Digital Circuits

↳ 1 or 0

1. Digital Codes:
   1. Binary Coded Decimal (BCD)
   2. Gray
   3. Parrity
   4. ASCII.

(1.) BCD:

MSB — Most significant bit

LSB — Least significant bit

Decimal (D)	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

2^3 2^2 2^1 2^0 2^n

$5_D \rightarrow 0\ 1\ 0\ 1$

$0.2^3 + 1.2^2 + 0.2^1 + 1.2^0$

$= 0 + 4 + 0 + 1 = 5$

$55_D \rightarrow 0101\ \ 0101$

Application: Numeric displays.

(2.) Gray:

1 bit changes for subsequent no.

Decimal	Gray code
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001
15	1000

(3) Parity: Error detection code.

In binary data transmission (long distance) unusual data change may occur.

Detects odd combination of change.

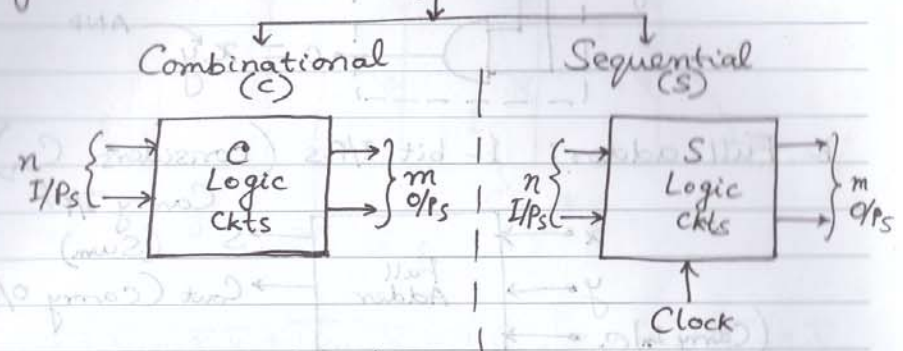
Parity → Odd, Even } Types

First 4 Binary Data/bits or Message	Last/ 5th bit Odd Parity	Last/ 5th bit Even Parity
0000	1	0
0001	0	1
0010	0	1
0011	1	0
0100	0	1
0101	1	0
0110	1	0
0111	0	1
1000	0	1
1001	1	0
1010	1	0
1011	0	1
1100	1	0
1101	0	1
1110	0	1
1111	1	0

(4) ASCII: American Standard Code for Information Interchange.
(Application: Keyboards for PCs)
Special/extended binary code → Alpha-numeric code.
1 byte: 8-bits : $2^8 = 256$ possible codes/values
Refer to ASCII table from internet.

eg. 'a' means 96_D or 61_H
'A' " 65_D or 41_H
'1' " 49_D or 31_H
',' " 44_D or $2C_H$

2. Digital circuits: Types

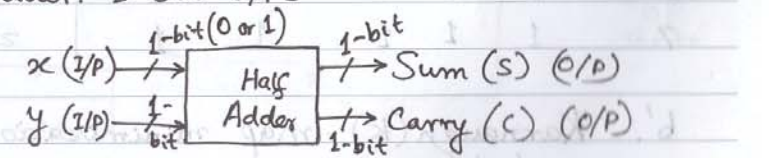


e.g. 1. Adders (Logic/binary adders)
2. Subtractor
3. Decoder/encoder
4. Multiplexer/de-mux (mux)

1. Flip-flops/latches
2. Counters/timers
3. Memory
4. Processors (micro-processors/controllers)

3. Digital adder (binary adder): Not a direct/AC V-adder

a) Half adder: For 1-bit I/Ps

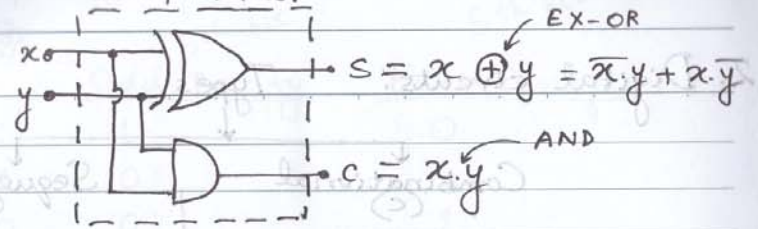


Truth table:

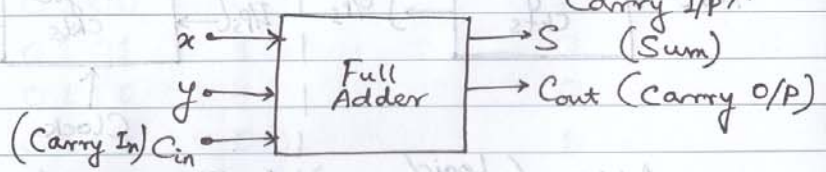
x	y	S	C	x y S C
0	0	0	0	0+0=0, 0
0	1	1	0	0+1=1, 0
1	0	1	0	1+0=1, 0
1	1	0	1	1+1=0, 1

S ← EX-OR O/P
C ← AND O/P

Half Adder



b. Full adder: 1-bit I/Ps (considers C_{in})



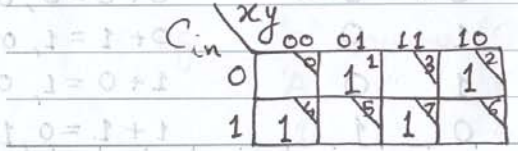
Truth Table:

	C_{in}	x	y	S	Cout
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

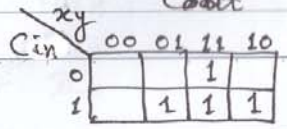
S & Cout are not matching with any standard logic gate.
↓
K-map minimization is reqd'.

b'. Karnaugh (K)-map minimization:
↳ Logical box pattern for arranging required values (8 possibilities for S & another 8 for Cout)

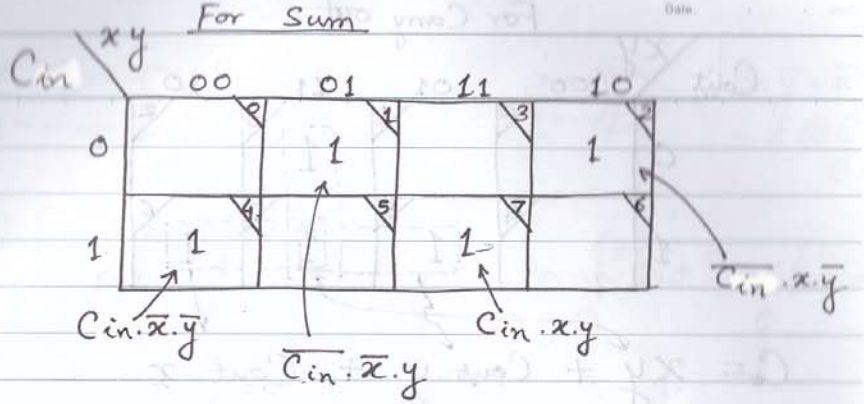
Sum (S)



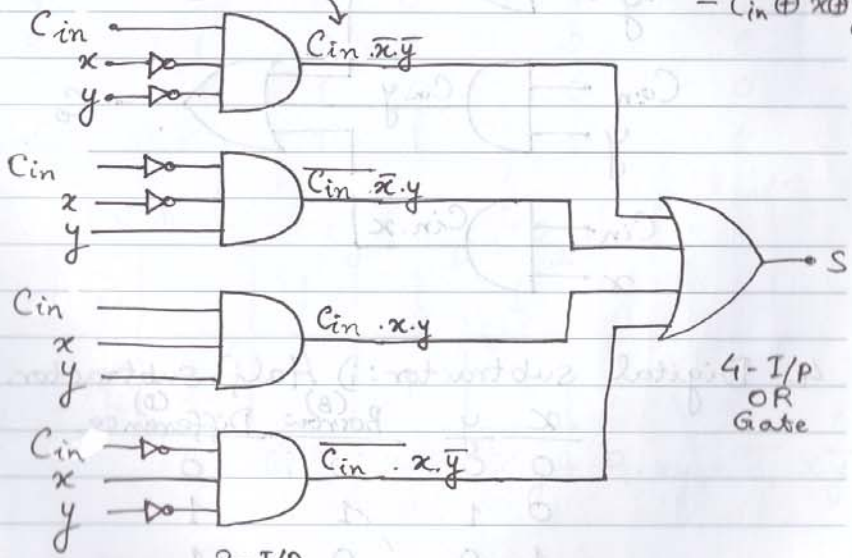
Cout

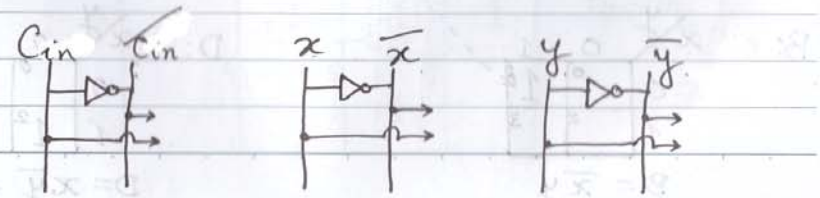


For Sum

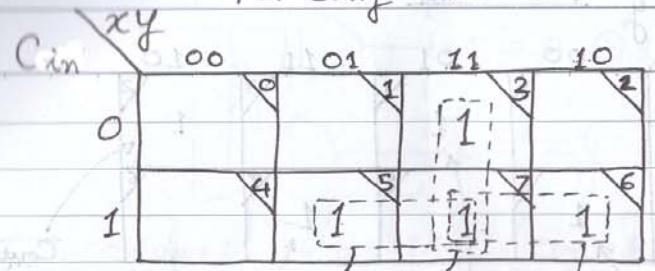


$S = C_{in}.\bar{x}.\bar{y} + \bar{C}_{in}.\bar{x}.y + C_{in}.x.y + \bar{C}_{in}.x.\bar{y}$
$= C_{in} \oplus x \oplus y$

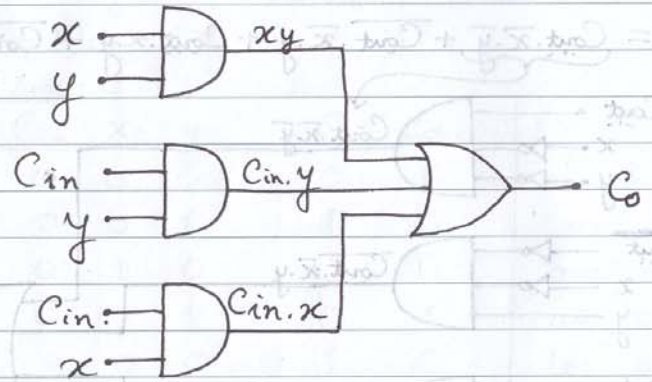




For Carry out



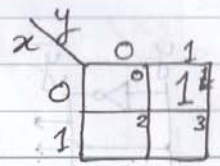
$C_o = xy + C_{in}.y + C_{in}.x$



4. Digital subtractor: i) Half subtractor.

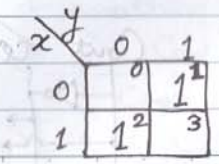
x	y	(B) Borrow	(D) Difference
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

B:

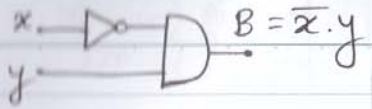


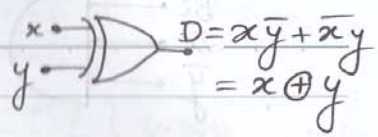
$B = \bar{x}y$

D:



$D = x\bar{y} + \bar{x}.y$

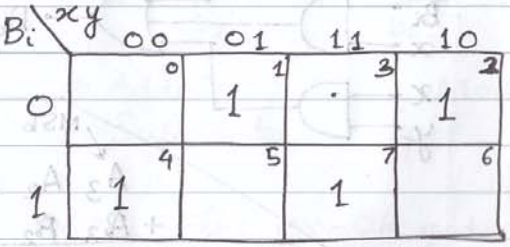




ii) Full Subtractor:

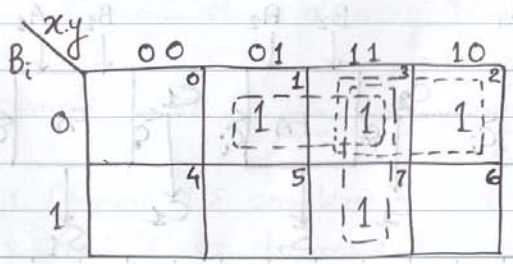
	(Bi) Borrow In	x	y	(Bo) Borrow Out	(D) Difference
0	0	0	0	0	0
1	0	0	1	1	1
2	0	1	0	1	1
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	0	0
6	1	1	0	0	0
7	1	1	1	1	1

D:

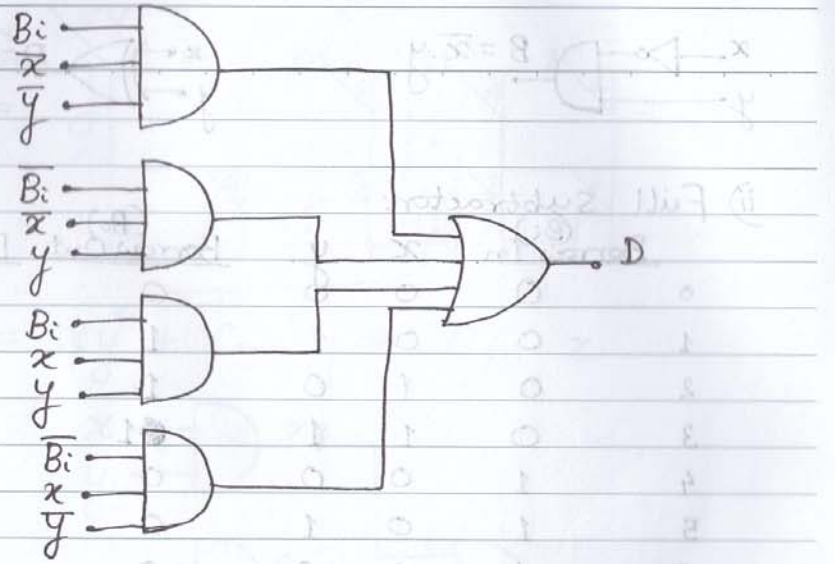


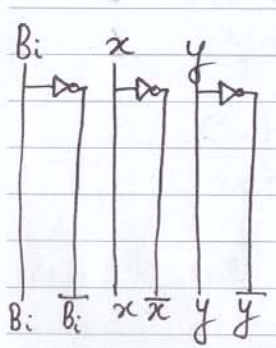
$D = B_i.\bar{x}.\bar{y} + \bar{B_i}.\bar{x}.y + B_i.xy + \bar{B_i}.x.\bar{y}$

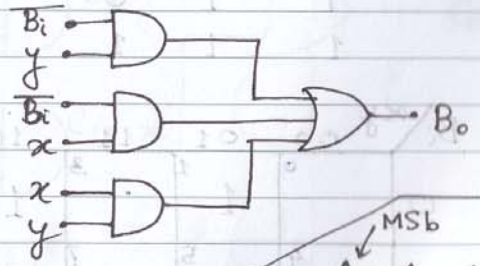
Bo:



$B_o = \bar{B_i}.y + \bar{B_i}.x + xy$







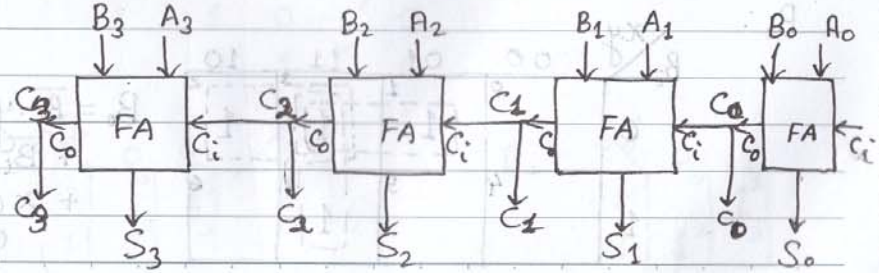
MSb LSb

$A_3\ A_2\ A_1\ A_0$

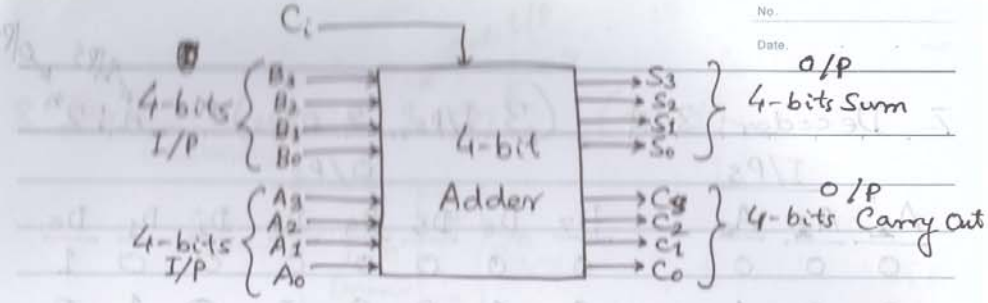
$+\ B_3\ B_2\ B_1\ B_0$

$C_3\ C_2\ C_1\ C_0,\ S_3\ S_2\ S_1\ S_0$

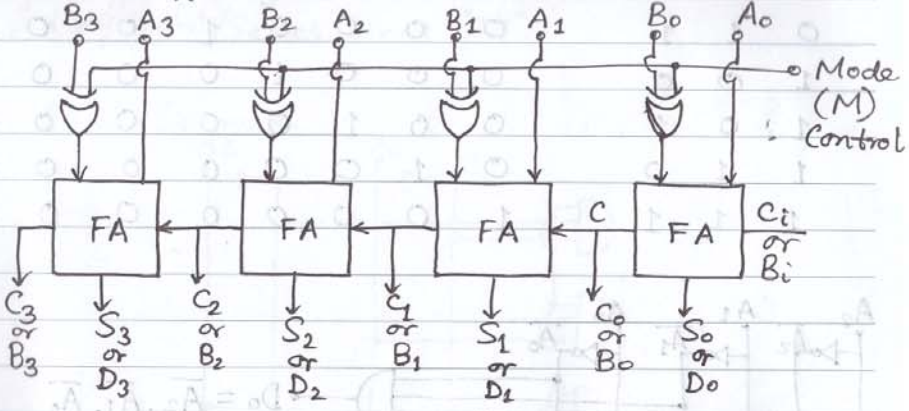
5. Parallel adder: 4-bits



FA: Full adder



6. Parallel adder/subtractor: 4-bits



M=0 ⇒ Adder

M=1 ⇒ Subtractor

} Mode control.

When, M=1, the ex-OR gates act like an inverter (NOT gate) for B input lines.

∴ A-B could be done by taking 2's complement of 'B' and then adding it to 'A'.

2's complement: $\underbrace{\text{1's complement}}_{\text{NOT operation}} + 1$ (NOT gate)

Steps: 1. Compute 1's complement (or NOT) or all bits of 'B' Separately

2. Add '1' to the result. $(\overline{B}+1)$

3. Add the values/bits to A by using parallel FA chain.

$A-B = (A+(\overline{B}+1))$