

CSE 140 Homework One

January 12, 2014

*Only **Problem Set Part B** will be graded. Turn in only **Problem Set Part B** which will be due on January 28, 2014 (Tuesday) at 4:00pm.*

1 Problem Set Part A

- Roth&Kinney, 6th Ed: 1.1
- Roth&Kinney, 6th Ed: 1.3
- Roth&Kinney, 6th Ed: 1.4
- Roth&Kinney, 6th Ed: 1.5
- Roth&Kinney, 6th Ed: 1.7
- Roth&Kinney, 6th Ed: 1.9
- Roth&Kinney, 6th Ed: 1.15
- Roth&Kinney, 6th Ed: 1.19
- Roth&Kinney, 6th Ed: 1.29
- Roth&Kinney, 6th Ed: 1.31
- Roth&Kinney, 6th Ed: 2.1
- Roth&Kinney, 6th Ed: 2.3
- Roth&Kinney, 6th Ed: 2.5
- Roth&Kinney, 6th Ed: 2.6
- Roth&Kinney, 6th Ed: 2.7

1 (Two's Complement Multiplication)

This question concerns Robertson's and Booth's multiplication algorithms for 2's complement numbers. In both cases, the multiplicand and the multiplier are n bits, while the product is $(2n - 1)$ bits. In review, in both multiplication algorithms, the multiplier is checked bit by bit starting from the least significant bit. The whole multiplication process is composed of n steps; the partial sum is initialized to 0 and takes up $(n + i - 1)$ bits space at the beginning of the i^{th} step ($i = 1, 2, \dots, n$). According to the multiplication algorithm, the multiplicand may be added to (or subtracted from) the n most significant bits of the partial sum, and the resultant partial sum is then shifted 1-bit to the right. All addition and subtraction operations are performed on an n bit adder.

(Part A) You are provided with the following pairs of 8-bit 2's complement numbers, for which multiplication is to be performed using both *Robertson's* and *Booth's* algorithms. For each pair, please report the number of operations (i.e., addition/subtraction operations) performed by each algorithm in the table.

Pair	Multiplicand	Multiplier	Robertson's # OPs	Booth's # OPs
1	00111001	00011100		
2	11111001	01101111		
3	00010110	11001101		
4	11010110	10100110		

The following table lists several pairs of **partially specified** 2's complement numbers (with the x 's representing unknown values) that are to be considered in the subsequent parts of this question.

Pair	Multiplicand	Multiplier
1	110000xx	1101x011
2	101011xx	0110x110
3	001111xx	1111x000
4	010001xx	1010x110
5	010110xx	0000x111

(Part B) In Robertson's algorithm, an **A/S** signal is used by the adder/subtractor to select between addition ($A/S = 0$) and subtraction ($A/S = 1$). Yet there is a defect in this A/S signal that makes it always stuck at **0**. Which of the pairs listed above will be impacted by this defect and end up generating an incorrect product as a result? Please clearly list all possible pairs and provide a brief reasoning.

(Part C) Booth's algorithm also uses an **A/S** signal to select between addition ($A/S = 0$) and subtraction ($A/S = 1$). This time, a defect makes this signal always stuck at **1**. Which of the pairs listed above will be impacted by this defect and end up generating an incorrect product as a result? Please clearly list all possible pairs and provide a brief reasoning.

For your convenience, here we duplicate the table shown on the previous page.

Pair	Multiplicand	Multiplier
1	110000xx	1101x011
2	101011xx	0110x110
3	001111xx	1111x000
4	010001xx	1010x110
5	010110xx	0000x111

(Part D) Assume that in Robertson's algorithm, during the right shift process, *sign extension* is performed on the partial sum accumulation such that the value to be shifted in is the most significant bit of the partial sum. Which of the pairs listed above will display a correct product and which ones an incorrect product at the conclusion of the multiplication process? Please clearly list all possible pairs and provide a brief reasoning.

(Part E) Assume that *sign extension* is performed during the right shift process in Booth's algorithm. Which of the pairs listed above will display a correct product and which ones an incorrect product at the conclusion of the multiplication process? Please clearly list all possible pairs and provide a brief reasoning.

2 (Error Detection and Correction)

You are given a 7-bit code consisting of 4 **base-3** data bits and 3 additional **base-3** parity bits added for error correction and detection capability.

Position	7	6	5	4	3	2	1
Bit	m_0	m_1	m_2	p_0	m_3	p_1	p_2

In this code, parity equations include the same data bits as the Hamming Code we discussed in class but the operation performed is the **modulo-3** of the sum of selected data bits as seen below. An example is provided in the following table where you can see the parity values for the given data values.

$$\begin{aligned}p_0 &= (m_0 + m_1 + m_2) \bmod 3 \\p_1 &= (m_0 + m_1 + m_3) \bmod 3 \\p_2 &= (m_0 + m_2 + m_3) \bmod 3\end{aligned}$$

Bit	m_0	m_1	m_2	p_0	m_3	p_1	p_2
Values	1	2	0	0	2	2	0

(Part A) First, we would like to assess the error detection capabilities of this code. Please specify whether this code can detect 1, 2 and 3 errors. For each of the specified number of errors, if they can be consistently detected, provide a brief reasoning in support of your answer. If it cannot be detected, please specify a failure combination that cannot be detected. Remember that each bit can be 0, 1 or 2 in the base-3 number system. When there is an error in a particular bit, it can change to any of the other two possible values in the base-3 number system. For example, if a bit is 1, it can erroneously change to 0 or 2. (*Hint*: To detect the error, at least one of the equations should fail when there is an error.)

1-bit error detection:

2-bit error detection:

3-bit error detection:

(Part B) Can this code identify the location of a 1-bit error? If it can, provide a brief reasoning and also *describe how we can correct the error if it is possible*. If it cannot, specify a pair of bits that cannot be distinguished from each other when there is an error in one of them. (*Hint*: To identify the error location, each error should generate a unique combination of parity equation failures.)

3 (Boolean Axioms)

The following incomplete truth tables show the 2 input $*$ and $+$ operations for a possible 3-value Boolean algebra:

$*$	0	A	1
0			
A			
1			

$+$	0	A	1
0			
A			
1			

For the following questions, you may find the Boolean Axioms on the back of the exam to be useful.

(Part A) How many 2-input 3-value functions are commutative?

(Part B) Of the commutative 2-input 3-value functions, how many of them have an identity element?

(Part C) Using what you know about the identity and commutativity axioms for Boolean algebra, as well as the fact that 0 and 1 are the identity elements for $+$ and $*$ respectively, fill in the above 2 tables.

(Part D) What is the number of valid 3-valued Boolean algebras? Please explain your reasoning (e.g. justify your answer by addressing the Boolean axioms).