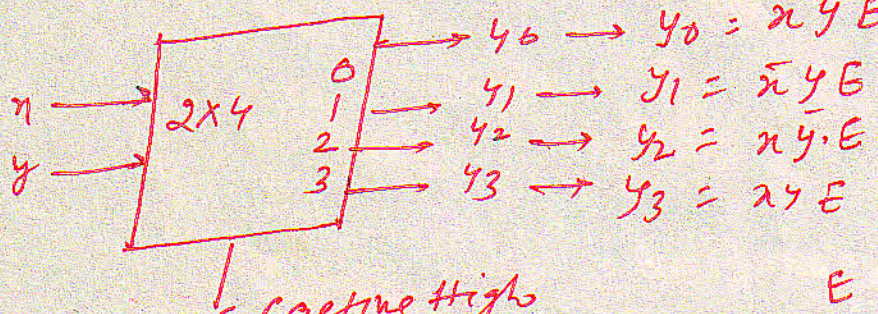
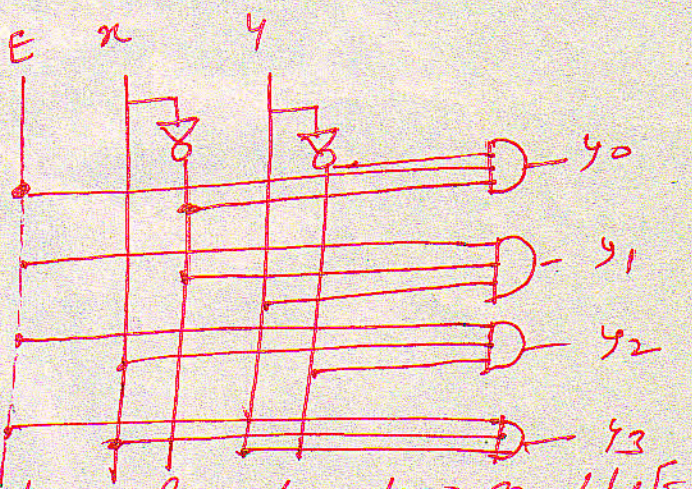
Decoder → Decoder is a combination ckt that convert binary information form n input lines to a maximum of 2^n unique output lines. In general decoder is n to m line converter $m \leq 2^n$ (If some unused or don't care conditions) most. If not all decoder also include one or more enable input to control the ckt operation. and assist cascading.

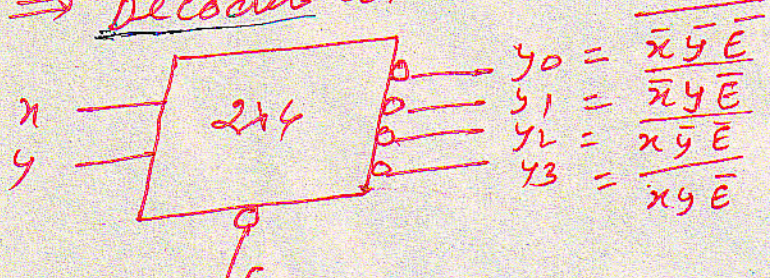
⇒ 2×4 decoder with active high o/p



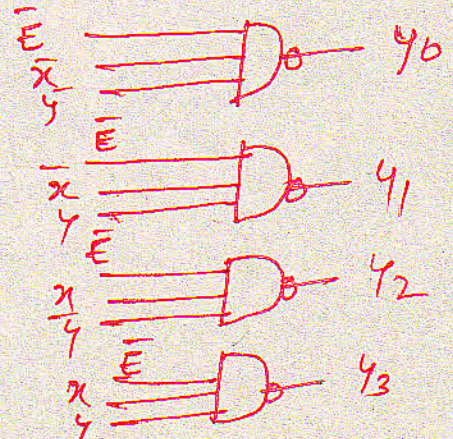
E	x	y	y_0	y_1	y_2	y_3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1
0	x	x	0	0	0	0



⇒ Decoder with active low o/p and active low enable G



E	x	y	y_0	y_1	y_2	y_3
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	x	x	1	1	1	1



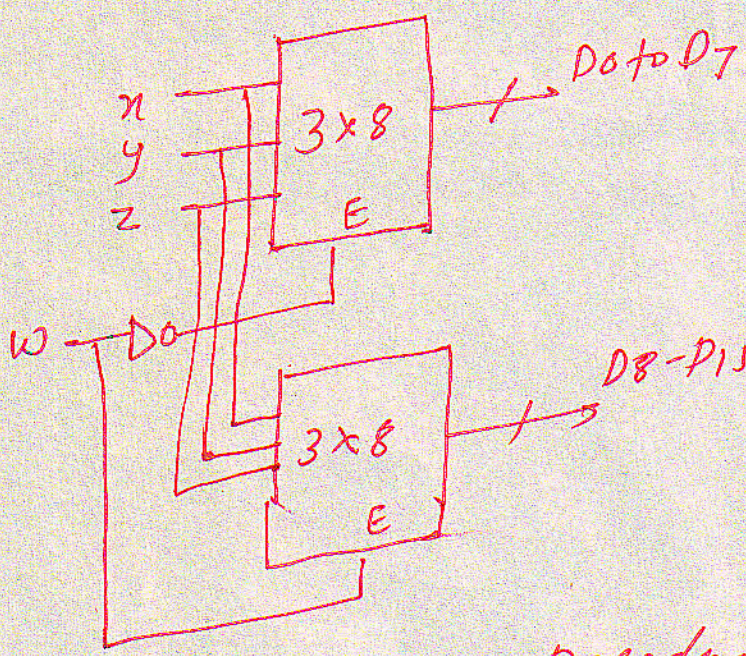
Demultiplexer (SIPO) — A decoder with enable input can function as demultiplexer that receives information as a single line and terminals of the 2^n o/p lines

E line of decoder is taken as input line and input line of decoder act as select lines

e.g. If $xy = 10$ the o/p y_2 will be same as E while all other o/ps are maintained at 1

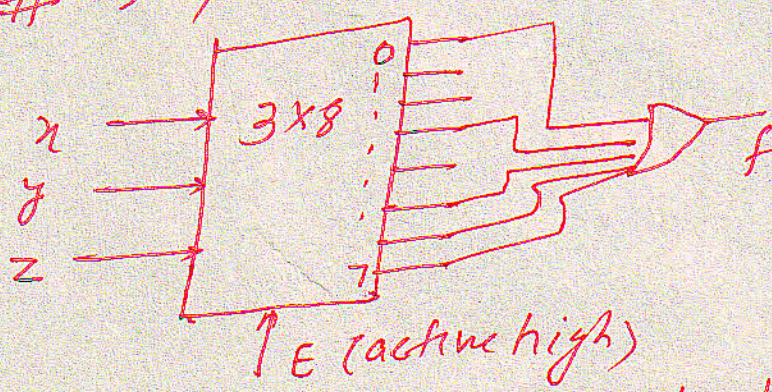
* The Enable E is also help in cascading decoder to form larger decoder

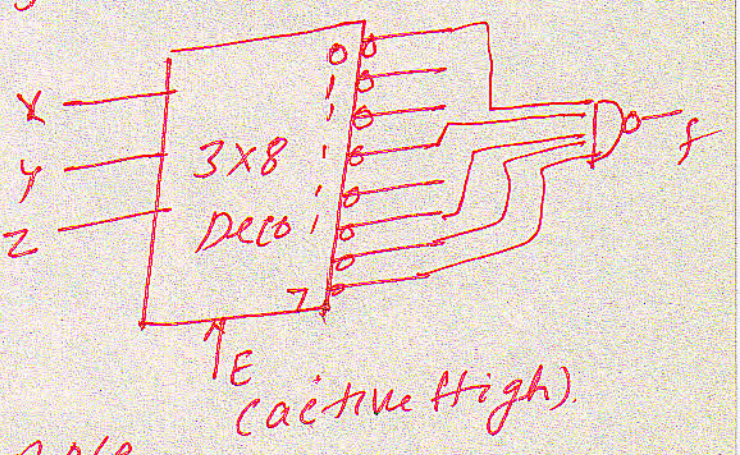
e: 4×16 decoder with 3×8 decoder



* If W = 0 top one is selected and bottom disabled
* Bottom decoder o/p are held at 0 and top generate the min terms. 0000 to 0111
* If W = 1 Bottom is selected with 1000 to 1111

Implement with Decoder $f(x,y,z) = \Sigma(0,3,5,6,7)$





f = Decoder + OR active high o/p

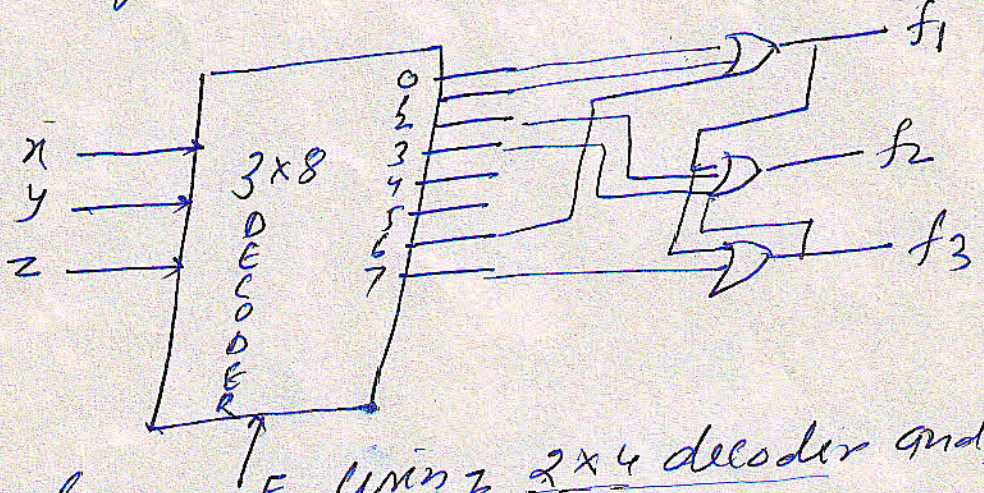
= Decoder + NAND active low o/ps

Implement combinational ckt defined by following function with decoder

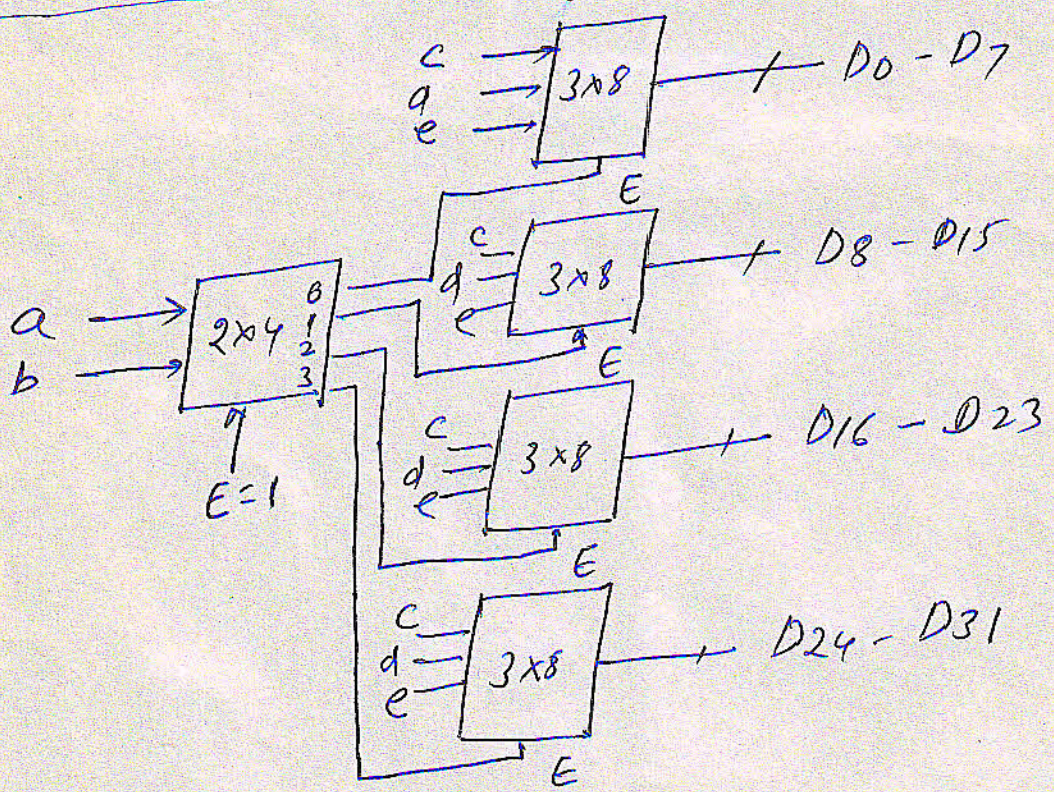
$f_1 = \bar{x}\bar{y} + xy\bar{z} = \Sigma(0, 1, 6)$

$f_2 = \bar{x} + y = \Sigma(0, 1, 2, 3, 6, 7)$

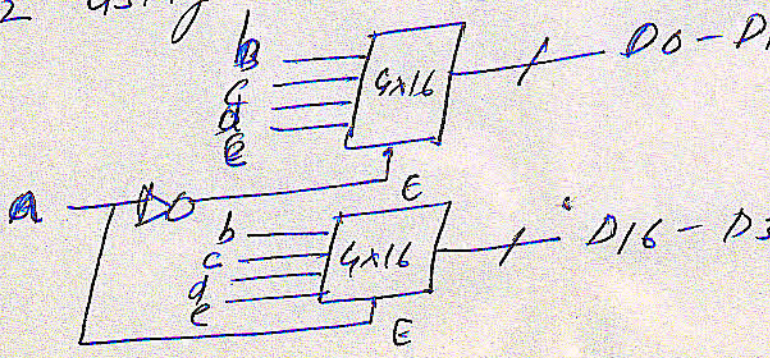
$f_3 = xy + \bar{x}\bar{y} = \Sigma(0, 1, 6, 7)$



5×32 decoder using 2×4 decoder and 3×8



5×32 using 4×16

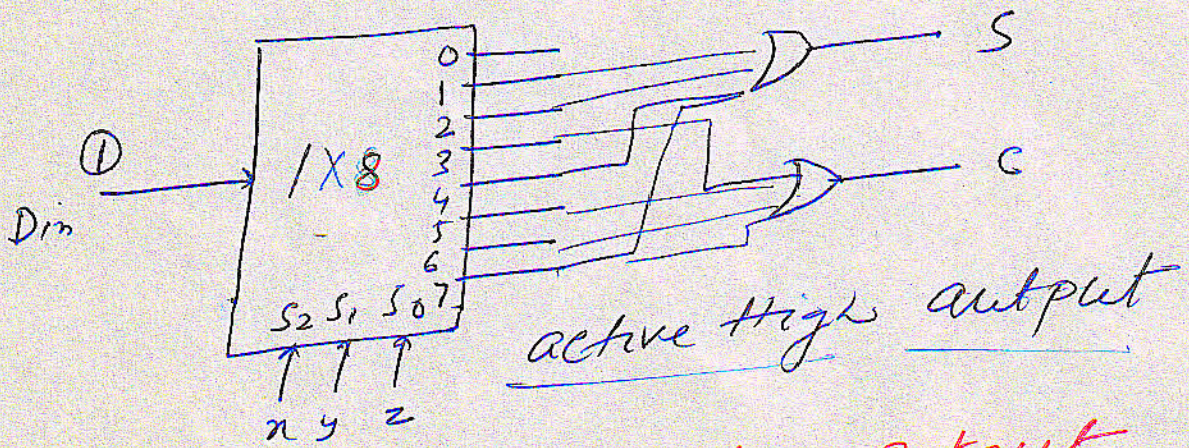


Boolean func with demultiplexer → one to many.

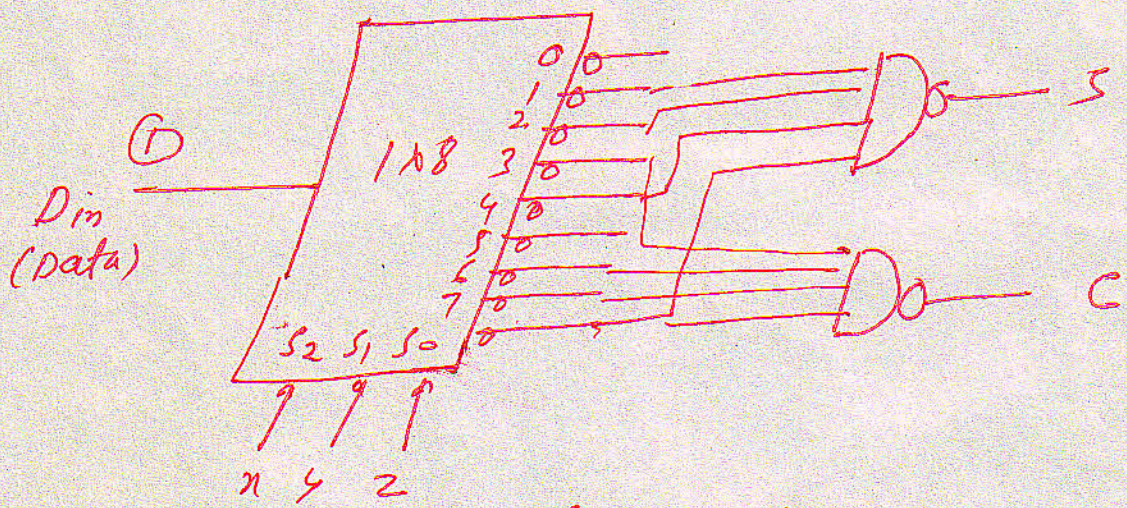
Implement FA

$S = \Sigma(1, 2, 4, 7)$

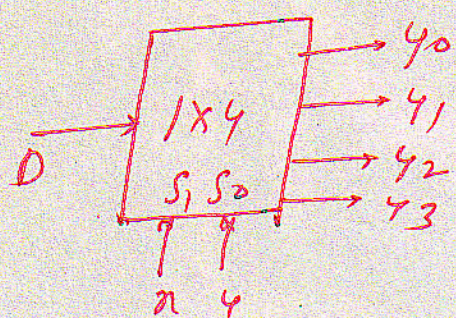
$C = \Sigma(3, 5, 6, 7)$



⇒ Demux with active low output



⇒ function table for De mux



S_1	S_0	Y_0	Y_1	Y_2	Y_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

$Y_0 = \bar{S}_1 \bar{S}_0 \cdot D$

$Y_1 = \bar{S}_1 S_0 \cdot D$

$Y_2 = S_1 \bar{S}_0 \cdot D$

$Y_3 = S_1 S_0 \cdot D$

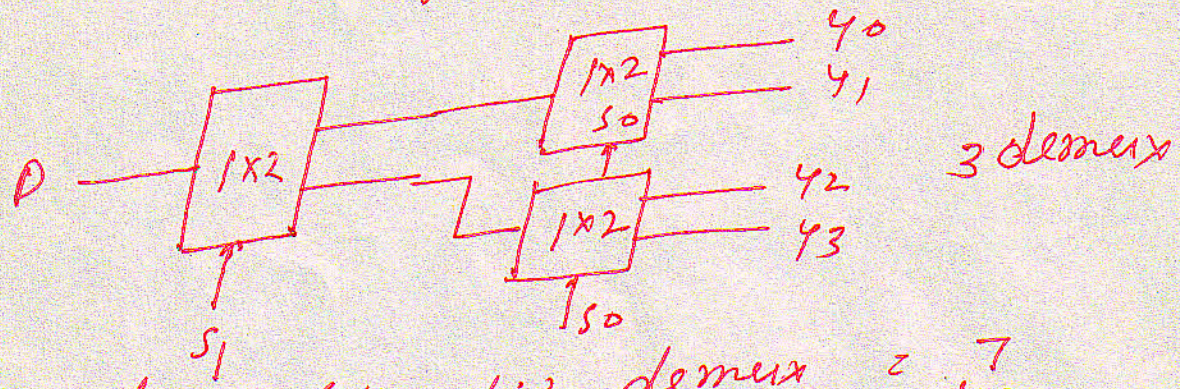
⇒ De mux to decoder → use select lines as input lines and input line as enable I/P

⇒ decoder to demux → use input lines as select lines and enable input as input line

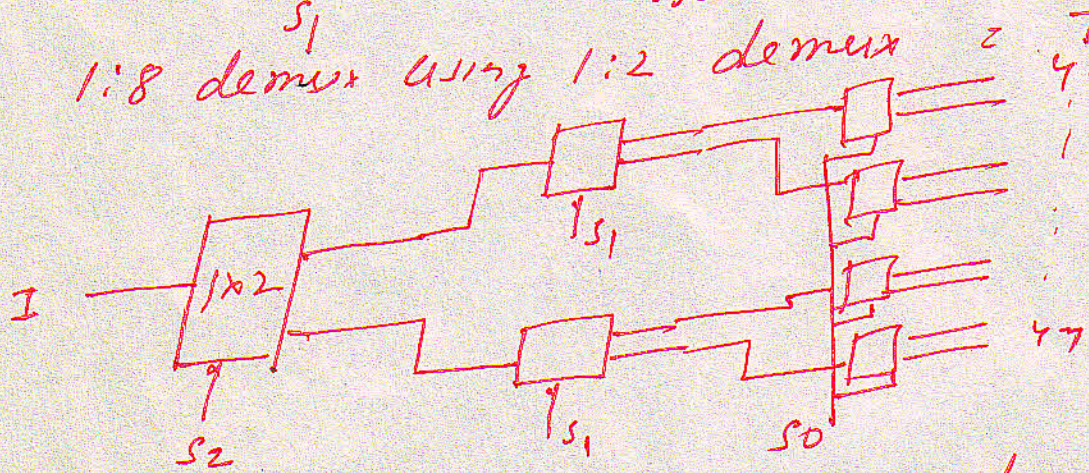
⇒ 1x4 De mux = 2x4 decoder 1x8 de mux : 3x8 decoder

1x16 " " = 4x16 "

⇒ 1:4 demux using 1:2 demux



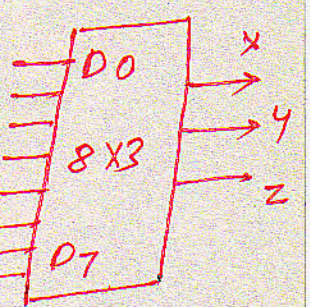
⇒ 1:8 demux using 1:2 demux



⇒ 1:16 using 1:4 Total 5 demux.

Encoder → It does reverse of decoder It has 2^n (or less) input lines and n o/p lines. The o/p lines generate the binary code for 2^n input variable i.e octal to binary encoder

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1



$x = D_4 + D_5 + D_6 + D_7$

$y = D_2 + D_3 + D_6 + D_7$

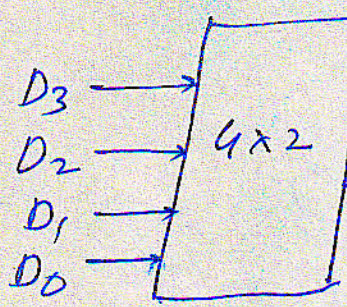
$z = D_1 + D_3 + D_5 + D_7$

* Encoders of this type are not available in IC package. They can be easily constructed with OR logic. The type of encoder available in IC form is called "priority encoder"

* D_0 is not used in any OR Logic. Binary output is all 0's in this case. All 0 o/ps is also obtained when all inputs are 0's. This discrepancy may be resolved by providing one more output to indicate that all inputs are not 0s

* only one input line can be high at any time If two or more input lines are high simultaniously, encoder has no meaning in the sense that it fails to provide an appropriate output

⇒ 4 line to 2 line encoder



D_3	D_2	D_1	D_0	y_1	y_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

for remaining I/P combn output are don't care

$y_1 = D_2 + D_3$, $y_0 = D_1 + D_3$

priority encoder → In priority encoder inputs are don't care not the outputs, for order of priority $D_3 > D_2 > D_1 > D_0$

D_3	D_2	D_1	D_0	y_1	y_0
0	0	0	1	0	0
0	0	1	x	0	1
0	1	x	x	1	0
1	x	x	x	1	1

$y_1 = D_2 + D_3$

$y_0 = D_3 + D_1 \bar{D_2}$

Code Converter — BCD → x-3

d	b_3	b_2	b_1	b_0	E_3	E_2	E_1	E_0
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

$E_1 = b_1 \odot b_0$, $E_0 = \overline{b_0}$

$E_2 = \bar{b}_2 b_0 + \bar{b}_2 b_1 + b_2 \bar{b}_1 \bar{b}_0$

$E_3 = b_3 + b_2 b_0 + b_2 b_1$

Binary → Gray.

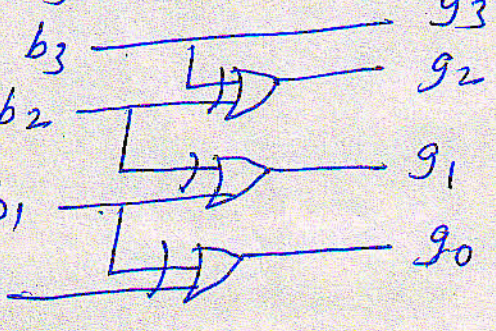
$b_3 b_2 b_1 b_0 \rightarrow g_3 g_2 g_1 g_0$

$g_3 = b_3$

$g_2 = b_3 \oplus b_2$

$g_1 = b_2 \oplus b_1$

$g_0 = b_1 \oplus b_0$



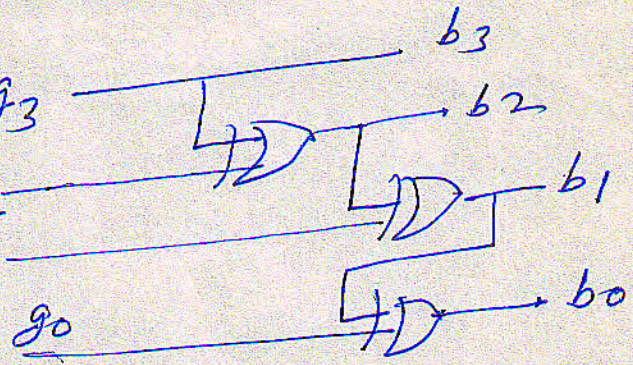
Gray to Binary.

$b_3 = g_3$

$b_2 = g_2 \oplus g_3$

$b_1 = g_1 \oplus g_2 \oplus g_3$

$b_0 = g_0 \oplus g_1 \oplus g_2 \oplus g_3$



BCD to Gray.

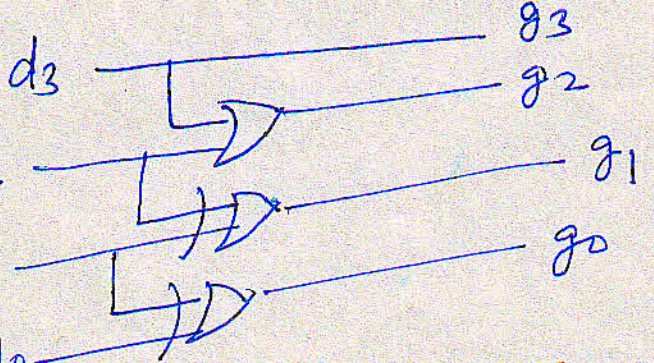
$d_3 d_2 d_1 d_0 \rightarrow g_3 g_2 g_1 g_0$

$g_3 = d_3$

$g_2 = d_2 + d_3$

$g_1 = d_1 \oplus d_2$

$g_0 = d_1 \oplus d_0$



Gray to BCD

$d_3 = g_3$, $d_2 = \bar{g}_3 g_2$, $d_1 = \bar{g}_2 g_1 + \bar{g}_3 g_2 \bar{g}_1$

$d_0 = \bar{g}_1 g_0 + g_1 \bar{g}_0$

$\Rightarrow$

g_3	g_2	g_1	g_0	d_3	d_2	d_1	d_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1

$d_3 = g_3$

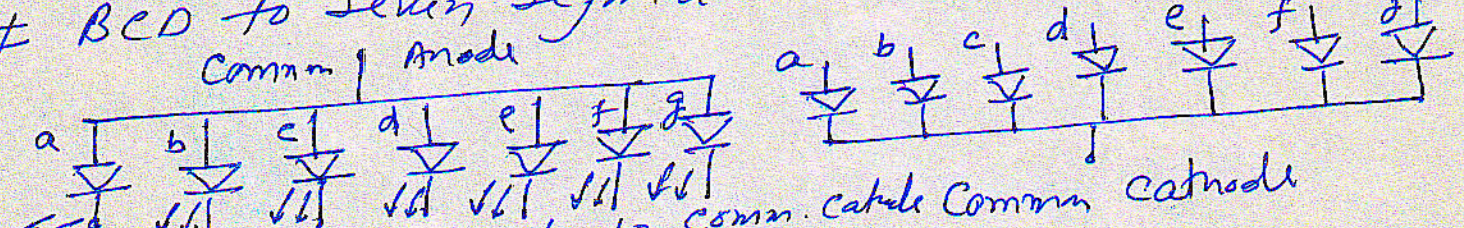
d_2

$\bar{g_3}\bar{g_2}$	$\bar{g_1}\bar{g_0}$	$\bar{g_1}g_0$	g_1g_0	$g_1\bar{g_0}$
$\bar{g_3}\bar{g_2}$	0	0	0	0
$\bar{g_3}g_2$	1	1	1	1
g_3g_2	0	0	x	x
$g_3\bar{g_2}$	x	x	x	x

$= g_3\bar{g_2}$

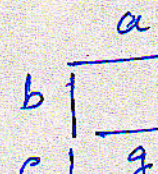
And so on.

BCD to Seven Segment :— $\rightarrow$ Comm Anode
$\hookrightarrow$ Comm cathode



Comm. cathode Comm cathode

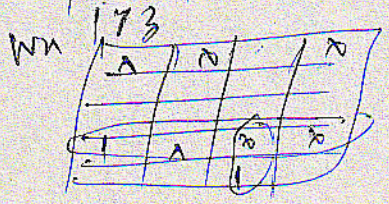
d	b_3	b_2	b_1	b_0	o/p to comm. cathode a	b	c	d	e	f	g
0 →	0	0	0	0	1	1	1	1	1	1	0
1 →	0	0	0	1	0	0	0	0	1	1	0
2 →	0	0	1	0	1	0	1	1	0	1	1
3 →	0	0	1	1	0	0	1	1	1	1	1
4 →	0	1	0	0	0	1	0	0	1	1	1
5 →	0	1	0	1	1	1	1	1	0	1	1
6 →	0	1	1	0	1	1	1	1	0	1	1
7 →	0	1	1	1	0	0	0	0	1	1	0
8 →	1	0	0	0	1	1	1	1	1	1	1
9 →	1	0	0	1	1	1	1	0	1	1	1



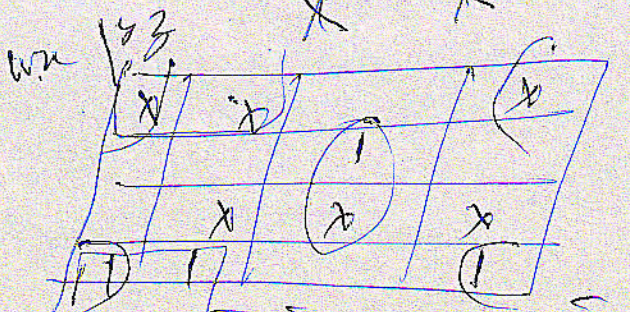
a, b, c, d, e, f, g Can be Solved by K-map.

Excess-3 to BCD Code Converter using Gates (NAND).

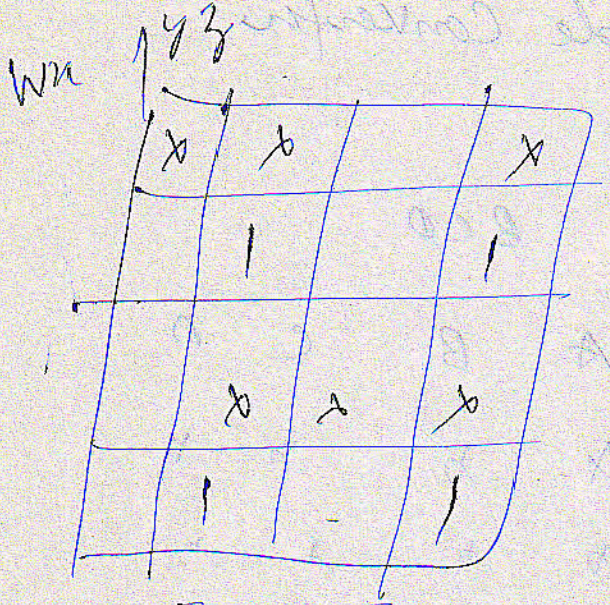
Excess-3				BCD			
W	X	Y	Z	A	B	C	D
0	0	0	0	x	x	x	x
0	0	0	1	x	x	x	x
0	0	1	0	x	x	x	x
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	0	0	0	1	1
0	1	1	1	0	1	0	0
1	0	0	0	0	1	0	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	1	1
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	1
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X



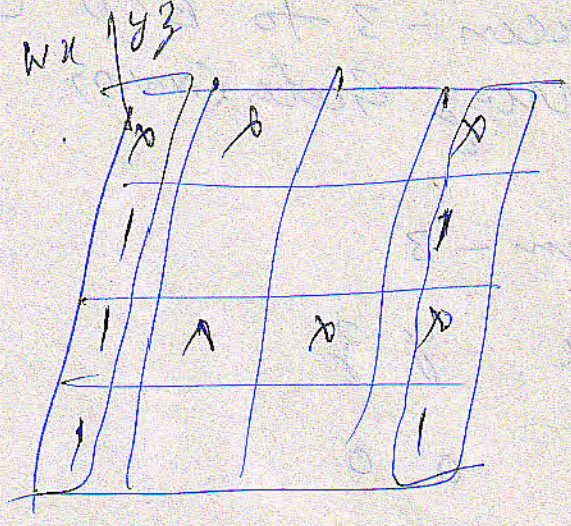
$A = WX + WYZ$



$= X\bar{Y} + XYZ + \bar{X}\bar{Z}$



$C = \bar{y}z + y\bar{z}$



$D = \bar{z}$