Module 1:
Computer Arithmetic and Logic

"Once your soul has been enlarged by a truth, it can never return to its original size."
-Blaise Pascal

Notes By

Giorgio Grigolo

Student of Mathematics and Computer Science
St.Aloysius College

2020

Contents

Part I

Computer Arithmetic and Logic

Chapter 1

Computer Arithmetic

1.1 Binary numbers

There are two types of binary numbers:

- Signed binary numbers

  This type of number hold a sign of magnitude.

- Unsigned binary numbers

  This type of number is regarded as positive.

1.1.1 Unsigned binary numbers

MSB							LSB
128	64	32	16	8	4	2	1
1	0	0	0	0	1	0	0

$$128 + 4 = 132$$

MSB							LSB
128	64	32	16	8	4	2	1
0	1	0	0	0	0	1	1

$$64 + 2 + 1 = 67$$

1.1.2 Signed binary numbers

- Method 1: Sign and magnitude

  The MSB holds a $\pm$ sign to determine the entire number's sign. If set to 0, it is positive, otherwise, negative.

  Consider an 8-bit register:

MSB							LSB
$\pm$	64	32	16	8	4	2	1
0	0	1	0	1	0	0	0

$$= +40$$

- Method 2: Two's Complement

  The MSB now holds its usual value (in case of an 8-bit register, 128) with the prefixion of a negative sign.

MSB							LSB
-128	64	32	16	8	4	2	1
1	0	0	0	0	0	0	1

$$= -128 + 1 = -127$$

Range: -128 – 127

1.2 Binary fractions

- Fixed point binary numbers

Unsigned:

Consider a 6-bit register:

8	4	2	1	.	$\frac{1}{2}$	$\frac{1}{4}$
1	0	0	1	.	1	1

$$= 8 + 1 + \frac{2}{4} + \frac{1}{4}$$

$$= 9\frac{3}{4}$$

Signed:

Method 1: Sign and magnitude

$\pm$	16	8	4	2	1	.	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	
0	0	1	1	0	0	.	1	1	0	0	*a)*
1	0	1	1	1	0	.	1	0	1	1	*b)*

$$a) = 8 + 4 + \frac{8}{16} + \frac{4}{16} = 12\frac{3}{4}$$

$$b) = -\left(8 + 4 + 2 + \frac{8}{16} + \frac{2}{16} + \frac{1}{16}\right) = -14\frac{1}{16}$$

Method 2: Two's Complement

-32	16	8	4	2	1	.	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	
0	1	1	0	0	0	.	1	1	0	0	*a)*
1	0	0	0	0	1	.	1	0	0	0	*b)*

$$a) = 16 + 8 + \frac{8}{16} + \frac{2}{4} + \frac{1}{4} = 24\frac{3}{4}$$

$$b) = -32 + 1 + \frac{1}{2} = -30\frac{1}{2}$$

Range

-32	16	8	4	2	1	.	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	
0	1	1	1	1	1	.	1	1	1	1	*a)*
0	0	0	0	0	0	.	0	0	0	1	*b)*
1	0	0	0	0	0	.	0	0	0	0	*c)*
1	1	1	1	1	1	.	1	1	1	1	*d)*

$$\text{a) Largest +ve number} = 31\frac{15}{16}$$

$$\text{b) Smallest +ve number} = \frac{1}{16}$$

$$\text{c) Largest magnitude -ve number} = -32$$

$$\text{d) Smallest magnitude -ve number} = -\frac{1}{16}$$

$$\therefore R = -32 \leq x \leq 31\frac{15}{16}$$

1.3 Floating point numbers

Being the binary equivalent of decimal standard form, it increases the range of a given register. Considering a 12-bit register it is usually represented as:

1	.	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	$\frac{1}{64}$	$\frac{1}{128}$	8	4	2	1

- Method 1: Sign and magnitude

  The MSB holds a ± sign to determine the entire number's sign. If set to 0, it is positive, otherwise, negative.

  Consider an 8-bit register:

±	.	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	$\frac{1}{64}$	$\frac{1}{128}$	8	4	2	1	
0	.	1	1	0	0	0	0	0	0	0	0	1	*a)*
1	.	1	1	0	0	0	0	0	1	0	0	1	*b)*

$$\text{a)} = \frac{3}{4} \times 2^{1}$$

$$\text{b)} = \frac{-3}{4} \times 2^{-1} = \frac{-3}{8}$$

- Method 2: Two's Complement

  Consider an 8-bit mantissa & a 4-bit exponent

1	.	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	$\frac{1}{64}$	$\frac{1}{128}$	8	4	2	1	
0	.	1	0	0	0	0	0	0	0	0	0	1	*a)*
0	.	0	1	0	0	0	0	0	0	0	1	0	*b)*
0	.	0	0	1	0	0	0	0	0	0	1	1	*c)*

$$a) = \frac{1}{2} \times 2^{1}$$

$$b) = \frac{1}{4} \times 2^{2}$$

$$c) = \frac{1}{8} \times 2^{3}$$

Above are represented three different binary values, however, when computed, return the same value: 1. To solve this problem, the process of normalization is applied to the number.

Normalization

For a number to be considered normalized, the binary point should be:

- between a 0 and a 1 for positive numbers
- between a 1 and a 0 for negative numbers

Example 1 *Convert* $\frac{1}{8}$ *to a normalized floating point binary representation.*

Step 1: Find fixed point representation

8	4	2	1	.	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$
0	0	0	0	.	0	0	1	0

For any number ≥ 1 float point to $\curvearrowright$
and any number ≤ 1 float point to $\curvearrowleft$

Step 2: Float the point

Since we moved the point towards the right, the number needs to be multiplied by
a number < 1 thus exponent needs to be set to 2^{-2}.

$$= \frac{1}{2} \times 2^{-2} = \frac{1}{8}$$

The result above represents $\frac{1}{8}$ in normalized binary fraction form.

Example 2 *Convert* $34\frac{3}{8}$ *to a normalized floating point binary representation*

Step 1: Find fixed point representation

-64	32	16	8	4	2	1	.	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$
0	1	0	0	0	1	0	.	0	1	1

$$= 34\frac{3}{8}$$

Step 2: Float the point

To achieve an arithmetic operation of 2^{-6} we move the point $\curvearrowleft$ 6 times and apply an exponent of 6

1	.	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	$\frac{1}{64}$	$\frac{1}{128}$	$\frac{1}{256}$	$\frac{1}{512}$	-8	4	2	1
0	.	1	0	0	0	1	0	0	1	1	0	1	1	0

$$= 34\frac{3}{8} \times 2^{-6} \times 2^{6}$$

The operation of $\times 2^{-6}$ is not performed arithmetically but visually by moving the point forwards, then reversed through the exponent part of the register.

Example 3 *Convert* -0.5625 *to a normalized floating point binary representation.*

Step 1: Find fixed point representation

-2	1	.	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$
1	1		0	1	1	1

$$-0.5625 = \frac{5625}{10000} = \frac{-9}{16}$$

$$\frac{-9}{16} = -1\frac{7}{16}$$

Step 2: Float the point

-1	.	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	-8	4	2	1
1	.	1	0	1	1	1	0	0	0	1

$$= -1\frac{7}{16} \times 2^{-1} \times 2^{1}$$

Range

-1	.	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	$\frac{1}{64}$	$\frac{1}{128}$	-8	4	2	1	
0	.	1	1	1	1	1	1	1	0	1	1	1	*a)*
0	.	1	0	0	0	0	0	0	1	0	0	0	*b)*
1		0	0	0	0	0	0	0	0	1	1	1	*c)*
1	.	0	1	1	1	1	1	1	1	0	0	0	*d)*

$$a = \text{largest +ve number} = 127_{10}$$

$$b = \text{smallest +ve number} = \frac{1}{152_{10}}$$

$$c = \text{smallest -ve number} = -128_{10}$$

$$d = \text{largest -ve number} = \frac{-65}{2^{16}{}_{10}}$$

$$\therefore \quad R: \quad -128_{10} \leq 127_{10}$$

Note that the range can be further extended by enlarging the exponent part of the binary number.

1.4 Hexadecimal numbers

1.4.1 Hexadecimal Addition

To perform hexadecimal addition, two digits are counted ($(A+F)_{16} = 25_{10}$) , then find the difference of that result and 16 ($25 - 16 = 9$). That number is to be written down in the respective resultant digit and a carry-over of 1 is added to the next digit for as many times as 16 goes into the initial sum. The process is repeated For as much as needed.

Exercise 1 *Perform hexadecimal addition on these numbers.*

$$\begin{array}{r} EFF \\ +\ ABC \\ \hline 19BB \end{array} \qquad \begin{array}{r} 2BC \\ +\ AF \\ \hline 36B \end{array}$$

$$\begin{array}{r} CFD1 \\ +\ ACD \\ \hline DA9E \end{array} \qquad \begin{array}{r} 2FCF \\ +\ AFE \\ \hline 3ACD \end{array}$$

1.4.2 Hexadecimal Subtraction

To perform hexadecimal subtraction, two digits are subtracted only if top one is heavy. If not, 1 must be removed from the previous digit and 16 must be added to the required digit. Consequently, normal subtraction occurs between the two initial digits.

Exercise 2 *Perform hexadecimal subtraction on these numbers.*

$$\begin{array}{r} EFF \\ -\ ABC \\ \hline 343 \end{array} \qquad \begin{array}{r} FE1 \\ -\ BA \\ \hline F36 \end{array}$$

$$\begin{array}{r} \overset{9}{\bar{A}}\,\overset{A^{16}}{\bar{B}}\,\overset{16}{E} \\ -\ FF \\ \hline 9BF \end{array} \qquad \begin{array}{r} AB23 \\ -\ FED \\ \hline 9B36 \end{array}$$

1.5 Binary Coded Decimals

1.6 Gray Code

Dec	Binary				Gray Code			
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0				
5	0	1	0	1				
6	0	1	1	0				
7	0	1	1	1				
8	1	0	0	0				
9	1	0	0	1				
A	1	0	1	0				
B	1	0	1	1				
C	1	1	0	0				
D	1	1	0	1				
E	1	1	1	0				
F	1	1	1	1				

1.7 Errors in Computer Arithmetic

- Precision Error:

  This error is associated to word length. If a number of 9-bits is to be represented in an 8-bit register, an overflow error would arise and therefore an imperfect representation is created.

- Accuracy Error:

  This error is associated to the closeness of the approximation of an exact value. This entails to the note that accuracy and precision are not the same thing. As an example: an accuracy error arises when the fraction $\frac{1}{3}$ is represented in an n-bit register since 3 is not a power of 2.

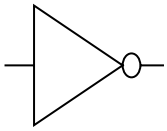
Chapter 2

Computer Logic

2.1 Logic Gates

2.1.1 Basic Logic Gates

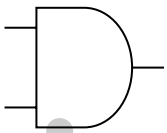
1. NOT Gate



A	B
0	1
1	0

$\boxed{B = \bar{A}}$

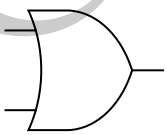
2. AND Gate



A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

$\boxed{B = A.B}$

3. OR Gate

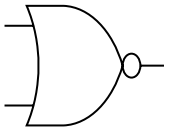


A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

$\boxed{C = A + B}$

2.1.2 Advanced Logic Gates

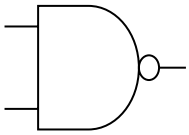
1. NOR Gate



A	B	C
0	0	1
0	1	0
1	0	0
1	1	0

$$\boxed{B = \overline{A + B}}$$

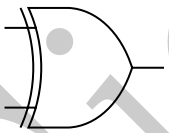
2. NAND Gate



A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

$$\boxed{B = \overline{A.B}}$$

3. XOR Gate



A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

$$\boxed{C = A \oplus B}$$

2.2 Laws of Boolean Algebra

- Associative Law

$$(A + B) + C = A + (B + C)$$
$$(A \cdot B) \cdot C = A \cdot (B \cdot C)$$

- Commutative Law

$$A + B = B + A$$
$$A \cdot B = B \cdot A$$

- Distributive Law

$$A + (B \cdot C) = (A \cdot B) + (A \cdot C)$$
$$A \cdot (B + C) = (A + B) \cdot (A + C)$$

- Tautology Law

$$A + A = A \qquad A + \bar{A} = 1$$
$$A \cdot A = A \qquad A \cdot \bar{A} = \emptyset$$

- Common Sense Law

$$A + 1 = 1 \qquad A + \emptyset = A$$

$$A \cdot 1 = A \qquad A \cdot \emptyset = \emptyset$$

- Absorption Law

$$A + (A \cdot B) = A$$

$$A \cdot (A \cdot B) = A$$

- Double Complement Law

$$\bar{\bar{A}} = A$$

$$\overline{\bar{A} \cdot \bar{B}} \neq A \cdot B \quad ; \quad \overline{(\overline{A \cdot B})} = A \cdot B$$

- De Morgan's Law

$$\bar{A} + \bar{B} = \overline{A \cdot B}$$

$$\bar{A} \cdot \bar{B} = \overline{A + B}$$

2.2.1 Exercises

Example 1 *Simplify* $\bar{A} \cdot \bar{B} + A \cdot \bar{B} + \bar{A} \cdot B$

$$\begin{aligned}
& \bar{A} \cdot \bar{B} + A \cdot B + \bar{A} \cdot B \\
= \; & \bar{B} \cdot (\bar{A} \cdot A) + (\bar{A} \cdot B) \\
= \; & (\bar{B} \cdot 1) + (\bar{A} \cdot B) \\
= \; & \bar{B} + (\bar{A} \cdot B) \\
= \; & (\bar{B} + \bar{A}) \cdot (\bar{B} + B) \\
= \; & (\bar{B} + \bar{A}) \cdot 1 \\
= \; & \overline{(A + B)}
\end{aligned}$$

Example 2 *Prove that* $A \cdot B + A \cdot \bar{B} \cdot C + A = A$

$$\begin{aligned} & A \cdot B + A \cdot \bar{B} \cdot C + A = A \\ = \; & A \cdot (B + \bar{B} \cdot C + 1) \\ = \; & A \cdot ((C \cdot 1) + 1) \\ = \; & A \cdot (C + 1) \\ = \; & A \end{aligned}$$

Example 3 *Simplify* $ABC + \bar{A}BC + A\bar{B}C + AB\bar{C}$

$$\begin{aligned} & ABC + \bar{A}BC + A\bar{B}C + AB\bar{C} \\ = \; & BC \cdot (A + \bar{A}) + A\bar{B}C + AB\bar{C} \\ = \; & B(C + A\bar{C}) + A\bar{B}C \\ = \; & B\left((A + C) \cdot (C + \bar{C})\right) + A\bar{B}C \\ = \; & B(A + C) + A\bar{B}C \\ = \; & BC + AB + A\bar{B}C \\ = \; & BC + A(B + (B \cdot C)) \\ = \; & BC + A\left((B + \bar{B}) \cdot (B + C)\right) \\ = \; & AB + AC + BC \end{aligned}$$

2.3 Applications of Logic Gates

2.3.1 BCD to Gray Code Convertor

Problem: Develop a logic circuit which takes in 4 inputs in BCD format and returns a Gray Code equivalent in 4-bits.

	BCD				Gray Code			
DEC	A	B	C	D	W	X	Y	Z
0	0	0	0	0	1	1	1	0
1	0	0	0	1	0	0	1	0
2	0	0	1	0	1	0	1	1
3	0	0	1	1	1	0	1	1
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	1	0	1
6	0	1	1	0	1	1	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	1	1	1
9	1	0	0	1	1	1	1	1
10	DON'T CARE							

$W =$

CD \ AB	00	01	11	10
00	0	0	X	1
01	0	0	X	1
11	0	0	X	X
10	0	0	X	X

$X =$

CD \ AB	00	01	11	10
00	0	1	X	1
01	0	1	X	1
11	0	1	X	X
10	0	1	X	X

$Y =$

CD \ AB	00	01	11	10
00	0	1	X	0
01	0	1	X	0
11	1	0	X	X
10	1	0	X	X

$Z =$

CD \ AB	00	01	11	10
00	0	0	X	0
01	1	1	X	1
11	0	0	X	X
10	1	1	X	X

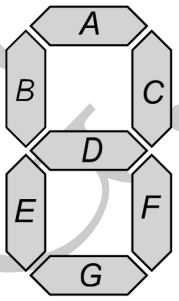
$$W = A \qquad X = A + B$$

$$Y = B \oplus C \qquad Z = C \oplus D$$

2.3.2 7-Segment LED Display

Problem: Develop a logic circuit that returns the state of the A segment inside a 1-digit 7-segment LED display.

DEC	W	X	Y	Z	A	B	C	D	E	F	G
0	0	0	0	0	1	1	1	0	1	1	1
1	0	0	0	1	0	0	1	0	0	1	0
2	0	0	1	0	1	0	1	1	1	0	1
3	0	0	1	1	1	0	1	1	0	1	1
4	0	1	0	0	0	1	1	1	0	1	0
5	0	1	0	1	1	1	0	1	0	1	1
6	0	1	1	0	1	1	0	1	1	1	1
7	0	1	1	1	1	0	1	0	0	1	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
10	DON'T CARE										
⋮											

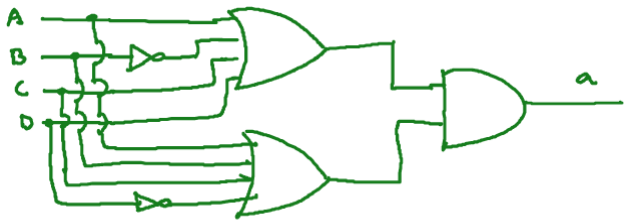


$A =$

YZ \ WX	00	01	11	10
00	1	0	X	1
01	0	1	X	1
11	1	1	X	X
10	1	1	X	X

$$A = \overline{\bar{W}X\bar{Y}\bar{Z} + \bar{W}\bar{X}\bar{Y}Z}$$

$$= (W + \bar{X} + Y + Z) \cdot (W + X + Y + \bar{Z})$$



2.3.3 Comparator

Problem: Develop a logic circuit that can be used to compare two 2-bit binary numbers x, y. The circuit should be able to detect if $x < y$, $x = y$ or $x > y$. Label your output: Less than (L), Equal (E), Greater than (G).

A	B	C	D	L	E	G
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

$x = AB, y = CD$

CD \ AB	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	0	1
10	1	1	0	0

CD \ AB	00	01	11	10
00	1	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	0	0	1

CD \ AB	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	0	0
10	0	0	1	0

$L = \bar{A}C + \bar{A}\bar{B}D + \bar{B}CD \quad E = (B \odot D) \cdot (A \odot C) \quad G = A\bar{C} + B\bar{C}\bar{D} + AB\bar{D}$

2.3.4 S&M to Two's Complement Converter

Problem: If $A_2A_1A_0$ is a 3 bit sign and magnitude binary number and $B_2B_1B_0$ is a 3 bit 2's Complement number draw the Karnaugh maps of $B_2B_1B_0$ in terms of $A_2A_1A_0$ and hence obtain the minimized boolean expression for $B_2B_1B_0$.

Decimal	S&M			2's Comp.		
	A_2	A_1	A_0	B_2	B_1	B_0
+0	0	0	0	0	0	0
1	0	0	0	0	0	1
2	0	0	1	0	1	0
3	0	0	1	0	1	1
-0	0	1	0	X	X	X
-1	1	0	1	1	1	1
-2	1	1	0	1	1	0
-3	1	1	1	1	0	1

$B_2 =$

A_0 \ A_2A_1	00	01	11	10
0	0	0	1	X
1	0	0	1	1

$B_1 =$

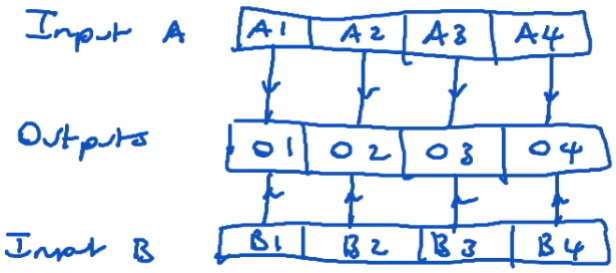
A_0 \ A_2A_1	00	01	11	10
0	0	1	1	X
1	0	1	0	1

$B_0 =$

A_0 \ A_2A_1	00	01	11	10
0	0	0	0	X
1	1	1	1	1

2.3.5 Multiplexer

Problem: If two 4-bit numbers are input to the multiplexer and a select line (to inputs A and B) can be high or low, develop a multiplexer such that the inputs A are switched on to the output when the select line is high otherwise B is switched when the select line is low.



2.3.6 2-bit Binary Adder

Consider two binary numbers X_1X_2 and Y_1Y_2. Assume they are added and the result had to be outputted from a logic circuit. Develop a logic circuit which performs this process.

X_1	X_2	Y_1	Y_2	A	B	C
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	1

$$X_1X_2Y_2+X_2Y_1Y_2+X_1Y_1$$

$$B = \bar{X}_1X_2\bar{Y}_1Y_2+X_1X_2Y_1Y_2+\bar{X}_1\bar{X}_2Y_1+Y_1\bar{Y}_2\bar{X}_1X_2+\bar{Y}_1\bar{Y}_2X_1+X_1\bar{X}_2\bar{Y}_1$$

Y_1Y_2 \ X_1X_2	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	0	1	1	1
10	0	0	1	1

$A =$

Y_1Y_2 \ X_1X_2	00	01	11	10
00	0	0	1	1
01	0	1	0	1
11	1	0	1	0
10	1	1	0	0

2.3.7 Odd Parity Check

Problem: A 3-bit binary code is to be used to transmit information from point A to point B. As a check on the information received, a parity check is to be used. If odd parity is to be used, develop a circuit which can be used

to add the parity bit at the transmitting end and a separate circuit which can be used to check parity at the receiving end.

A	B	C	P
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

2.3.8 Panel Display Fuel

Problem: A panel display has five lamps L_1, L_2, L_3, L_4, L_5 such that they indicate the level of fuel in the tank. The level-sensing circuit produces a binary number from zero to ten to represent the fuel level, where zero indicates "no fuel" and ten indicates "full tank".

Level Reading	Lamps Lit
0 to 2	L_1
3 to 4	L_1, L_2
5 to 6	L_1, L_2, L_3
7 to 8	L_1, L_2, L_3, L_4
9 to 10	L_1, L_2, L_3, L_4, L_5
1	0
1	1
1	1