d_0$$

$$d_1 = \underline{\quad} \oplus p_0 \oplus d_2$$

$$d_0 = \underline{\quad} \oplus p_1 \oplus p_2$$

(Part B) Assume that **no errors** can occur in the **parity bits**. Under this assumption we are trying to minimize the number of parity bits in the code, yet we still want to be able to correct any single bit error in the **data bits**. How many parity bits do we need at a minimum? Please provide a brief explanation.

(Part C) Normally 3 parity bits in a 7-bit Hamming code do not provide the ability to correct more than one errors. However, if we are able to get extra information regarding the number of errors and the possible positions where an error can occur, we may be able to correct more errors.

Assume we know for sure that **no errors** can occur in the **parity bits**, while exactly **3 errors** exist in the **data bits**. Under this assumption, do you think it is possible to correct all these 3 errors? If you think so, please briefly explain your error correction approach. Otherwise, please show an example wherein **3 errors** in data bits can not be corrected. (*Hint: you can think about the idea of flipping all the data bits.*)

(Part D) Assume we know for sure that **no errors** can occur in the **parity bits**, while exactly **2 errors** exist in the **data bits**. Under this assumption, do you think it is possible to correct all these 2 errors? Please perform the same analysis as in (Part C). (*Hint: The equations you have completed in (Part A) indicate that the value of d_3 can be determined exactly by the three parity bits. This correctness of d_3 subsequently enables you to perform error correction analysis on the other 3 data bits.*)

2 (Boolean Algebra)

You are given a computation structure over the set of elements $\{A, B, C, D\}$ with the two operators $\ominus$ and $\otimes$ defined by the following tables.

$\ominus$	A	B	C	D
A	A	B	A	B
B	B	B	B	B
C	A	B	C	D
D	B	B	D	D

$\otimes$	A	B	C	D
A	A	A	C	C
B	A	B	C	D
C	C	C	C	C
D	C	D	C	D

In general, a computation structure is a **Boolean Algebra** iff all the six Huntington axioms hold. For the given computation structure, we have already been told that the *Distributivity Property* holds. Furthermore, the *Cardinality Bound* property is trivial to verify and it obviously holds here since we have four elements. So the question of whether this computation structure is a Boolean algebra or not hinges on whether the other four axioms hold for this computation structure. Could you please check and let us know whether this computation structure is a **Boolean Algebra** by checking the remaining four Huntington axioms? Please also provide a brief reasoning for your decision for each of the four axioms.

Your justification:

Axiom 1 (Closure) holds; **True / False**

Reasoning:

Axiom 2 (Identity) holds; **True / False**

Reasoning:

Axiom 3 (Commutativity) holds; **True / False**

Reasoning:

Axiom 5 (Complement) holds; **True / False**

Reasoning:

The given computation structure is a Boolean Algebra. **True / False**

3 (Karnaugh Maps)

It is possible to use a Karnaugh map in order to find a three level implementation of a boolean function. Thus, rather than constraining ourselves to the simple two level implementation of the Sum-of-Products or the Product-of-Sums form, we may derive extra levels, thus generating factored representations. As an example, look at the following Karnaugh map.

xy \ zw	00	01	11	10
00	0	0	1	0
01	1	1	1	1
11	1	1	1	1
10	0	1	1	1

=

Figure 1: Function $f = y + zw + xw + xz$

xy \ zw	00	01	11	10
00	0	0	1	0
01	1	1	1	1
11	1	1	1	1
10	0	*	1	*

Figure 2: $y + zw$

xy \ zw	00	01	11	10
00	0	0	*	0
01	*	*	*	*
11	*	*	*	*
10	0	1	*	1

Figure 3: $(x+y)(z+w)$

Thus, we can circle the implicants and arrive at the factored solution:
 $F = y + zw + (x+y)(z+w)$

We are giving you the following Karnaugh map. The same process as described in the previous example can be used in the dual sense to find the prime implicants of the first Karnaugh map and then the implicants of the residual Karnaugh map. Please use this process in order to derive a factored form that matches the logic template given in the next picture and complete the schematic by marking the correct inputs. (Assume inverted inputs exist.)

ab \ cd	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	0	1
10	1	1	0	0

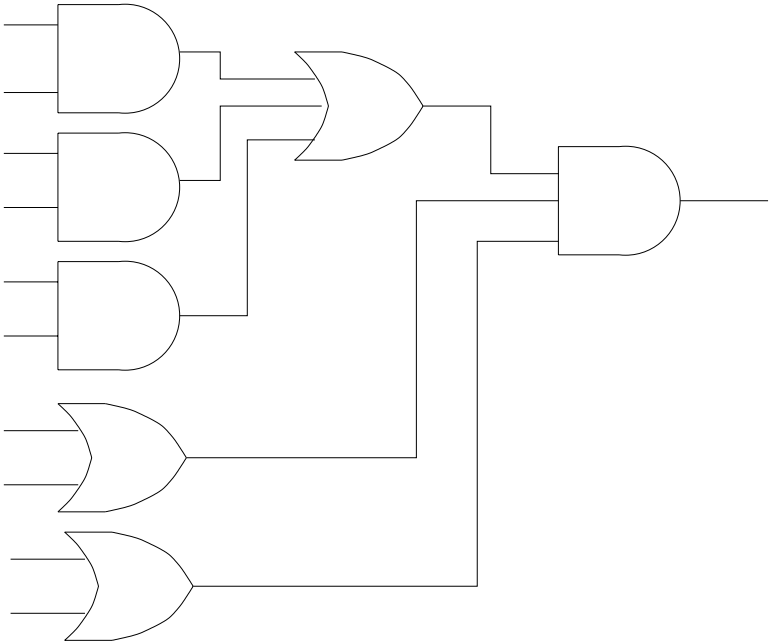
=

ab \ cd	00	01	11	10
00				
01				
11				
10				

*

ab \ cd	00	01	11	10
00				
01				
11				
10				

=



4 (Tabulation method and Quine-McCluskey algorithm)

(Part A) The Tabulation method is applied to the minterms of a 4-variable Boolean function. You know that the *minimum distance* between any two minterms in this function is *two* and there are two elements in the G_3 group in the list of 0-subcubes. At most how many elements can be in the G_2 group of 0-subcubes in this case?

(Part B) The Tabulation method is applied to the minterms of a 4-variable Boolean function. The minterm distribution of this function in the list of 0-subcubes is the following: 1 element in the G_0 group, 2 elements in the G_1 group, 2 elements in the G_2 group, 2 elements in the G_3 group, and 1 element in the G_4 group. What is the **minimum possible gate count** of the sum-of-products implementation of this function? Specify the type and the number of inputs for each gate in this minimum implementation for full credit.

(Part C) The Tabulation method is applied to the minterms of a 4-variable Boolean function. If in the list of 0-subcubes the G_2 group is empty, then at most how many cube merging **iterations** are needed to generate all prime cubes? (A cube merging *iteration* denotes the **entire process** of merging the list of $(i-1)$ -subcubes to generate the complete list of i -subcubes.)

(Part D) Three **initial** (i.e., no essentiality, row/column dominance operations whatsoever have yet taken place) Quine-McCluskey tables of **4-variable** functions are shown below, within which each column represents an unspecified minterm to be covered and each row denotes a prime implicant. For each table, identify whether it can represent a legal Quine-McCluskey table or not. If your answer is **Yes**, please provide one possible Karnaugh map for this set of minterms and implicants. If your answer is **No**, specify the reason for your answer for full credit.

Q-M Table 1

P_1	x	x		x	x					
P_2		x	x		x	x				
P_3			x			x		x		x
P_4							x	x	x	x

Q-M Table 2

P_1		x		x	x	x
P_2	x		x			
P_3			x	x		
P_4		x			x	

Q-M Table 3

P_1	x	x		x	x					
P_2		x	x		x	x				
P_3				x		x	x			
P_4			x			x		x		x
P_5							x	x	x	x