To convert any # of base ER you take the digit x the r base to the power n = # of digits - 1 so.

conversion of any base to decimal

Positional — Polynomial

$(4021,2)_5 = 4\times5^3 + 0\times5^2 + 2\times5^1 + 1\times5^0 + 2\times5^{-1} = (511,4)_{10}$

5 ↑ base

$500 + 0 + 10 + 1 + 0,4 = 511,4$

same for binary

8 4 2 1

$(1011)_2$ $1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 = (11)_{10}$

$8 + 0 + 2 + 1 = 11$

common bases

binary (base 2) = 1, 0 decimal (base 10) = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

octa: (base 8) = 0, 1, 2, 3, 4, 5, 6, 7 hexa: (base 16) = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A 10, B 11, C 12, D 13, E 14, F 15

conversion to Binary ✱ (works for dec to anything ✱)

decimal

$(62,375)_{10}$ to binary (remainder)

$162/2 = 81$ r = 0 ↑

$81/2 = 40$ r = 1

$40/2 = 20$ r = 0

$20/2 = 10$ r = 0

$10/2 = 5$ r = 0

$5/2 = 2$ r = 1

$2/1 = 1$ r = 0

$1/2 = 0$ r = 1

assemble from bottom to top (reverse) $(10100010)_2$

but for decimal $(0,375)_{10}$

$0,375 \times 2 = 0,750$

$0,750 \times 2 = 1,500$

$0,500 \times 2 = 1,000$ ↓ assemble down

$(10100010,011)_2 = (162,375)_{10}$

Octa takes 3 bits (binary) for one bit Octa

add zero for (3) . 8 4 2 1

$(10110100,001011)_2 = (010\ 110\ 100,\ 001\ 011)_2$

$(2\ \ 6\ \ 4,\ \ 1\ \ 3)_8$

hexa takes 4 bits for (one digit (hexa)

8 4 2 1 ↑ binary

added 2 (0) to make 4 digits ↓

$(10110100,001011)_2$ $(1011\ 0100,\ 0010\ 1100)_2$

$(11\ \ 4,\ \ 2\ \ 12)_{16}$ so since 11 = B ...

$(B4,2C)_{16}$

to convert any base to binary convert it to decimal then to binary.

adding binary #'s

$\begin{array}{r} 0 \\ +0 \\ \hline 0 \end{array}$ $\begin{array}{r} 0 \\ +1 \\ \hline 1 \end{array}$ $\begin{array}{r} {\scriptstyle 1}\downarrow \\ +1 \\ \hline 10 \end{array}$ $\begin{array}{r} {\scriptstyle 1\,1} \\ +1 \\ +1 \\ \hline 11 \end{array}$

ex: $\begin{array}{r} 1001 \\ +0110 \\ \hline 1111 \end{array}$ ex: $\begin{array}{r} {\scriptstyle 1}\,0\,0\,{\scriptstyle 1}\,1 \\ +1\,0\,0\,1 \\ \hline 1\,0\,1\,0 \end{array}$

$\begin{array}{r} {\scriptstyle 1\,1}\;{\scriptstyle 1\,1\,0\,0} \\ +0\,1\,0\,1 \\ \hline 10001 \end{array}$

Radix complement

10's compliment and 2's compliment

10's and 2's $[N]_r = r^n - (n)_r$ r = base. n = # of digits

(doesn't work for $n = 0$)

(10's comp of) $(\overset{1}{7}\overset{2}{2}\overset{3}{0}\overset{4}{9}\overset{5}{2})_{10}$

$[72092]_{10} = 10^5 - 72092$

$= 99999 - 72092$

$\begin{array}{r} 99999 \\ -72092 \\ \hline 27908 \end{array} = (27908)_{10}$

(2's comp of) $= (\overset{1}{1}\overset{2}{0}\overset{3}{1}\overset{4}{0}\overset{5}{0}\overset{6}{1})_2$

$[N]_r = r^n - (N)_r$

$= 2^6 - (01001)_2$

$= 1000000 - 101001 = (010111)_2$

$2^6 = 64$

64 32 16 8 4 2 1

1 0 0 0 0 0 0

Short cut for [2's]

same $(101001)_2$ ← keep all digits until the first one then alternate

101001

010111

Subtraction

$\begin{array}{r} 0 \\ -0 \\ \hline 0 \end{array}$ $\begin{array}{r} 0 \\ -1 \\ \hline 11 \end{array}$ $\begin{array}{r} 1 \\ -0 \\ \hline 1 \end{array}$ $\begin{array}{r} 1 \\ -1 \\ \hline 0 \end{array}$ $\begin{array}{r} 1 \\ -1 \\ \hline 1 \end{array}$

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

Diminished radix compliment

9's compliment and 1's compliment

$[N]_{r-1} = (r^n - 1) - (N)_r$

ex: 9's $[\overset{1}{5}\overset{2}{4}\overset{3}{6}\overset{4}{7}\overset{5}{0}\overset{6}{0}]_9$

$[N]_{r-1} = (r^n - 1) - (n)r$

$= 10^6 - 1 = 546700$

$= 999999 - 546700$

$\begin{array}{r} 999999 \\ -546700 \\ \hline 453299 \end{array}$

1's of $[\overset{1}{1}\overset{2}{0}\overset{3}{1}\overset{4}{1}\overset{5}{0}\overset{6}{0}\overset{7}{0}]$

$= (10^7 - 1) - (1011000)_2$

$= (0100111)_2$

Short cut for (1's)

alternate every digit

1011000

0100111

Complement's have the properties of minus signs.

* Using 2's *

A = 1010100 B = 1000011

B - A = C

1000011 - 1010100 = C

B + [A] = C

-A = 0101100

1000011 + 0101100 = 1101111 C = 1101111

in 2's comp disregard any end carry

1 1 1 1 1
1 0 1 0 1 0 0
+ 0 1 1 1 1 0 1
X 0 0 1 0 0 0 1

A - B = D

[B] = 0111101

Using 1's comp

A = 1010100

B = 1000011

[B] = 0111100

A - B = A + [B]

1 1 1 1 1
1 0 1 0 1 0 0
+ 0 1 1 1 1 0 0
0 0 1 0 0 0 1
0 0 1 0 0 0 1

Subtracting with 9's/10's

$[N]_r = r^n - (n)_r$ = 10's

$[N]_{r-1} = (r^n - 1) - (N)_r$ 9's

72 − 32 = 40

$(72)_{10} - (32)_{10}$

$= 10^2 - 32$

$= 100 - 32$

$= 68$

1 72
+ 68
X 40

disregard end carry for 10

Using 9's comp

$[N]_{r-1} = (r^n - 1) - (n)_r$ $(72)_{10} - (32)_{10}$

$= (10^2 - 1) - (32)_{10}$

$= 99 - 32$

$= 67$

1 72
+ 67
99
40

end carry adds on

Note: When taking the complement of a signed binary # do not change the signed bit!

Signed Binary #'s

signed binary #s are represented with either a 1 or a 0 bit

1 = negative

0 = positive

-3 = 10000011

where 1 is -

Overflow overflow happens when adding signed bits exceeds the range of positive #s

ex: 4 bits machines have a range from 0-15

8 bit " " " " 0-255

* A carry over of 1 does not indicate overflow!

(6 bit) ex: +17 = 010001

+16 = 010000

100001 ← overflow

(7 bit) 0010001

0010000

0100001 ← no overflow!

Binary Coded Decimal is the same as converting hex to binary!

	8	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
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

ex: $(9750)_{10} = (1001011101010000)_{BCD}$

1 = True or high voltage

0 = False low voltage

chap #2

There are 3 basic logic operations AND, OR, NOT

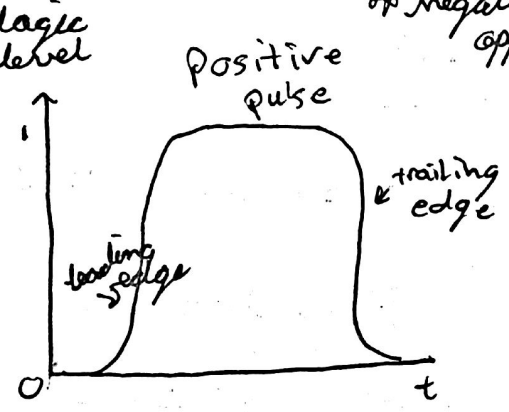
AND is like multiplication

OR is like +

A or B

NOT is opposite

gates AND and OR can have many inputs



negative opposite

$F(x,y,z) = xy' + z x'$ F = Function

literal = x, x', y', z

terms are implementation with gates (xy') (zx')

order of operations

parentheses

NOT

AND

OR

Boolean identities

①	$x+0=x$	②	$x \cdot 1 = x$
③	$x+x=1$	④	$x \cdot x = 0$ or $=x$
⑤	$x+x=x$	⑥	$x+1=1$
⑦	$x \cdot 0 = 0$	⑧	$(x)=x$
		⑨	$\bar{x}x = 0$
⑦	$(x+y)' = x'y'$	> De Morgan	
⑧	$(xy)' = x'+y'$		
⑨	$x+xy = x$	> Absorption	
⑩	$x(x+y) = x$		

①	$x+y = y+x$	commutative
②	$xy = yx$	"
③	$x+(y+z) = (x+y)+z$	associative
④	$x(yz) = (xy)z$	"
⑤	$x(y+z) = xy+xz$	Distributive
⑥	$x+yz = (x+y)(x+z)$	associative

$F_1(A,B,C) = \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}\bar{C} + A\bar{B}C + AB\bar{C} + ABC$

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

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

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

$= \bar{A}B + A$

$F(A,B,C,D) = (AB'(C+BD) + \bar{A}\bar{B})C$

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

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

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

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

$= \bar{B}C$

NAND NOR XOR = exclusive or and XNOR

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

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

There's a standard way of writing Boolean expressions SOP and POS

SOP = $F(\ldots) = (\ldots) + (\ldots) + (\ldots)$

so ~~(and)~~ = (AND's) or (And's) or (. . .

When using sum of products each row in a truth table represent's one minterm

$F(A,B,C) = B + \bar{B}\bar{C}$ ← is SOP

Decimal	A B C	D	minterms		Max terms
0	0 0 0	0	m_0	$\bar{A}\bar{B}\bar{C}\bar{D}$	M_0
1	0 0 0	1	m_1	$\bar{A}\bar{B}\bar{C}D$	M_1
2	0 0 1	0	m_2	$\bar{A}\bar{B}C\bar{D}$	M_2
3	0 0 1	1	m_3	$\bar{A}\bar{B}CD$	M_3
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	m_9	$A\bar{B}\bar{C}D$	M_9

M's are $\overline{m}$, so ~~$M_0 = \overline{ABCD}$~~ $M_0 = A+B+C+D$

$\overline{\bar{A}\bar{B}\bar{C}\bar{D}} = A+B+C+D$ via DeMorgan's rule

so instead of writing $F(A,B,C,D) = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D \ldots$

we can write $F(A,B,C,D) = m_0 + m_1 + m_2 \ldots$

or $= \sum m_i\,(0,1,2,3,4,5,6,7,8,9)$

translate from SOP to POS

$F(A,B,C,D) = (\bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C)$ ← take not of the entire thing

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

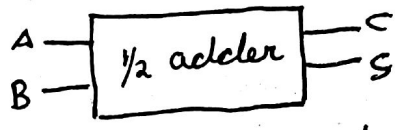
$= (A+B+C) \cdot (A+B+\bar{C})$

Half adder

The half adder accepts two binary bits digits on the input and 2 outputs a sum and a carry.

ex:

a	b	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



so if A = 1 and b = c then C = 0
S = 1

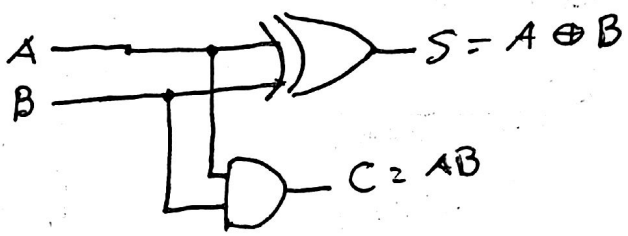
$$\begin{array}{r} a \\ +\,b \\ \hline c\ s \end{array} \quad - \quad \begin{array}{r} 1 \\ +\,0 \\ \hline 0\ 1 \end{array}$$

logic functions

$S = A'B + AB'$

$S = A \oplus B$

$C = AB$



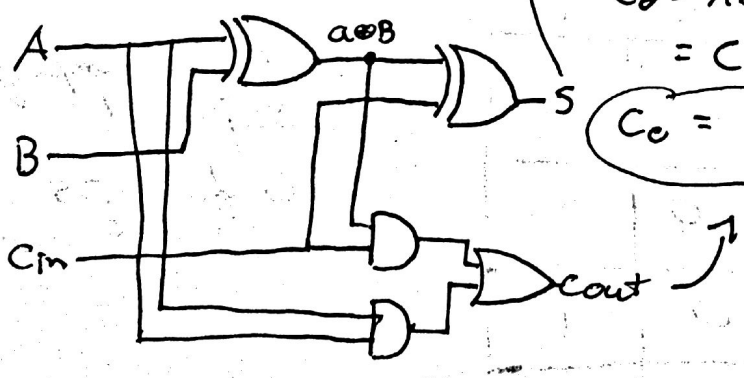
Full adder

A	B	C	C_o	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

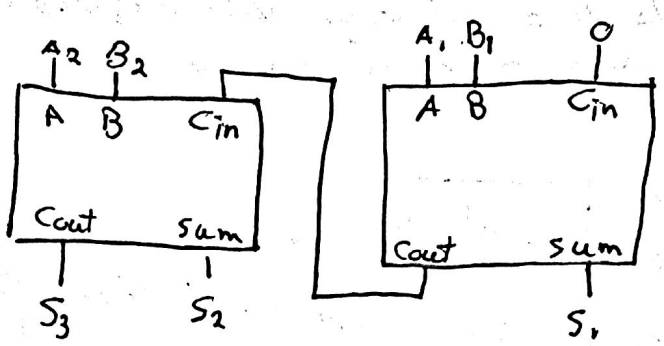
The difference between the 1/2 adder and the full adder is that the full adder accepts 2 input bits and a carry. It also has 2 outputs sum and carryOut

$$S = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$
$$= \bar{A}(\bar{B}C + B\bar{C}) + A(\overline{\bar{B}C + B\bar{C}})$$
$$= \bar{A}(B \oplus C) + A(\overline{B \oplus C})$$
$$= \bar{A}\,\diamond + A\bar{\diamond}$$
$$= A \oplus \diamond = A \oplus B \oplus C = S$$

$$C_o = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$
$$= C(\bar{A}B + A\bar{B}) + AB(\bar{C} + C)$$
$$C_o = C(A \oplus B) + AB$$



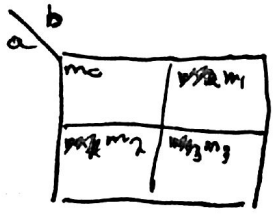
2-bit parallel adder



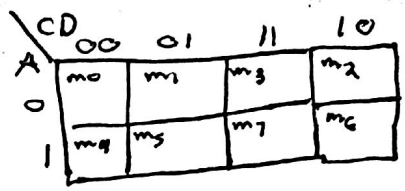
$$\begin{array}{r} A_2\ A_1 \\ +\ B_2\ B_1 \\ \hline S_3\ S_2\ S_1 \end{array}$$

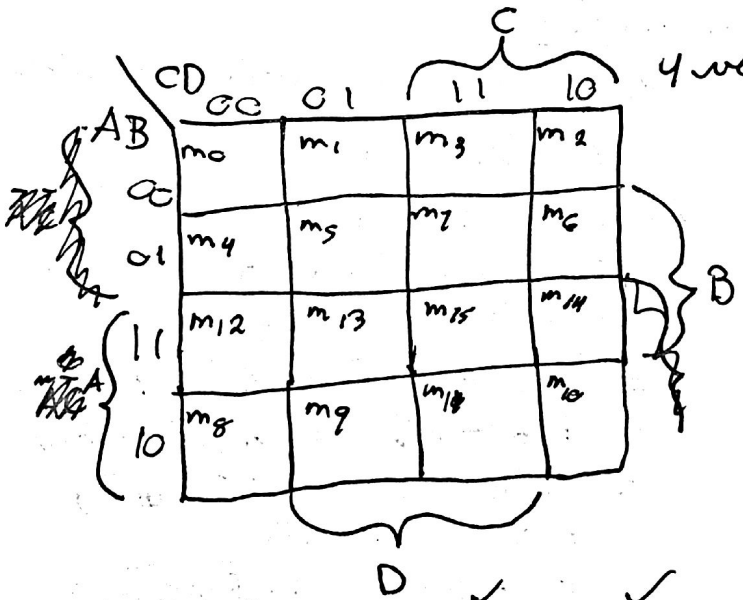
chap #3 K-maps

2 variable K map



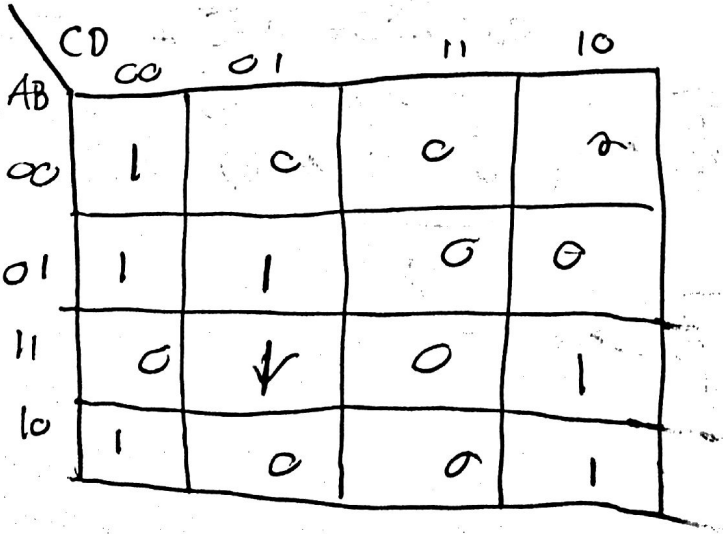
3 variable



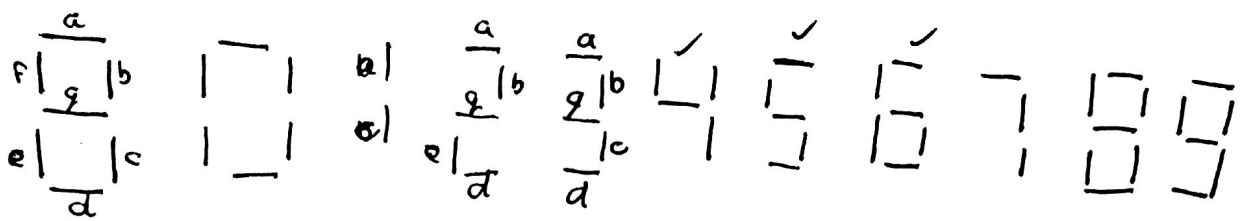


4 variable

$$F(A,B,C,D) = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}\bar{D} + A\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}D + AB\bar{C}D + ABC$$
$$+ A\bar{B}C\bar{D}$$



use 1's to simplify to SOP and 0 for POS and take opposite

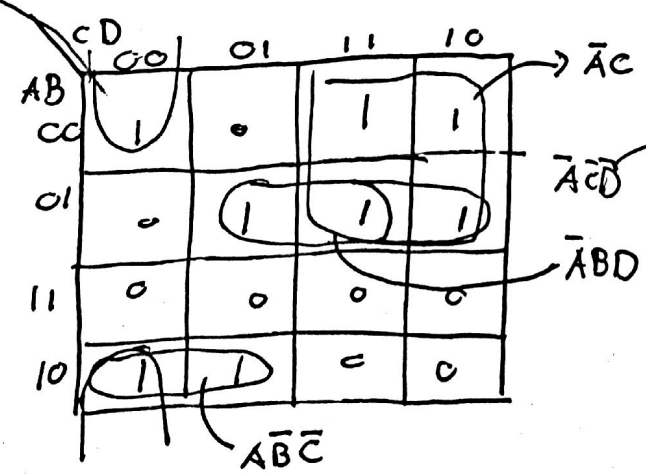


	8 4 2 1 A B C D	a	b	c	d	e	f	g
0	0 0 0 0	1	1	1	1	1	1	0
1	0 0 0 1	0	1	1	0	0	0	0
2	0 0 1 0	1	1	0	1	1	0	1
3	0 0 1 1	1	1	1	1	0	0	1
4	0 1 0 0	0	1	1	0	0	1	1
5	0 1 0 1	1	0	1	1	0	1	1
6	0 1 1 0	1	0	1	1	ϕ	1	1
7	0 1 1 1	1	1	1	0	0	0	0
8	1 0 0 0	1	1	1	1	1	1	1
9	1 0 0 1	1	1	1	1	0	1	1

don't care is

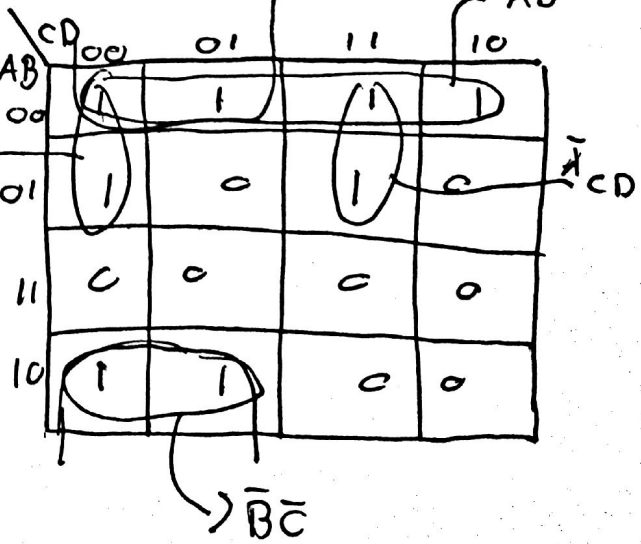
$a = \bar{A}(c + BD) + \bar{B}\bar{C}(A + \bar{D})$

$\bar{B}\bar{C}\bar{D}$

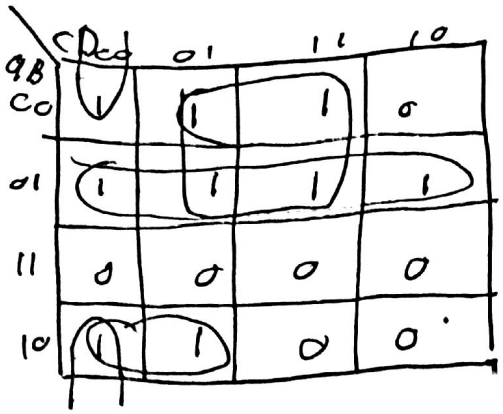


$b = \bar{A}\bar{B} + \bar{A}CD + B\bar{C} + \bar{A}BD$

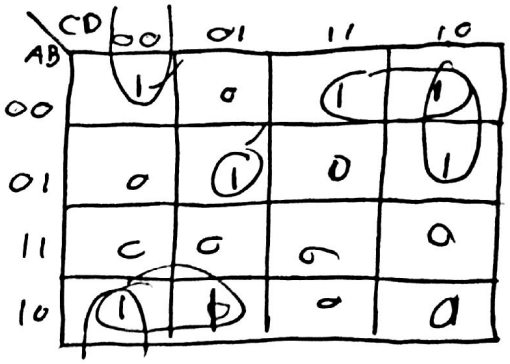
$\bar{A}\bar{B}$



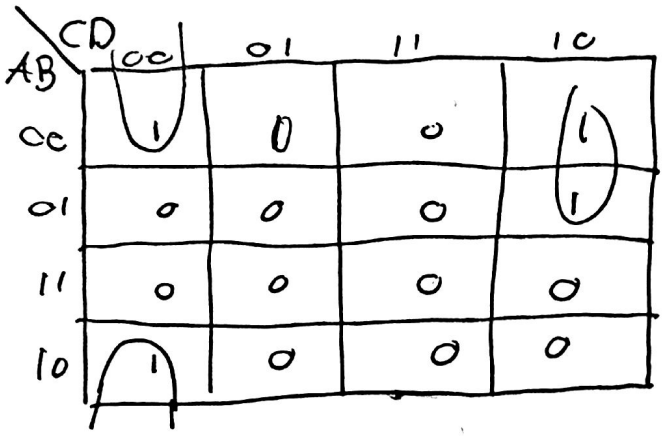
(c) $c = \bar{B}\bar{C}\bar{D} + \bar{A}B + \bar{A}D + A\bar{B}\bar{C}$



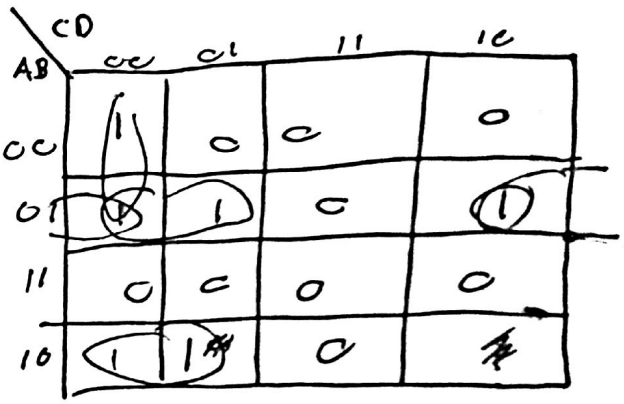
$d = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}D + A\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}CD$



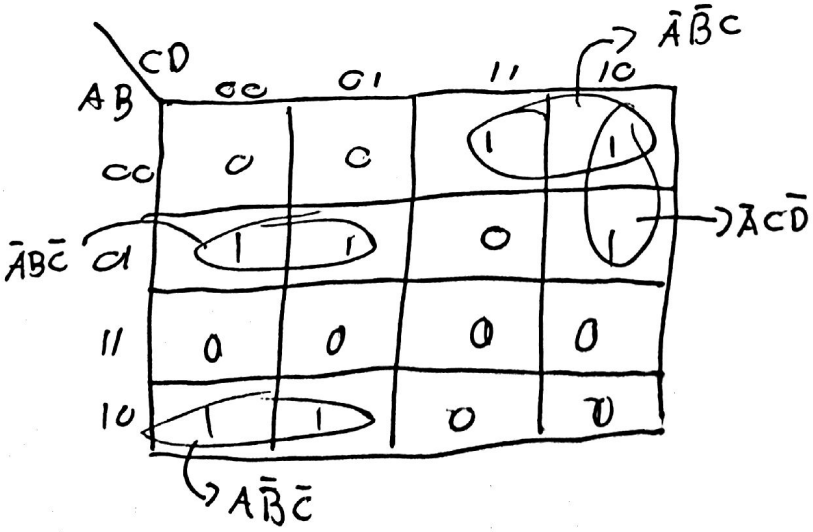
$e = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}C\bar{D}$



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



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



A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines. → n-to-2^n decoder

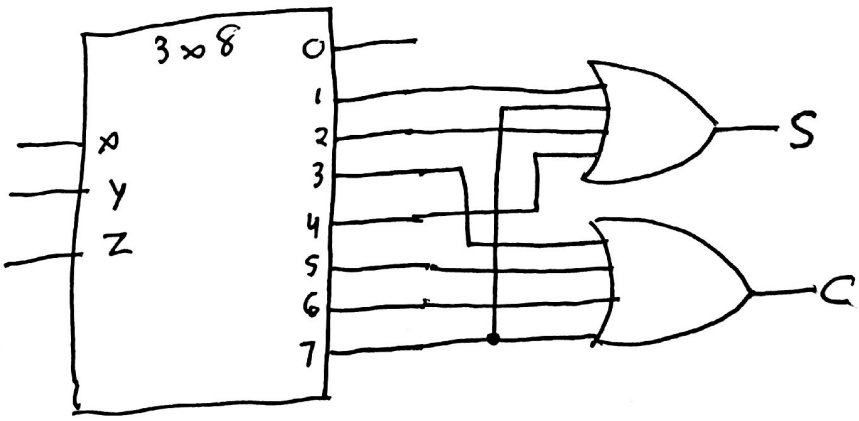
Many devices have an additional enable input, which is used to "activate" or "deactivate" the device
1 = device is on
0 = device is off all outputs = 0

3 to 8 line decoder

takes in 3 inputs and their decoded to 8 outputs. $D_0 \ldots D_7$ where each output represents 1 midterm

$S(x,y,z) = \Sigma m(1,2,4,7)$
→ SOP
$C(x,y,z) = \Sigma m(3,5,6,7)$

from this we can tell we need a 3 to 8 decoder since there are 3 inputs and 8 outputs.



active high gates are for min terms

low = max term

Multiplexers are like a switch a MUX has inputs and the one output and an input switch. the MUX will output an input depending on the input switch

2 to 1 multiplexer