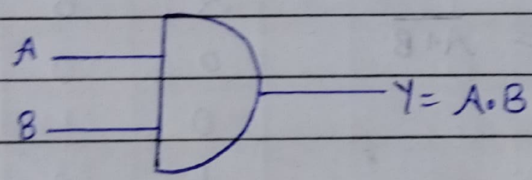
Logic Gates

There are 3 basic gates and 2 universal gates.

- Basic gates: AND, OR, NOT
- Universal gates: NAND ↓ (NOT + AND), NOR ↓ (NOT + OR)

(i) AND gate

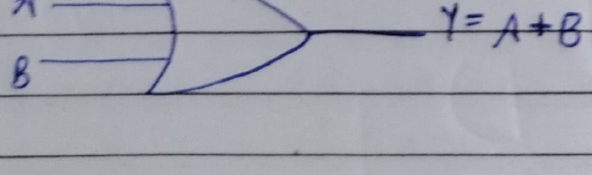
Operation is multiplication.



I/P		O/P
A	B	Y = A.B
0	0	0
0	1	0
1	0	0
1	1	1

(ii) OR gate

Operation is addition.



(I/P) A	B	(O/P) Y = A+B
0	0	0
0	1	1
1	0	1
1	1	1

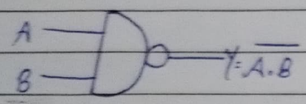
(iii) NOT gate

Operation is compliment.

I/P	O/P
1	0
0	1

(iv) NAND gate

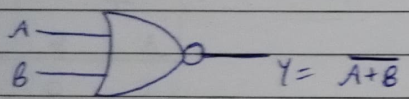
Operation is compliment and multiplication.



I/P		O/P
A	B	$Y = \overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

(v) NOR gate

Operation is compliment and addition.



I/P		O/P
A	B	$Y = \overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

Boolean Algebra (set of rules to minimize the complexity of logic circuits)

→ RULES

1. $A + 0 = A$
2. $A + 1 = 1$
3. $A \cdot 0 = 0$
4. $A \cdot 1 = 1$
5. $A + A = A$
6. $A + \bar{A} = 1$
7. $A \cdot A = A$
8. $A \cdot \bar{A} = 0$
9. $\bar{\bar{A}} = A$
10. $A + AB = A$
11. $A + \bar{A}B = A + B$
12. $(A+B)(A+C) = A + BC$

→ De Morgan's Theorem

1. $\overline{A.B} = \bar{A} + \bar{B}$
2. $\overline{A+B} = \bar{A} . \bar{B}$

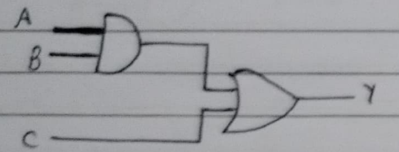
→ Laws

1. Comulative :- $A+B = B+A$; $A+B+C = A+C+B = B+A+C = C+A+B$
   $A.B = B.A$; $A.B.C = A.C.B = B.A.C = C.A.B$
2. Associative :- $(A+B)+C = A+(B+C)$
   $(A.B).C = A.(B.C)$
3. Distributive :- $A.(B+C) = A.B + A.C$
   $A + B.C = (A+B).(A+C)$

Formation of Truth Table and Boolean Equations for Simple problems.

1. Find the truth table of logic circuits.

(i)



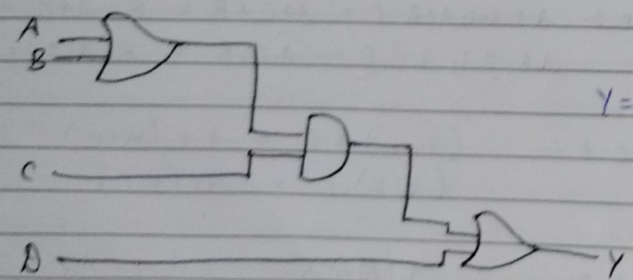
$2^3 = 08$ no. of I/P

A	B	A.B	C	Y
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	0	1

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

A	B	C	$Y = A.B + C$
0	0	0	0
0	0	1	1
0	1	0	0
1	0	0	0
0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	1

(ii)



$Y = (A+B).C + D$

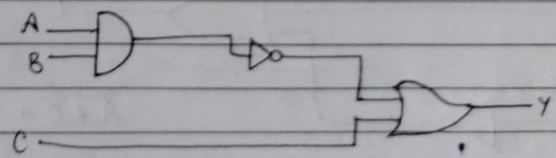
$2^4 = 16$

A	B	C	D	$Y = (A+B).C + D$
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	1	0	0	0
1	0	0	0	
1	1	0	0	
1	0	1	0	
1	0	0	1	
0	1	1	0	
0	1	0	1	
0	0	1	1	
1	1	1	0	
1	0	1	1	
0	1	1	1	
1	1	0	1	
1	1	1	1	1

Aug 8, 24

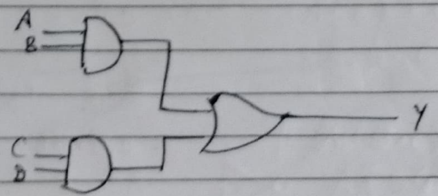
2. Find boolean eq: (expressions) for the logic diagram.

(i)



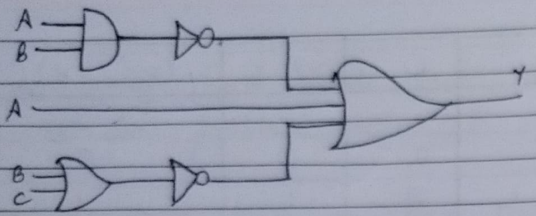
$Y = \overline{A.B} + C$

(ii)



$Y = (A.B) + (C.D)$

(iii)



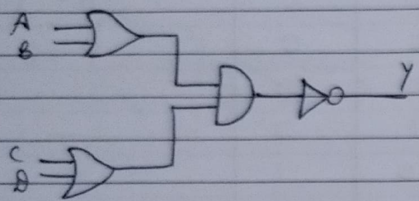
$Y = \overline{A.B} + A + \overline{B+C}$

$= \bar{A} + \bar{B} + A + \bar{B}.\bar{C}$

$= 1 + \bar{B} + \bar{B}.\bar{C}$

$= 1 + \bar{B}$

(iv)



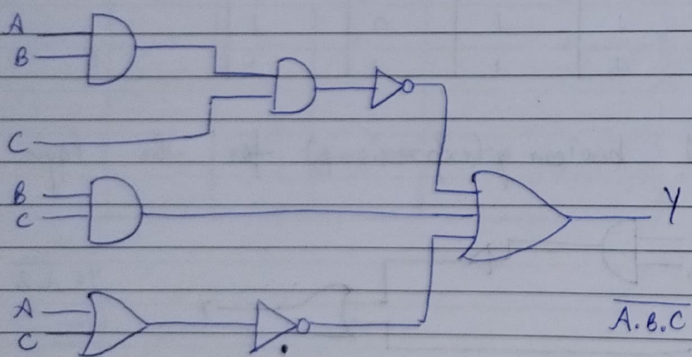
$Y = \overline{(A+B).(C+D)}$

$= \overline{(A+B)} + \overline{(C+D)}$

$= \bar{A}.\bar{B} + \bar{C}.\bar{D}$

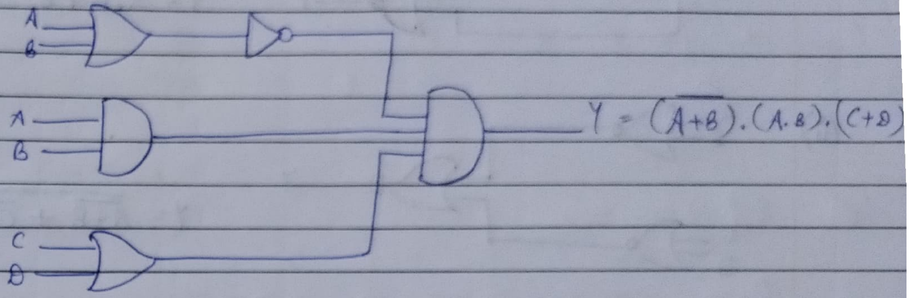
3. Implementation of boolean eq^n with gates.

(i) $\overline{A.B.C} + B.C + \overline{A+C}$

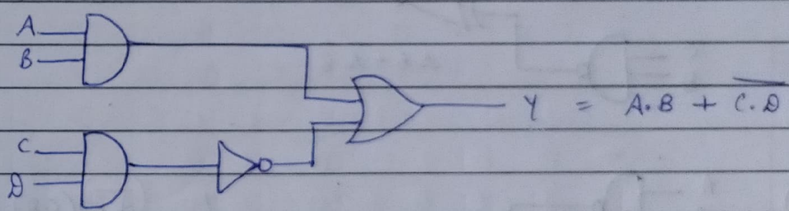


$\overline{A.B.C} + B.C + \overline{A+C}$

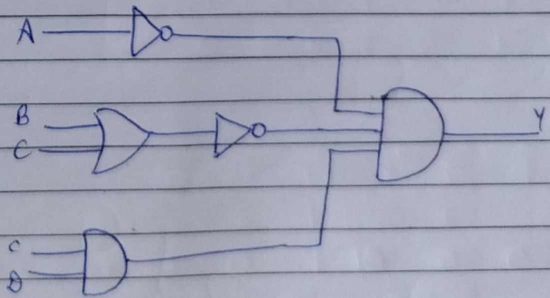
(ii) $\overline{(A+B)}.(A.B).(C+D)$



(iii) $A.B + \overline{C.D}$



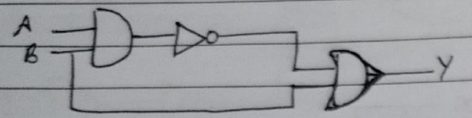
(iv) $\bar{A}.(\overline{B+C}).C.D$



$\bar{A}.(\overline{B+C}).C.D$

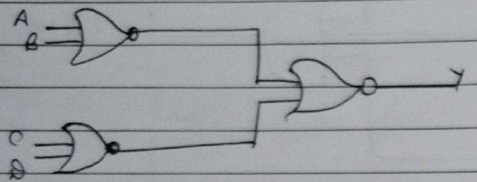
4. Write a boolean expression for logic circuits.

(i)



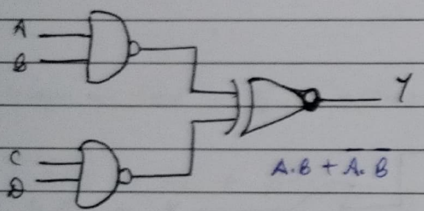
$Y = \overline{A\cdot B} + B$

(ii)



$Y = \overline{\overline{A+B} + \overline{C+D}}$

(iii)

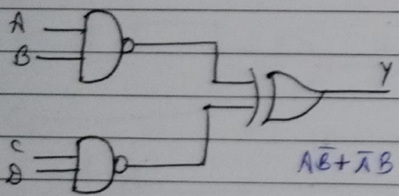


$Y = \overline{AB}\cdot\overline{CD} + \overline{\overline{A\cdot B}} + \overline{\overline{C\cdot D}}$

$= \overline{AB}\cdot\overline{CD} + AB + C\cdot D$

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

(iv)



$Y = (\overline{A\cdot B})\cdot(C\cdot D) + (A\cdot B)\cdot(\overline{C\cdot D})$

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

* Addition

Rules

1. 0 + 0 = 0
2. 0 + 1 = 1
3. 1 + 1 = 0 (10 → 1 carry)
4. 1 + 0 = 1

Questions

(1)

```
  0 0 0 1 1 0 1 0
+ 0 0 0 0 1 1 0 0
  ---------------
  0 0 1 0 0 1 1 0
```

(2)

```
  0 0 0 1 0 0 1 1
+ 0 0 1 1 1 1 1 0
  ---------------
  0 1 0 1 0 0 0 0
```

(3)

```
  0 0 0 1 1 0 1 0
  0 0 0 1 0 0 1 1
  0 0 0 0 1 1 0 0
+ 0 0 1 1 1 1 1 0
  ---------------
  0 1 1 1 0 1 1 1
```

(4)

```
    1 1 1 1 1
    1 0 0 1 0
    0 1 1 0 1
+   1 1 0 1 1
  -----------
 10 1 1 0 0 1
```

⑤

```
     1  1  1     1
     1  1  1  1  1
     1  0  0  1  0
    10  1  1  0  1
+   11  1  0  1  1
   101  1  0  0  1
```

Subtraction

Rules

① 0 - 0 = 0

② 0 - 1 = 1 (borrow 1)

③ 1 - 0 = 1

④ 1 - 1 = 0

①

```
   1 0 1 1 0 1 1
 - 0 0 1 0 0 1 0
   1 0 0 1 0 0 1
```

②

```
   0 → 1   0 → 1
   1   0   1   0   1 1   0
 -     1   0   1   0 1   0
   0   1   0   1   1 0   0
```

③

```
   0 → 1 → 1 → 1
   1   0   0   0   1   0   1
 -     1   0   1   1   0   0
   0   0  ~~0~~1   1   0   0   1
```

④

```
   0 → 1 → 1 → 1 → 0 → 1
   1   0   0   0   1   0   1   1   0
 -         1   1   1   1   0   1   0
   0   1   0   0   1   1   1   0   0
```

⑤

```
           0 → 0 → 1
   1   0   1   1   0   1
 - 1   0   0   1   1   1
   0   0   0   1   1   0
```

⑥

```
       0   0   1   0   0
   1   1   1   0  ~~0~~1   1   0
 - 1   0   1   0   1   1   1
   0   0   1   1   1   1   1
```

Aug 14, 2

Multiplication

Rules

① $0 \cdot 0 = 0$

② $0 \cdot 1 = 0$

③ $1 \cdot 0 = 0$

④ $1 \cdot 1 = 1$

①

$$
\begin{array}{r}
1011 \\
\times\ 1001 \\
\hline
1011 \\
0000\times \\
0000\times\times \\
1011\times\times\times \\
\hline
1100011 \\
\hline
\end{array}
$$

Date: / /20
Page:

(2) 0 0 1 0 1 0 0 1
x 0 0 0 0 0 1 1 0

0 0 0 0 0 0 0 0
0 0 1 0 1 0 0 1 x
0 0 1 0 1 0 0 1 x x
0 0 0 0 0 0 0 0 x x x
0 0 0 0 0 0 0 0 x x x x
0000 0 0 0 0 x x x x x
00000 0 0 0 x x x x x x
000000 0 0 x x x x x x x

000000 0 1 1 1 1 0 110

but bit ⟹ 110 only 8, ans is 0011110110.

(3) 00010111 x 00000011

0 0 0 1 0 1 1 1
x 1 1
0 0 0 1 0 1 1 1
0 0 0 1 0 1 1 1 x

0 0 1 0 0 0 1 0 1

(4) 11101 x 10001

1 1 1 0 1
x 1 0 0 0 1

1 1 1 0 1
0 0 0 0 0 x
0 0 0 0 0 x x
0 0 0 0 0 x x x
1 1 1 0 1 x x x x

1 1 1 1 0 1 1 0 1

Date: / /20
Page:

Division

Divisor ⟌ Divident ⌊ Quotient

Rules
(1) 0 ÷ 0 =
(2) 0 ÷ 1 =
(3) 1 ÷ 0 =
(4) 1 ÷ 1 =

(1) 110 div. by 11

11 ⟌ 110 ⌊ 10
11
00
00
2

(2) 1111 ÷ 110

110 ⟌ 1111 ⌊ 10.1
110
x 110
110
x

(3) 101010 ÷ 00000110

110 ⟌ 101010 ⌊ 110.11
110
x 110
110
x 0100
0000
x 100

Aug 16;24

8421 Code → BCD (binary code decimal)

denoted numbers 0 to 15.

A B C D → 8421

No. →	8	4	2	1	A B C D
0	0	0	0	0	$\bar{A}\bar{B}\bar{C}\bar{D}$
1	0	0	0	1	$\bar{A}\bar{B}\bar{C}D$
2	0	0	1	0	$\bar{A}\bar{B}C\bar{D}$
3	0	0	1	1	$\bar{A}\bar{B}CD$
4	0	1	0	0	$\bar{A}B\bar{C}\bar{D}$
5	0	1	0	1	$\bar{A}B\bar{C}D$
6	0	1	1	0	$\bar{A}BC\bar{D}$
7	0	1	1	1	$\bar{A}BCD$
8	1	0	0	0	$A\bar{B}\bar{C}\bar{D}$
9	1	0	0	1	$A\bar{B}\bar{C}D$
10	1	0	1	0	$A\bar{B}C\bar{D}$
11	1	0	1	1	$A\bar{B}CD$
12	1	1	0	0	$AB\bar{C}\bar{D}$
13	1	1	0	1	$AB\bar{C}D$
14	1	1	1	0	$ABC\bar{D}$
15	1	1	1	1	$ABCD$

BCD code has valid code from 0 to 9 and has invalid code from 10 to 15.

→ It is a binary code of 4 bits. We have $2^4 = 16$ combinations but only 10 out of 16 combinations are considered as a valid in BCD, i.e, 0-9 and the rest are invalid in BCD, i.e, 10-15.

Ques Convert to decimal

(i) $\underline{0010}\,\underline{0011}\,\underline{0100}$ ⇒ 234

(ii) $\underline{0110}\,\underline{0101}$ ⇒ 65

(iii) $\underline{0011}\,\underline{0100}.\underline{1000}$ ⇒ 34.8

Ques Convert to BCD

(i) 64 ⇒ 01100100

(ii) 34.8 ⇒ 00110100.1000

Addition of BCD Code Numbers

4-digit sum ≤ 9 → valid

4-digit sum ≥ 9 (in ques) → invalid, so to make it valid add 06 = 0110

(i) 1001 + 0101

$$\begin{array}{r} 1\,0\,0\,1 \\ +\ 0\,1\,0\,1 \\ \hline 1\,1\,1\,0 \end{array} = 14$$

as 14 > 9, so,

$$\begin{array}{r} 1\,1\,1\,0 \\ +\ 0\,1\,1\,0 \\ \hline \underbrace{0001}_{1}\ \underbrace{0100}_{4} \end{array} \quad (+6)$$

(ii) 0011 + 0100

$$\begin{array}{r} 0\,0\,1\,1 \\ 0\,1\,0\,0 \\ \hline 0\,1\,1\,1 \end{array} = 7$$

(iii) 00100011 + 00010101

  0 0 1 0 0 0 1 1
+ 0 0 0 1 0 1 0 1
  0 0 1 1 1 0 0 0 = 38

(iv) 1001 + 1001

  1 0 0 1
+ 1 0 0 1
1 0 0 1 0 = 12

add 6,

  1 0 0 1 0
+   0 1 1 0
  1 1 0 0 0 = 18

(v) 00010110 + 00010101

  0 0 0 1 0 1 1 0
+ 0 0 0 1 0 1 0 1
  0 0 1 0 1 0 1 1 ⇒ 211

+6, 00101011
  + 0110
  00110001

⇒ 31

Date: / /20
Page:

Aug 27, 24

Excess - 3 Code

It is an another BCD code. Excess-3 code is a digital code in which decimal digits is converted to BCD and then adding decimal 3 to each decimal digit BCD. When no. > 3, then add 3.
Excess-3 code contains no. greater than 3.

⇒ Method for conversion of decimal to Excess-3 code

① Add 3 to each decimal digits. Convert obtained result to 4-bit binary code.

eg. → ⓐ 9
↳ 9 + 3 = 12, i.e., 1100

$$\begin{array}{r} 1001 \\ 0011 \\ \hline 1100 \end{array}$$

ⓑ 654
↳ 654 + 3 = 987, i.e., ~~011000101 0111~~
100110000111

$$\begin{array}{rrr} 6 & 5 & 4 \\ +3 & +3 & +3 \\ \hline 9 & 8 & 7 \end{array}$$

$$\begin{array}{rrr} 0110 & 0101 & 0100 \\ +0011 & +0011 & +0011 \\ \hline 1001 & 1000 & 0111 \end{array}$$

ⓒ 159
↳ 159 + 3 = 4

$$\begin{array}{rrr} 0001 & 0101 & 1001 \\ +0011 & +0011 & +0011 \\ \hline \underbrace{0100}_{4} & \underbrace{1000}_{8} & \underbrace{1100}_{12} \end{array}$$

→ 4812

Uses → Excess-3 code is having self complementary property. This property is helpful in digital computers for performing substraction operations.

$\overline{4}$ $\overline{8}$ $\overline{1}$ $\overline{2}$ (LSP)

0010
0011
0101

$2 + 3 = 5$

=> 481 → excess-3 code ← 5

Let's make a table for BCD and excess-3 code

Decimal no.	BCD code	Excess-3 code
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	10100	1011
9	1001	1100

Gray Code

It is used to change the single bit.

It exhibits only a single bit change from one code no. to next. Hence, it is popular as the min. change code. That's why this code is not suitable for performing arithematic operations. It is not an arithematic code.

BINARY to GRAY CODE

→ Let's take a binary number & convert in gray code.

① 1 0 1 1 is a first ques. (neglect carry)

1 0 1 1

1 1 1 0 is a gray code

② 111001010

1 1 1 0 0 1 0 1 0

1 0 0 1 0 1 1 1 1 is gray code

③ Table for binary to gray code

Decimal	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	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1

GRAY CODE TO BINARY

① 1001

1 0 0 1

↓ + + +

1 1 1 0

② 0101

0 1 0 1

↓ + ↓ + ↓ + ↓

0 1 1 0

③ 100101111

1 0 0 1 0 1 1 1 1

↓ + ↓ + ↓ + ↓ + ↓ + ↓ + ↓ + ↓ + ↓

1 1 1 0 0 1 0 1 0

Ques (i) 49 , convert into binary, gray & excess-3 code

49 → 01001001 (~~binary~~)

→ 01101101 (gray)

→

binary,

2	49	
2	24	1
2	12	0
2	6	0
2	3	0
	1	1

⇒ 110001

1 1 0 0 0 1 binary

1 0 1 0 0 1 gray code

$$\begin{array}{cc} 4 & 9 \\ +3 & +3 \\ \hline 7 & 12 \end{array}$$

$\hookrightarrow 2+3=5$

⇒ 71 , 5 excess-3 code

(ii) 38

2	38	
2	19	0
2	9	1
2	4	1
2	2	0
	1	0

⇒ 100110 binary

⇒ 1 1 0 1 0 1 gray code

$$\begin{array}{cc} 3 & 8 \\ +3 & +3 \\ \hline 6 & 11 \end{array}$$

$\hookrightarrow 1+3=4$

⇒ 61 , 4 excess-3 code

→ 1's complement
bits changed, $1 \rightarrow 0$
$0 \rightarrow 1$

for eg.
⊙ 11011 → 00100

Ques Sub. $\underbrace{0011}_{\text{subtrahend}}$ from $\underbrace{0101}_{\text{minuend}}$ using 1's complement.
(which is substracted) (from which is substracted)

by 1's, 0011 ⟶ 1100

add 1's complement and minuend,

$$
\begin{array}{r}
0101 \\
+\ 1100 \\
\hline
10001 \\
\hline
+\ 1 \\
\hline
\underline{0010}
\end{array}
$$

add carry again

Ques Sub. 100111 from 110011 by 1's complement.

by one's complement, 100111 → 011000

add,

$$
\begin{array}{r}
011000 \\
+\ 110011 \\
\hline
1001011 \\
+\ 1 \\
\hline
\underline{001100}
\end{array}
$$

Date: / /20
Page:

→ 2's complement

① 101101

1's complement → 010010
add 1 → + 1

$$\begin{array}{r} 010010 \\ +\quad 1 \\ \hline 010011 \end{array}$$

→ 9's complement

9's complement of a binary number is obtained by subtracting each bits of number by '9' individual.

eg. → find 9's complement of 63.
2 bits in 63, i.e., 6 and 3
sub. ~~add~~ 2 bits of 9, i.e., 99

$$\begin{array}{r} 9\ 9 \\ -\ 6\ 3 \\ \hline 3\ 6 \end{array}$$

→ 10's complement

① find 10's complement of 63.

9's complement,

$$\begin{array}{r} 99 \\ -63 \\ \hline 36 \end{array}$$

add 1

$$\begin{array}{r} +1 \\ \hline 37 \end{array}$$

37 → 10's complement

Date: / /20
Page:

→ 9's complement & 10's complement subtraction

① Subtract 82 from 93 using 9's complement.
⇒ When (subtrahend < minored
82 < 93)

→ 9's complement of subtrahend

$$\begin{array}{r} 99 \\ -\ 82 \\ \hline 17 \end{array}$$

→ Add in minored,

$$\begin{array}{r} 17 \\ +\ 93 \\ \hline 110 \\ +\ \ 1 \\ \hline 11 \\ \hline \end{array}$$ -Ans

② Subtract 82 from 93 using 10's complement.
⇒ → 10's complement of subtrahend

9's complement $$\begin{array}{r} 99 \\ -\ 82 \\ \hline 17 \end{array}$$

add 1 $$\begin{array}{r} +\ \ 1 \\ \hline 18 \end{array}$$

→ Add with minored,

$$\begin{array}{r} 9\ 3 \\ +\ 1\ 8 \\ \hline 1\ 1\ 1 \end{array}$$

↳ discard the carry

Ans ⇒ 11

when
subtrahend > minored

① Subtract 93 from 82 using 9's complement
⇒ (subtrahend) (minored)

→ 9's complement of subtrahend

$$\begin{array}{r} 9\ 9 \\ -\ 9\ 3 \\ \hline 0\ 6 \end{array}$$

→ Add with minored

$$\begin{array}{r} 8\ 2 \\ +\ 0\ 6 \\ \hline 8\ 8 \end{array}$$

→ Sub 88

Date: / /20
Page:

② Subtract 93 (subtrahend) from 82 (minored) using 10's complement
⇒

→ 10's complement of 93

$$\begin{array}{r} 99 \\ -\ 93 \\ \hline 06 \\ +1 \\ \hline 07 \end{array}$$

→ Add with minored,

$$\begin{array}{r} 82 \\ +\ 07 \\ \hline 89 \end{array}$$

→ 9's complement

$$\begin{array}{r} 99 \\ -\ 89 \\ \hline 10 \end{array}$$ Ans.

1's Complement Subtraction

when, sub < minored

① Subtract 0011 using 1's complement from 0101
→ 1's complement of subtrahend

0011 → 1100

→ Add minored,

$$
\begin{array}{r}
1\ 1\ 0\ 0 \\
+\ 0\ 1\ 0\ 1 \\
\hline
1\ 0\ 0\ 0\ 1 \\
+\ 1 \\
\hline
0\ 0\ 1\ 0
\end{array}
$$

Ans.

2's Complement Subtraction

① Sub. 0011 from 0101.

→ 2's complement of subtrahend

$$
\begin{array}{r}
0011 \rightarrow 1100 \\
+\ 1 \\
\hline
1101
\end{array}
$$

→ Add with minored

$$
\begin{array}{r}
1\ 1\ 0\ 1 \\
0\ 1\ 0\ 1 \\
\hline
1\ 0\ 0\ 1\ 0
\end{array}
$$

→ discard carry

Ans - 0010

ERROR DETECTION AND CORRECTION TECHNIQUES

① Error Detection Code → Every digit of digital system must be correct, an error in any digit can cause a problem because

(a) Parity Code → the computer may recognise it as something else.

An error in one bit, i.e., 1010 error can be detect by many method.

1 → Parity code
2 → Hamming code

Parity Code → A system that checks for error in multibits binary numbers by counting the number of 1's.
It is of two types – ① Even Parity bit
② Odd Parity bit

① Even Parity Bit → An error checking system that require a binary number to have even no. of 1's. An even parity no. of 1's are even.

② Odd Parity Bit → An error checking system that require a binary no. to have odd no. of 1's.

→ for eg. → by checking with odd parity
0 0 1 0 0 1 → error (2 one's)

make 3 one's so 0 0 1 0 0 1 1

→ by checking with even parity
0 0 1 0 0 1 1 → error (3 one's)

make 4 one's so 0 0 1 0 0 1 1 1

Odd Parity code : A bit appended to binary no. to make no. of 1's even or odd.

1. No. of 1's are odd, at the receiver with odd parity bits, zero is added.

2. No. of 1's is even, at the receiver with odd parity bits, one is added.

Even Parity code :

1. No. of 1's are even, at the receiver with even parity bits, zero is added.

2. No. of 1's are odd, at the receiver with even parity bits, one is added.

No. of 1's is even → add 1, odd → add 0 } odd

even → add 0, odd → add 1 } even

→ The discuss parity bit or check bit can't detect 2 errors in the same code 01000011 is transmitted instead of 011

Examples:-

	for even parity	for odd parity
① 00000100	1	0
② 01000100	0	1
③ 00111100	0	1

Parity bit + Data

Error Correcting using Parity

We can use error correction code, i.e., hamming code to overcome the problem of parity detection code.

In parity detection code it is not possible to correct the corrupted data. It is the code used for detecting error and correcting it. It's a single error correction not only provides for detection of bits error but also identifies the bit that is in error. So the parity bits is depend on the no. of information bits.

$$\boxed{2^p \geq n+p+1}$$

↳ for error detection & correction

Q n = data transmitted
p = parity bit
2^p = total no. of parity bits

where an eg. is
⊙ if $n = 4$,

⇒ as acc. to formula, $2^p \geq n + p + 1$

$p = 0$, $p = 1$, $p = 2$, $p = 3$ $(8 \geq 8)$
× × × ✓

so, 3 parity bits are added
Hence, for 4 bit data, 3 parity bits are used.

When the no. of parity bit are determined, then the total no. of information bit is 7 (seven) $(n+p)$ $(4+3)$.

where to add parity bits in information bit is given by formula,
position ~~$2^n = 0, 1, 2, 3, 4 \ldots$~~

as 3 bits are added,
so $2^n = 0, 1, 2$ (upto 3)

$2^0 = 1$
$2^1 = 2$
$2^2 = 4$

Parity bits are located at the position, i.e., 1, 2, 4

P_1	P_2	D_3	P_4	D_5	D_6	D_7
001	010	011	100	101	110	111

Date: / /20
Page:

P_1	P_2	D_3	P_4	D_5	D_6	D_7
001	010	011	100	101	110	111

III II I

Pairing of no. of 1's,

I^{st} position → 1, 3, 5, 7
II^{nd} position → 2, 3, 6, 7
III^{rd} position → 4, 5, 6, 7

→ Set a parity bit to 1 if total no. of 1's in a position, checked is odd and otherwise 0.

Ques Code word → 011100101010, is transmitted and 011100101110 is received. Determine the error if even parity is used.

→ length of transmitted code = 12
which is consists of parity bits also.

$2^p \geq n+p+1$
$2^5 \geq 12+5+1$ } p = 5] Nos ×
$32 \geq 18$

Given, data + parity = 12
$(n+p) = 12$

$\Rightarrow 2^p = n+p$
$2^4 = 12+4$
$16 = 16$ So, $p = 4$.

for 4 bits, $2^n = 0, 1, 2, 3$

$2^0 = 1$, $2^1 = 2$, $2^2 = 4$, $2^3 = 8$ } no. of positions

Date : / /20
Page :

	1	2	3	4	5	6	7	8	9	10	11	12
Bits	P_1	P_2	D_3	P_4	D_5	D_6	D_7	P_8	D_9	D_{10}	D_{11}	D_{12}
Allocation Binary	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100
Received code →	0	1	1	1	0	0	1	0	1	1	1	0

Ist position → 1, 3, 5, 7, ~~8~~, 9, 11 = 6 Add 0
II position → 2, 3, 6, 7, 10, 11 = 6 Add 0
III position → 4, 5, 6, 7, 12 = 5 Add 1
IV position → 8, 9, 10, 11, 12 = 5 Add 1

in this bit position, if we see their corresponding binary value 1 at the right-most position. In received code there are 4 1's so the parity checked is correct. We take 0 at LSB

in Ist position, even no. of 1's in 1, 3, 5, 7, 9, 11 is on 3, 5, 9; add 0
odd no. of 1's is at 1, 7, 11; add 1.
total 1's of received code in 1, 3, 5, 7, 9, 11 position are 4, so add 0.

In II position, total 1's of received code in 2, 3, 6, 7, 10, 11 position are 5, so add 1.

In III position, total 1's of received code in 4, 5, 6, 7, 12 are 2, so add 0.

In IV position, total 1's of received code is 8, 9, 10, 11, 12 are 3, so add 1.

Ans → $P_4 P_3 P_2 P_1$ → 1010

Q- Construct the error correcting code 1001 using even parity.

length of code = 4

$2^p \geq n+p+1$ (1 include bz parity is not given)

$2^3 \geq 4+3+1$

$8 \geq 8$

2^n, $n = 0, 1, 2$ $p = 3$, no. of parities.

$2^0 = 1$, $2^1 = 2$, $2^2 = 4$ position of parity

$n+p = 4+3 = 7$,

Bits	1	2	3	4	5	6	7
Allocation	P_1	P_2	D_3	P_4	D_5	D_6	D_7
Binary	0001	0010	0011	0100	0101	0110	0111
Code			1		0	0	1

		no. of 1's	Add
I	position → 1, 3, 5, 7	2	0
II	position → 2, 3, 6, 7	2	0
III	position → 4, 5, 6, 7	1	1

hence, 100.

III II I

P_4 P_2 P_1

Data → 0 0 1 1 0 0 1

P_1 P_2 D_3 P_4 D_5 D_6 D_7

Sept 9, 2024

Date: / /20
Page:

Ques Code word → 011100101010 is transmitted and 011100101110 is received. find the error if even parity is used.

→ length of code word or transmitted = 12

data + parity = length

$n + p = 12$

$2^p \geq n+p+1$

but neglect 1, bz $n+p = 12$, p included.

$2^p \geq n+p$

$2^4 \geq 12+4$

$16 \geq 16$

hence, $p = 4$.

for 4 bits, $2^n, n = 0, 1, 2, 3$

$2^0 = 1$, $2^1 = 2$, $2^2 = 4$, $2^3 = 8$

position of parity is 1, 2, 4 and 8.

Bits	1	2	3	4	5	6	7	8	9	10	11	12
Allocation	P_1	P_2	D_3	P_4	D_5	D_6	D_7	P_8	D_9	D_{10}	D_{11}	D_{12}
Binary Value	0001 (IV III II I)	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100
Received code	0	1	1	1	0	0	1	0	1	1	1	0

	1's position	no. of 1's in received code	Add
I^{st} position →	1, 3, 5, 7, 9, 11	4	0
II position →	2, 3, 6, 7, 10, 11	5	1
III position →	4, 5, 6, 7, 12	2	0
IV position →	8, 9, 10, 11, 12	3	1

hence in reverse direction → 1010

Date: / 20
Page:

Code → 011100$\underline{1010}$10
Transmitted → 0111001011110
↳ error.

Ques→ If we have hamming code 1000010 at received code. Is it correct or incorrect format.

hamming code = n + p = 7

$2^p \geq n+p$
$2^4 \geq 7+4$
$16 \geq 11$
$p=2$

Logical Simplification

De-Morgan's Theorem

① $\overline{A\cdot B} = \overline{A}+\overline{B}$

② $\overline{A+B} = \overline{A}\cdot\overline{B}$

Statement →

Ques ① $\overline{A+\overline{BC}}$

$\overline{A}\cdot\overline{\overline{BC}}$

$\overline{A}\cdot BC$

② $\overline{AB+CD}$

$\overline{AB}\cdot\overline{CD}$

$(\overline{A}+\overline{B})\cdot(\overline{C}+\overline{D})$

③ $(\overline{\overline{AB+BC}+\overline{A\cdot\overline{B}}})$

$\overline{\overline{AB+BC}}\cdot\overline{\overline{A\cdot\overline{B}}}$

$\cancel{\overline{AB}\cdot\overline{BC}\cdot\overline{A\cdot\overline{B}}}$

$\cancel{\overline{A}+\overline{B}\cdot\overline{B}+\overline{C}\cdot A\cdot\overline{B}}$

$(AB+BC)\cdot A\cdot\overline{B}$

$ABA\overline{B}+BCA\overline{B}$

$A(0)+CA(0)$

$=0$

④ $\overline{(A+B)\cdot(\overline{C}\cdot\overline{D}+E+\overline{F})}$

$\overline{A+B}+\overline{\overline{C}\cdot\overline{D}+E+\overline{F}}$

$\overline{A}\cdot\overline{B}+\overline{\overline{C}\cdot\overline{D}}\cdot\overline{E+\overline{F}}$

$\overline{A}\cdot\overline{B}+\overline{\overline{C}}+\overline{\overline{D}}\cdot\overline{E}\cdot\overline{\overline{F}}$

$\overline{A}\cdot\overline{B}+C+\overline{\overline{D}}\cdot\overline{E}\cdot F$

$=(\overline{A}\cdot\overline{B}+C+D)\overline{E}\cdot F$

SOP and POS forms

(Sum of product) (Product of sum)

① SOP → When 2 or more product terms are summed by boolean addition, that is called sum of product or min term (m_0).

{denoted by Σ.}

eg. →

① $A \cdot B + A \cdot \bar{C}$

② $A + \cancel{B} \cdot \bar{C} + A \cdot B \cdot C + \bar{A} \cdot C$

{ $A=1$, $\bar{A}=0$ }

Standard form → eg. → $A \cdot B \cdot \bar{C} + \bar{A} \cdot B \cdot C$ ← (Canonical form)

(all 3 variables present in both factor)

② POS → When 2 or more summed terms are multiplied by boolean multiplication, that is called product of sum ~~of~~ or maxterm (M_0).

{denoted by π} { $A = 0$, $\bar{A} = 1$ }

eg. →

① $(A+B) \cdot (C+D)$

② $(A+\bar{B}) \cdot (\bar{A}+B)$ → standard form

→ 4 variable minterm (m_0) and maxterm (M_0)

4 variable Mo and mo table

Variable ABCD	SOP minterm (m_i)	POS maxterm (M_i)
0000	$\bar{A}\bar{B}\bar{C}\bar{D} = m_0$	$A+B+C+D = M_0$
0001	$\bar{A}\bar{B}\bar{C}D = m_1$	$A+B+C+\bar{D} = M_1$
0010	$\bar{A}\bar{B}C\bar{D} = m_2$	$A+B+\bar{C}+D = M_2$
0011	$\bar{A}\bar{B}CD = m_3$	$A+B+\bar{C}+\bar{D} = M_3$
0100	$\bar{A}B\bar{C}\bar{D} = m_4$	$A+\bar{B}+C+D = M_4$
0101	$\bar{A}B\bar{C}D = m_5$	$A+\bar{B}+C+\bar{D} = M_5$
0110	$\bar{A}BC\bar{D} = m_6$	$A+\bar{B}+\bar{C}+D = M_6$
0111	$\bar{A}BCD = m_7$	$A+\bar{B}+\bar{C}+\bar{D} = M_7$
1000	$A\bar{B}\bar{C}\bar{D} = m_8$	$\bar{A}+B+C+D = M_8$
1001	$A\bar{B}\bar{C}D = m_9$	$\bar{A}+B+C+\bar{D} = M_9$
1010	$A\bar{B}C\bar{D} = m_{10}$	$\bar{A}+B+\bar{C}+D = M_{10}$
1011	$A\bar{B}CD = m_{11}$	$\bar{A}+B+\bar{C}+\bar{D} = M_{11}$
1100	$AB\bar{C}\bar{D} = m_{12}$	$\bar{A}+\bar{B}+C+D = M_{12}$
1101	$AB\bar{C}D = m_{13}$	$\bar{A}+\bar{B}+C+\bar{D} = M_{13}$
1110	$ABC\bar{D} = m_{14}$	$\bar{A}+\bar{B}+\bar{C}+D = M_{14}$
1111	$ABCD = m_{15}$	$\bar{A}+\bar{B}+\bar{C}+\bar{D} = M_{15}$

→ Standard (canonical) form of SOP

$$Y = \Sigma(\bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + \cdots + ABCD)$$
$$= \Sigma(m_0 + m_1 + \cdots + m_{15})$$
$$= \Sigma(0 + 1 + \cdots + 15)$$
$$= \Sigma m\,(0, 1, \cdots, 15)$$

Date: / /20
Page:

→ Standard (Canonical) form of POS

$$Y = \pi\ (A+B+C+D \cdot A+B+C+\bar{D} \ldots \cdot \bar{A}+\bar{B}+\bar{C}+\bar{D})$$
$$= \pi\ (M_0 \cdot M_1 \cdot \ldots \cdot M_{15})$$
$$= \pi\ (0 \cdot 1 \cdot \ldots \cdot 15)$$
$$= \pi M\ (0, 1, \ldots, 15)$$

sept 11, 24

Ques Convert boolean expression $(AB + A\bar{C} + \bar{B}C)$ into canonical SOP form.

⇒ $AB + A\bar{C} + \bar{B}C$

we know, $C+\bar{C}=1$, $B+\bar{B}=1$

$A+\bar{A}=1$

$AB(C+\bar{C}) + A\bar{C}(B+\bar{B}) + \bar{B}C(A+\bar{A})$

$ABC + \underline{AB\bar{C} + AB\bar{C}} + A\bar{B}\bar{C} + A\bar{B}C + \bar{A}\bar{B}C$

$\Rightarrow ABC + AB\bar{C} + A\bar{B}\bar{C} + A\bar{B}C + \bar{A}\bar{B}C$ $[A+A=A]$

Ques Convert boolean exp. into standard POS form

① $(A+\bar{B}+\bar{C}) \cdot (A+\bar{B}+\bar{C}+D) \cdot (\bar{B}+C+D)$

we know, $A \cdot \bar{A} = 0$

$(A+\bar{B}+\bar{C}+D \cdot \bar{D}) \cdot (A+\bar{B}+\bar{C}+D) \cdot (A \cdot \bar{A}+\bar{B}+C+D)$

by distributive law, $(A+B \cdot C) = (A+B) \cdot (A+C)$

$\Rightarrow (A+\bar{B}+\bar{C}+D) \cdot (A+\bar{B}+\bar{C}+\bar{D}) \cdot (A+\bar{B}+\bar{C}+D) \cdot (A+\bar{B}+C+D)$
$(\bar{A}+\bar{B}+C+D)$

$= (A+\bar{B}+\bar{C}+D) \cdot (A+\bar{B}+\bar{C}+\bar{D}) \cdot (A+\bar{B}+C+D) \cdot$
$(\bar{A}+\bar{B}+C+D)$

(as $A \cdot A = A$)

- Convert in canonical SOP form

1. $Y = \bar{A}\bar{B}D + A\bar{B}C + B\bar{C}D$

   as $(A+\bar{A}) = 1$

$\rightarrow \bar{A}\bar{B}(\bar{C}+C)D + A\bar{B}C(D+\bar{D}) + (A+\bar{A})B\bar{C}D$

$= \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}CD + A\bar{B}CD + A\bar{B}C\bar{D} + AB\bar{C}D + \bar{A}B\bar{C}$

- Convert in canonical POS form

1. $Y = (A+B).(\bar{B}+\bar{C}).(A+B+C)$

   as $A.\bar{A} = 0$

$(A+B+C.\bar{C}).(\bar{B}+\bar{C}+A.\bar{A}).(A+B+C)$

$(A+B+C).(A+B+\bar{C}) . (A+\bar{B}+\bar{C}).(\bar{A}+\bar{B}+\bar{C}).(A+B+C)$

$= (A+B+C).(A+B+\bar{C}).(A+\bar{B}+\bar{C}).(\bar{A}+\bar{B}+\bar{C})$

- $Y = A.\bar{B} + B$ is an expression. Construct a truth table.

$Y = A.\bar{B} + B$ (sum of product)

$= A.\bar{B} + B.(A+\bar{A})$

$= A.\bar{B} + A.B + \bar{A}.B$

truth table,

A	B	Y	
0	0	0	
0	1	1	$\bar{A}.B$
1	0	1	$A.\bar{B}$
1	1	1	$A.B$

Date: / /20
Page:

→ Construct a truth table for POS, $Y = (A+B+\overline{C}) \cdot (A+\overline{B}+C) \cdot (\overline{A}+B+\overline{C}) \cdot (\overline{A}+\overline{B}+\overline{C})$

A	B	C	Y	
0	0	0	0	
0	0	1	1	$A+B+\overline{C}$
0	1	0	1	$A+\overline{B}+C$
1	0	0	1	
1	1	0	1	
1	0	1	1	$\overline{A}+B+\overline{C}$
0	1	1	1	
1	1	1	1	$\overline{A}+\overline{B}+\overline{C}$

KARNAUGH MAP (K-map)

K-map is a graphical method for simplyfying the boolean function. It can be used to represent the information of truth table and to minimize the SOP and POS expression.

$2^n = p$ n= Variables, p= no. of boxes

$\Rightarrow 2^0 = 1$

$\Rightarrow 2^1 = 2$

$\Rightarrow 2^2 = 4$ upto 6 Variables.

$\Rightarrow 2^3 = 8$

$\Rightarrow 2^4 = 16$

$\Rightarrow 2^5 = 32$

$\Rightarrow 2^6 = 64$

in SOP, $\bar{A}=0$, $\bar{\bar{A}}=1=A$

↳ filling should be done by 1

for 2 variables, AB

AB	0	1
0	00	01
1	10	11