Logic Design

Unit - 1

The basic gates - review of basic logic gates, positive & negative logic, introduction to HDL.

Combinational logic circuits - SOPs method, TT to K-map, pairs, quads & octets, K-map simplifn, Don't care cond's, POS methods, POS simplifn by Quine-McCluskey method, Hazards & Hazard covers, HDL implementn models.

Introduction: Digital electronic circuits operate with voltages of 2-logic levels namely logic low & logic high. The range of voltages corresponding to logic low is represented with '0'. Similarly, that with logic is high is '1'.

Logic gate: the basic digital electronic circuit that has one & more inputs & single output. Hence the logic gates are the building blocks of any digital sys.

Classifn of logic gates: 3 categories - basic, universal & special gates.

Basic gates: AND, OR & NOT gates

AND **Universal gates**: NAND, NOR

Special gates: EX-OR, EX-NOR.

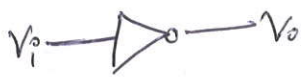
HDL: Hardware Description Language, is used to describe the structure & behaviour of electronic circuits. It is

The basic gates : - NOT , OR , AND

Three logic circuits, the OR, the AND & the inverter (NOT) gates can be used to produce any digital sys.

The Inverter (NOT gate) :

Its 7404 - TTL.



<i/p> <o/p>

Vi	Vo
L	H
H	L

Vi	Vo
0	1
1	0

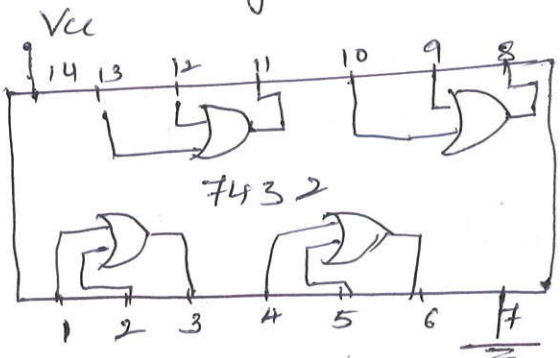
TTL - transistor - transistor logic

Transistor - semiconductor device used to amplify (or) switch electronic signal & electric power

LED - Light Emitting Diode - to indicate a digital signal.

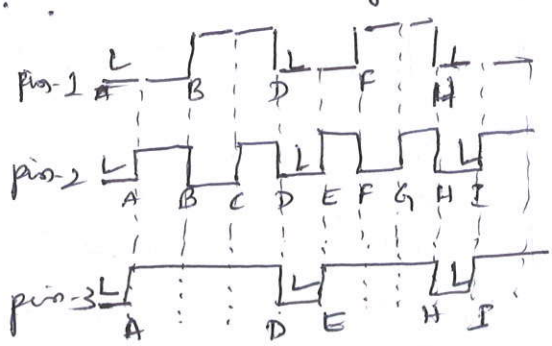
The OR gate:

Fig shows the pin-diagram of a 7432, a TTL quad 2-i/p OR gate. After connecting a supply voltage of +5V to pin-14 & a ground pin to pin-7, you can connect one more OR gates to other TTL devices.



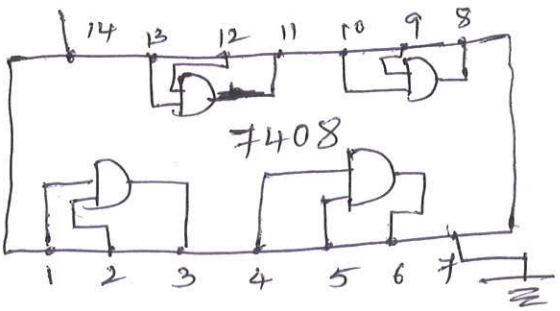
(a) pinout diagram

output (pin-3) is low only when both the inputs are low.

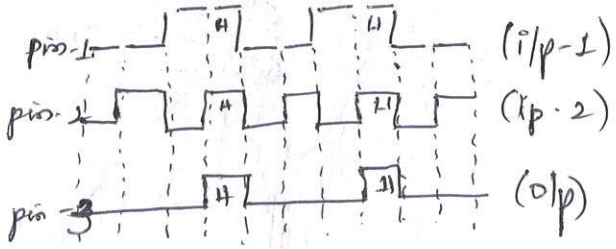


The AND-gate

Fig. shows a pin-out diagram of 7408, a TTL quad-2 i/p AND gate. After connecting a supply voltage of +5V to pin-14 and ground to pin-7, you can connect one of more of AND gates to other TTL device.



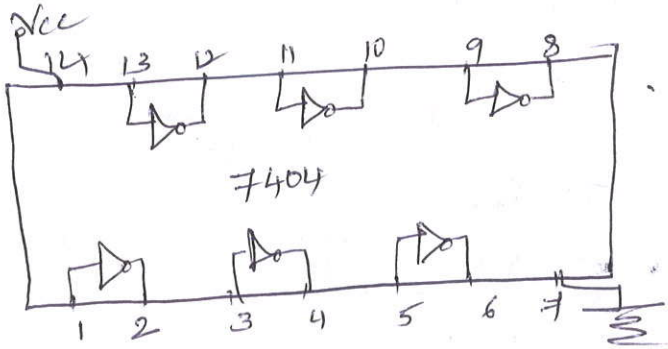
(a) pin-out diagram



o/p is high when both the i/ps are high.

(b) Timing diag

Inverter:



. 6 NOT gates i.e., 6 inverters

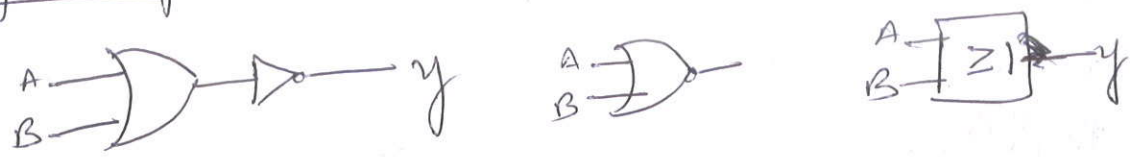
Boolean Algebra:

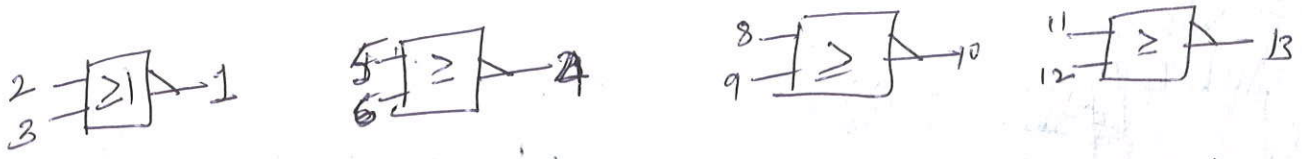
* NOT operation : $y = \bar{A}$ (y equal NOT A)
  $y = 0 = 1$
  $y = 1 = 0$

* OR operation : $y = A + B$
  $y = 1 + 0 = 1$ (where $A = 1$ & $B = 0$)
  $y = 0 + 1 = 1$ ($\because A = 0$ & $B = 1$)

* AND operation : $y = A \cdot B$
  $y = 1 \cdot 0 = 0$
  $y = 0 \cdot 1 = 1$.

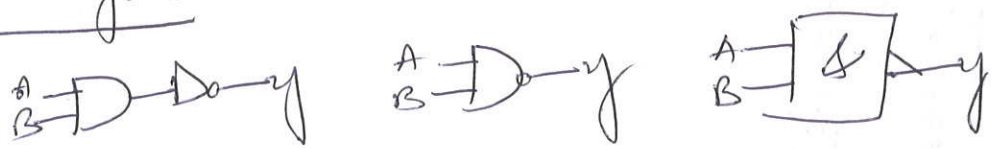
NOR gate Symbol : 7402

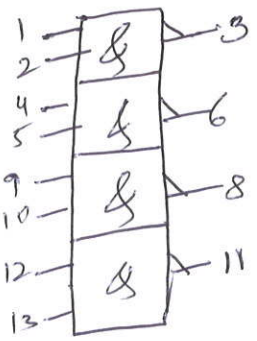




Equation $y = \overline{A + B}$ (y equals NOT A OR B)

NAND gate Symbol :





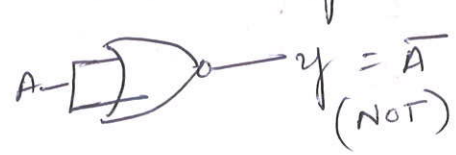
7 - GRD

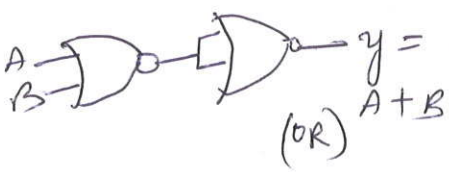
14 - Vcc

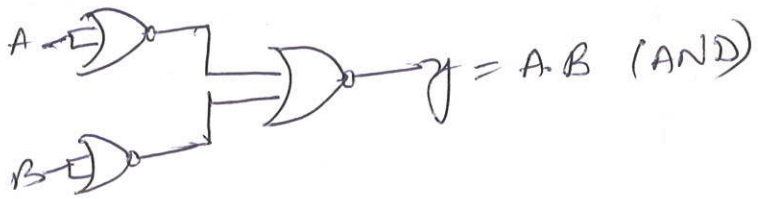
$y = \overline{A \cdot B}$ (y equals NOT A AND B)

Universal logic gates - NOR, NAND :

* Universality of NOR gate
  - All other logic gates can be obtained from NOR gates

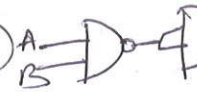
(a) 

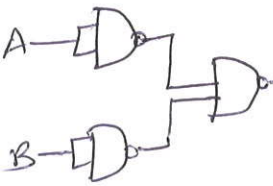
(b) 

(c) 

* Universality of NAND gate
  - All other logic gates can be obtained from NAND gates.

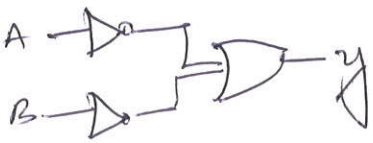
(a)  $y = \overline{A}$ (NOT)

(b)  $y = A \cdot B$ (AND)

(c)  $y = A + B$ (OR)

NOR gate:

* Bubbled OR-gate: fig. shows inverters on the i/p lines of an AND gate.

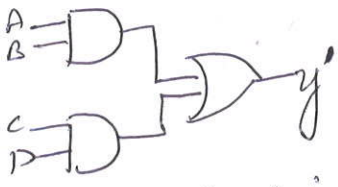


* De Morgan's second theorem.

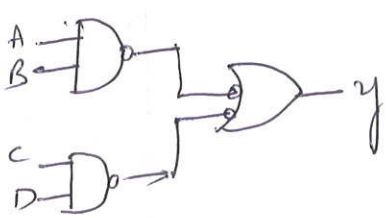
$$\overline{A \cdot B} = \overline{A} + \overline{B}$$

The complement of a product is equal to the sum of the complements.

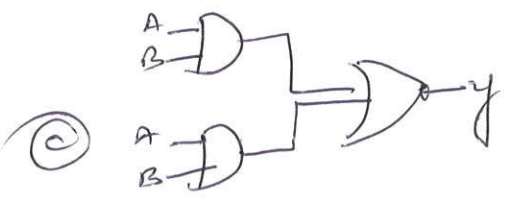
AND-OR-NOT gate:

(a) 

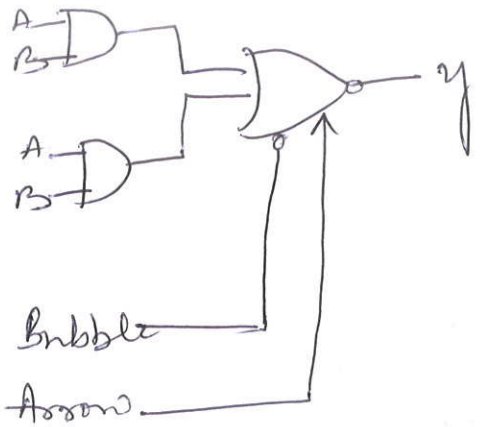
AND-OR circuit

(b) 

NAND-NAND circuit

(c) 

* Expandable AND - OR - NOT gate :



The widest AND - OR - NOT gate available in 7400 series is 4-wide. The 2 additional i/ps, labeled bubble & arrow allow it for 6 or 8-wide circuit.

Positive & Negative logic

Use of binary 0 for low-voltage & a binary 1 for high voltage is called positive logic.

Use of binary 1 for low & binary 0 for high voltage is called negative logic.

Don't care conditions in logic design

- In some digital systems, certain i/p cond^n never occurs during normal oper^n; therefore, the corresponding output never appears. Since the o/p never appears, it is indicated by X in the TT. The X is called a don't-care condition. The designer can assign either 0 (or) 1 to the o/p, whichever is optimal, & produces a simpler logic circuit. Thus don't care cond^{n}s provide flexibility.

Eg: Odd-parity bit generator for (Binary Coded Decimal) BCD (0-9).

w	x	y	z	parity bit	
0	0	0	0	1	
0	0	0	1	0	
0	0	1	0	0	
0	0	1	1	1	
0	1	0	0	0	
0	1	0	1	1	
0	1	1	0	1	
0	1	1	1	0	
1	0	0	0	0	
1	0	0	1	1	
1	0	1	0	-	X
1	0	1	1	-	
1	1	0	0	-	X
1	1	0	1	-	
1	1	1	0	-	
1	1	1	1	-	X

$p(w, x, y, z) = \Sigma m(0, 3, 5, 6, 9) + dc(10, 11, 12, 13, 14, 15)$

The normal odd bit parity generator is,

$$P(w,x,y,z) = \Sigma m(0,3,5,6,9,10,12,15)$$

$$= \bar{w}\bar{x}\bar{y}\bar{z} + \bar{w}\bar{x}yz + \bar{w}x\bar{y}z + w\bar{x}\bar{y}z + \bar{w}xy\bar{z} + w\bar{x}y\bar{z} + wx\bar{y}\bar{z} + wxyz$$

$$= \bar{w}\bar{x}(\bar{y}\bar{z} + yz) + \bar{w}x(\bar{y}z + y\bar{z}) + w\bar{x}(\bar{y}z + y\bar{z}) + wx(\bar{y}\bar{z} + yz)$$

$$= \bar{w}\bar{x}(\overline{y \oplus z}) + \bar{w}x(y \oplus z) + w\bar{x}(y \oplus z) + wx(\overline{y \oplus z})$$

$$= (\overline{y \oplus z})(\bar{w}\bar{x} + wx) + y \oplus z(\bar{w}x + w\bar{x})$$

$$= \overline{(y \oplus z)}\,\overline{(w \oplus x)} + y \oplus z\,(w \oplus x)$$

$$\Rightarrow (w \oplus x)(y \oplus z) + \overline{(w \oplus x)}\,\overline{(y \oplus z)}$$

$\therefore \bar{A}\bar{B} + AB \Downarrow \overline{A \oplus B}$

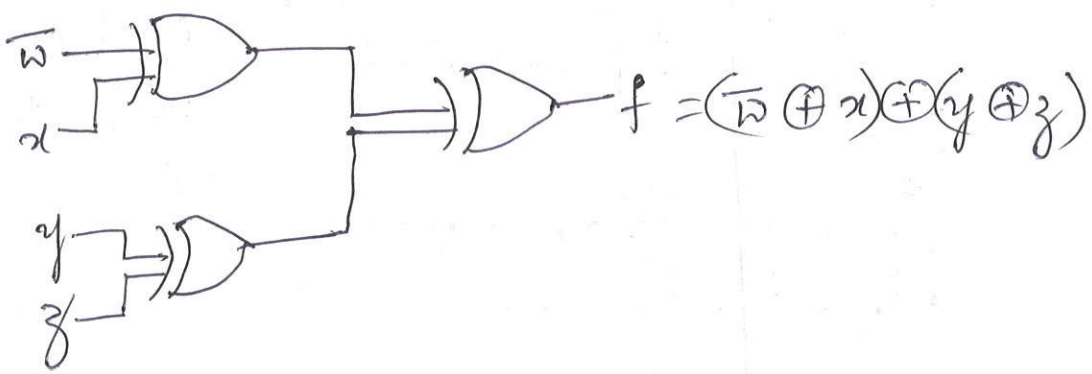
$$= \overline{(w \oplus x) \oplus (y \oplus z)}$$

$\therefore \overline{A \oplus B} = \bar{A} \oplus B$

$$= \overline{w \oplus x} \oplus (y \oplus z)$$

$$\boxed{P(w,x,y,z) = (\bar{w} \oplus x) \oplus (y \oplus z)}$$

Realization of odd-bit parity generator using only XOR gates



Using k-map for single-1:

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

Without don't care cond^n,

$f = A\bar{B}\bar{C}D$ → ⟨i⟩

Now with don't care cond^n,

$f = AD$ → ⟨ii⟩

Realiz^n of ⟨i⟩ & ⟨ii⟩

Product of Sum method: POS

→ Given a TT, you identify the f^{n}damental sum needed for a logic-design.

→ Then by ANDing these sums, you get the POS equ^n corresponding to the TT.

→ With SOP or POS, the f^{n}damental sum produces an o/p '0' for the corresponding i/p cond^n.

A	B	C	Y	
0	0	0	0	→ $A+B+C$
0	0	1	1	
0	1	0	1	
0	1	1	0	→ $A+\bar{B}+\bar{C}$
1	0	0	1	
1	0	1	1	
1	1	0	0	→ $\bar{A}+\bar{B}+C$
1	1	1	1	

Locate each o/p '0' in the TT & write down its fundamental sum.

Conversion b/w SOP and POS:

- We have seen that SOP represent" is obtained by considering "ones" in the TT while POS comes considering "zeroes".
- In SOP, each dip one @ output gives one AND term which is finally ORed.
- → In POS', each zero gives one OR term which is finally ANDed.
- → Thus SOP and POS occupy complementary locat's in a TT and one represent" can be obtained from the other by
  - ↳ identifying complementary bc"s
  - ↳ changing min-term to max-term (or) reverse & finally
  - ↳ changing summation by product (of) reverse.

POS Simplificat"

→ Convert the TT into k-map. After grouping the 1s, write the SOP equat" & draw the NAND-NAND circuit. This is the SOP for 'y'.

→ Complement the k-map. Group the 1s, write the SOP equat", & draw the NAND-NAND circuit for $\bar{y}$. This is the complementary NAND-NAND circuit.

→ Convert the complementary NAND-NAND circuit to a dual NOR-NOR circuit by changing all NAND gates to NOR gates & complementing all signals. What remains is the POS solⁿ for 'y'.

→ Compare the NAND-NAND ckts with the NOR-NOR ckt (step-3). One can use whichever ckt you prefe,

Simplification by Quine - McCluskey Method

Algorithm:

1) List all minterms in the binary form.
2) Arrange the minterms according to number of 1's.
3) Compare each binary no with every term in the adjacent next higher category and if they differ only by one position, put a check mark & copy the term in the next column with "–" in the position that they differed.
4) Apply the same process described in step-3 for the resultant column & continue these cycles until a single pass through cycle yields no further elimination of literals.
5) List all the prime-implicants.
6) Select the minimum no of prime implicants which must cover all the minterms.

Eg: 1) Simplify the foll Boolean f^n's by using Quine McClusky method. $f(A,B,C,D) = \Sigma m(0,2,3,6,7,8,10,12,13)$

Soln:

Step-1: List all minterms in binary form as column (a)

Step 2: Arrange minterms according to no of 1's, as in col-(b)

minterm	Bin-representn
m_0	0 0 0 0
m_2	0 0 1 0
m_3	0 0 1 1
m_6	0 1 1 0
m_7	0 1 1 1
m_8	1 0 0 0
m_{10}	1 0 1 0
m_{12}	1 1 0 0
m_{13}	1 1 0 1

col-(a)

minterm	binary represent
m_0	0 0 0 0 ✓
m_2	0 0 1 0 ✓
m_8	1 0 0 0 ✓
m_3	0 0 1 1 ✓
m_6	0 1 1 0 ✓
m_{10}	1 0 1 0 ✓
m_{12}	1 1 0 0 ✓
m_7	0 1 1 1 ✓
m_{13}	1 1 0 1 ✓

col-(b)

→ P.T.O

Quine M

Step-3: Compare each binary no with every term in the adjacent next higher category and if they differs only by one position, put a check mark and copy the term in the next column with '–' in the position that they differed.

Step 4: Apply the same process as in Step-3 for the resultant columns & continue these cycles until no elimn of literal occurs.

min-term	Bin-repn	
0, 2	0 0 – 0	✓
0, 8	– 0 0 0	✓
2, 3	0 0 1 –	✓
2, 6	0 – 1 0	✓
2, 10	– 0 1 0	✓
8, 10	1 0 – 0	✓
8, 12	1 – 0 0	
3, 7	0 – 1 1	✓
6, 7	0 1 1 –	✓
12, 13	1 1 0 –	

col-(C)

min-term	Bin-repn
0, 2, 8, 10	– 0 – 0
2, 3, 6, 7	0 – 1 –

col-(d)

Step-5: List the prime implicants

Prime-implicants		Bin-repn
$A\bar{C}\bar{D}$	8, 12	1 – 0 0
$AB\bar{C}$	12, 13	1 1 0 –
$\bar{B}\bar{D}$	0, 2, 8, 10	– 0 – 0
$\bar{A}C$	2, 3, 6, 7	0 – 1 –

Step-6: Select the min-no of prime-implicants which must cover all the min-terms.

Prime Implicants	m_0 (col-1)	m_2 (col-2)	m_3 (col-3)	m_6 (col-4)	m_7 (col-5)	m_8 (col-6)	m_{10} (col-7)	m_{12} (col-8)	m_{13} (col-9)
$A\bar{C}\bar{D}$ (8,12)									
$AB\bar{C}$ (12,13) ✓								⊙	⊙
$\bar{B}\bar{D}$ (0,2,8,10) ✓	⊙	⊙				⊙	⊙		
$\bar{A}C$ (2,3,6,7) ✓		⊙	⊙	⊙	⊙				

Search for single dot columns & select the prime implicants corresponding to that dot by putting the check mark in front of it. Search for multi-dot columns one-by-one. If the corresponding min-terms is already included in the final expression, ignore the min-terms & go to next-dot column.

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

Quine McCluskey using don't care terms

Unit - 2

Data - Processing Circuits

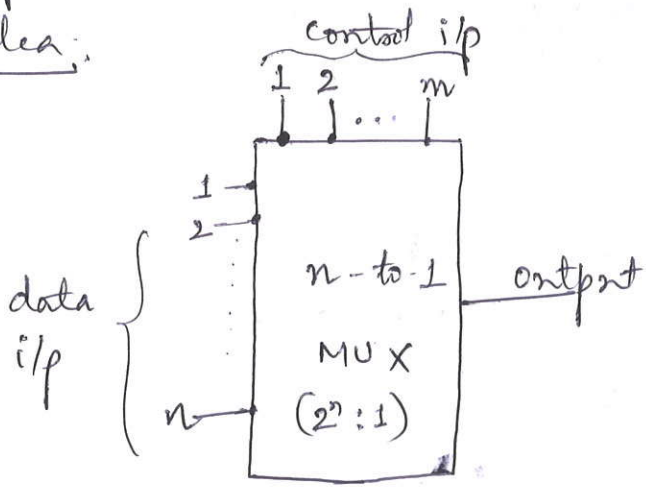
- Logic ckts that process binary data

Eg: MUX, De-Mux, 1-of-16 decoder, BCD to decimal decoders, Seven segment decoders, encoders, EX-OR gates, parity checkers/generators, magnitude comparators, read-only memories, PLAs & PALs.

Multiplexers: - data selector

- means many into one.
- is a ckt with many i/ps ~~bay~~ but only 1-o/p. By applying control signals, we can steer any i/p to the o/p. Thus it is also called a data selector & control i/ps are termed as select i/ps.

General idea:



The ckt has n - i/p signals
m - control signal
& 1 - o/p signal

Note: m-control signals can select atmost 2^m i/p signals that $n \leq 2^m$.

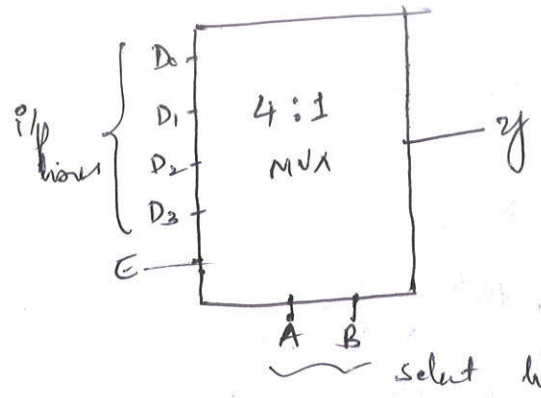
i.e., 2^n-to-1 MUX

Ckt

Logic

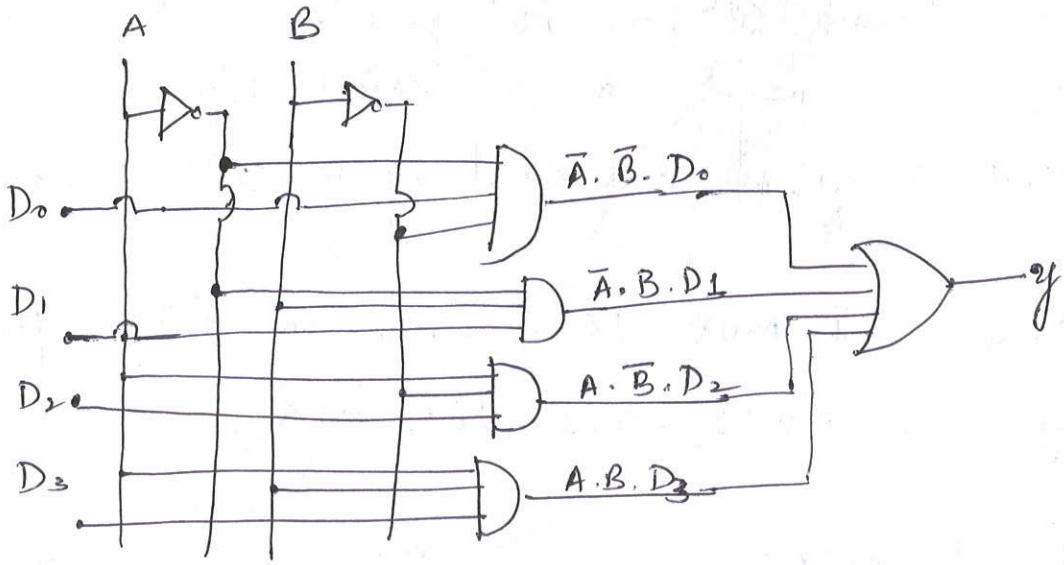
In oth
which
& al
@ lo
D

Ckt diagram: 4-to-1 MUX



Fn Table

A	B	y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3



Logic equation of the above ckt:

$$y = \bar{A}\bar{B}.D_0 + \bar{A}.B.D_1 + A\bar{B}.D_2 + AB D_3$$

If A=0 and B=0

$$= \bar{0}.\bar{0}.D_0 + \bar{0}.0.D_1 + 0.\bar{0}.D_2 + 0.0\,D_3$$

$$= 1.1\,D_0 + 1.0.D_1 + 0.1.D_2 + 0.0\,D_3$$

$$= 1.D_0 + 0 + 0 + 0$$

$$\boxed{y = D_0}$$

In other words, for AB=00, the first AND gate to which D_0 is connected remains active & equal to D_0 & all other AND gates are in-active with o/p held @ logic 0. Thus, MUX o/p 'y' is same as D_0.

Illy, for AB=01, 2nd AND gate will be active & all other AND gates remain inactive. Thus o/p y=D1. Following same procedure we can complete the TT.

If we want 5:1 MUX, then how many select lines are reqd.?

- There is no 5th combin^n possible with 2 select lines & hence we need a 3rd select i/p.
- With 3 select i/p, we can select upto $2^3 = 8$ data i/prs.
- Commercial MUX ICs come in integer power of 2, Eg: 2:1, 4:1, 8:1, 16:1 MUX.

16-to-1 MUX:

The i/p bits are labelled D_0 to D_{15}. Only one of these is transmitted to o/p, which one depends on the value of control i/p ABCD.

Eg: When ABCD = 0000, the upper AND gate is enabled while all other AND gates are disabled. Therefor, data bit D_0 is transmitted to the o/p giving $y = D_0$. Similarly $y = D_{15}$, when ABCD = 1111.

The 74150 is a 16-to-1 TTL MUX.

Pins 1-to 18 and 16-23 - are data lines
Pins 11, 13, 14, 15 are control lines - ABCD.
Pin 10 is the o/p.

A low-
if equal

On the
& forces
the val

Strobe/ Enable	
L	
L	
L	
.	
.	
.	
L	
H	

Bubbles

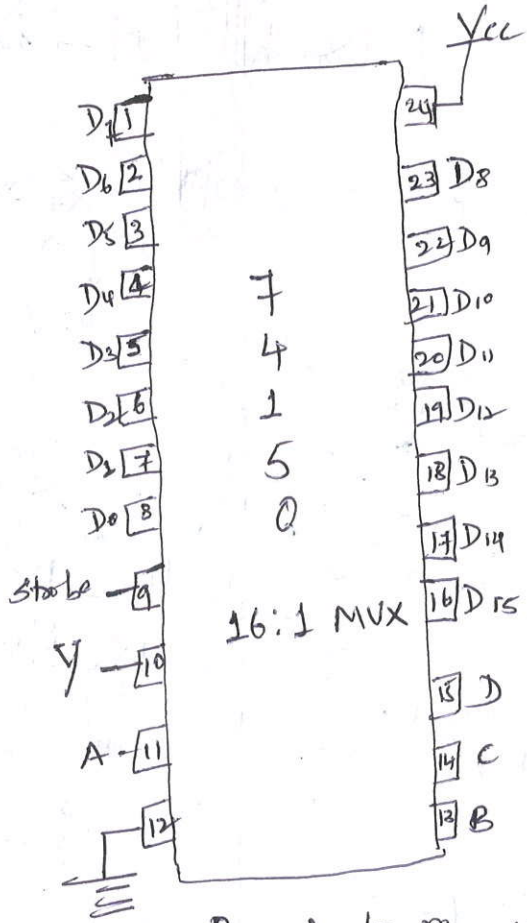
A low-strobe enables the multiplexer ($\overline{E}$), so that o/p 'y' equals the complement of the i/p-data bit:

i.e., $y = \overline{D_n}$ where n = decimal equivalent of ABCD.

On the other hand, a high strobe disables the MUX & forces the o/p into the high state. With a high strobe, the value of ABCD does NOT matter.

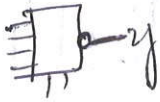
Strobe/ Enable	A	B	C	D	Y
L	L	L	L	L	$\overline{D_0}$
L	L	L	L	H	$\overline{D_1}$
L	L	L	H	L	$\overline{D_2}$
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
L	H	H	H	H	$\overline{D_{15}}$
H	X	X	X	X	H

74150 Truth Table

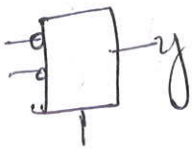


Pinout diagram of 74150

Bubbles on Signal lines:



y ⇒ is the o/p that is the complement of the selected data-bit.



y ⇒ bubble on i/p pin, means the signal is active low.

Universal logic ckt:

* MUX is sometimes called as universal logic ckt because 2^n-to-1 MUX can be used as design soln for any n-vari TT.

*Eg: TT can be realized using 8-to-1 MUX.

$f(A, B, C, D)$

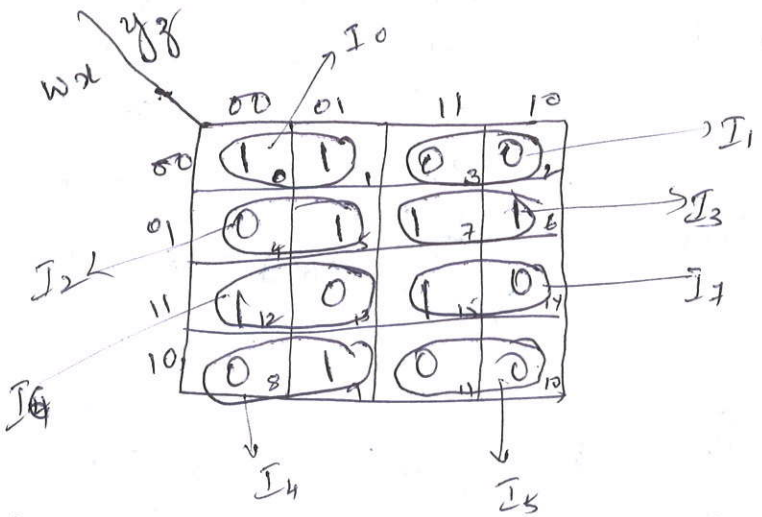
Let's consider A, B & C varis → to be fed as Select lines

fourth var D → has to be present as a data line.

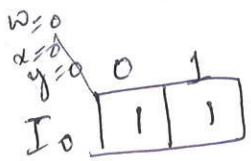
Eg: ① $f(w, x, y, z) = \Sigma m(0, 1, 5, 6, 7, 9, 12, 15)$

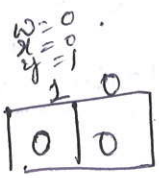
↓

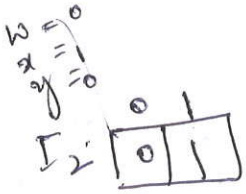
write this in the form of TT (or) K-map.

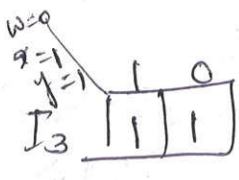


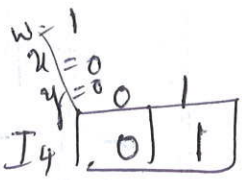
Submaps:

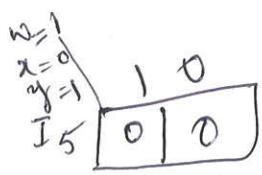


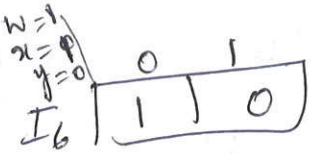


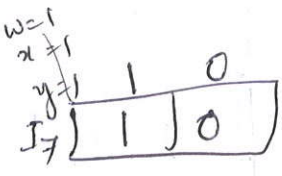












$I_0 =$

$I_1 =$

$I_2 =$

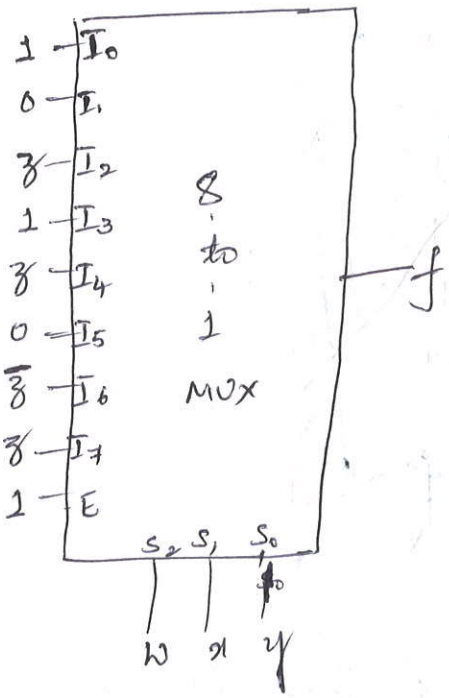
② Realizo

I_0 Sub

I_1 Su

I_2 Sub

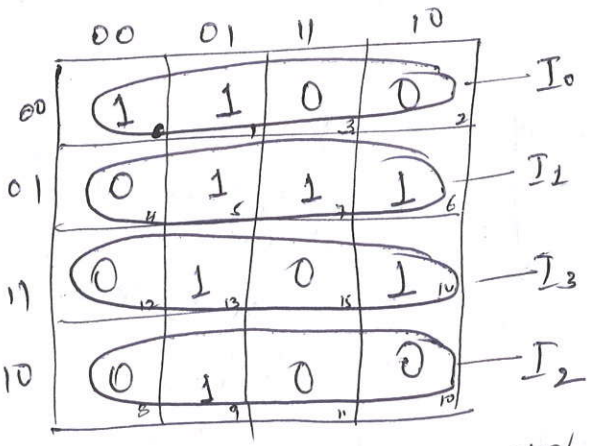
$I_0 = \bar{z} + z = 0 + 1 = 1$
$I_1 = 0$
$I_2 = z$
$I_3 = z + \bar{z} = 1$
$I_4 = z$
$I_5 = 0$
$I_6 = \bar{z}$
$I_7 = z$
$E = 1$.



(2)

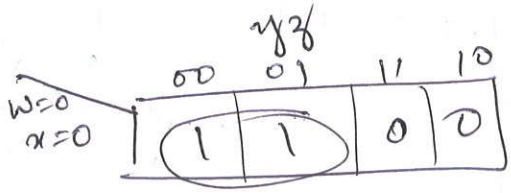
Realizn of 4-var boolean fn using 4 to 1 MUX

$f(w, x, y, z) = \Sigma m(0, 1, 5, 6, 7, 9, 13, 14)$



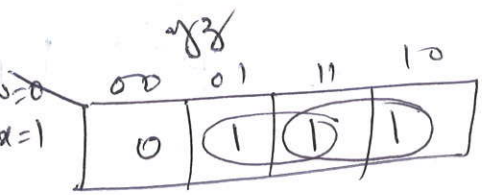
wx — select lines
yz - data lines

I_0 submap:



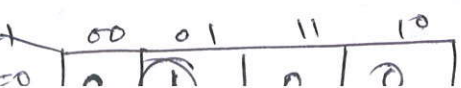
$I_0 = \bar{y}$

I_1 submap:



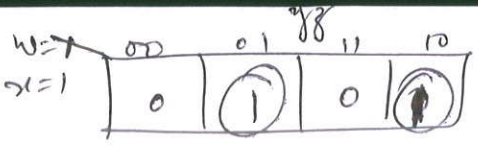
$I_1 = z + y$

I_2 sub map:



$I_2 = \bar{y}z$

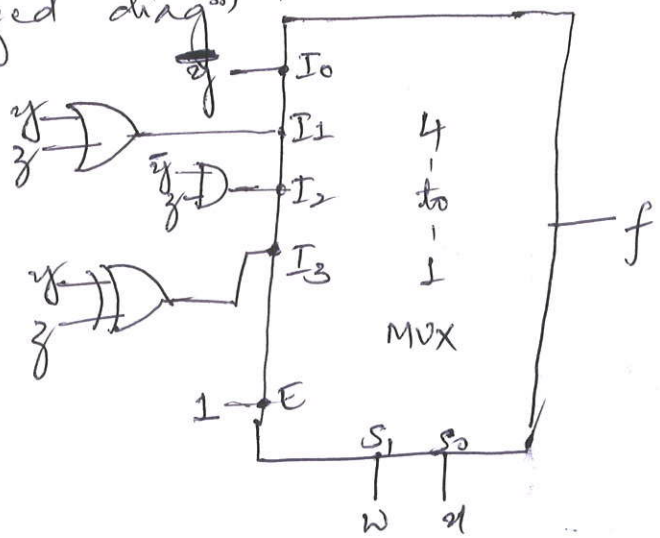
I_3 sub map:



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

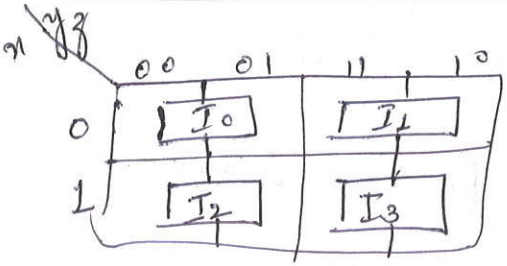
$I_3 = y \oplus z$

Realized diagram:



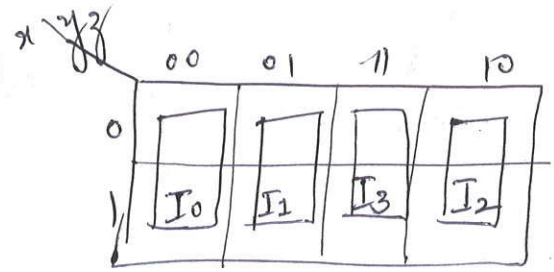
Realization of 3 vari boolean fn using 4-to-1 MUX

(i) 2 ways: (1)



⇒ x y-select lines
z - data line

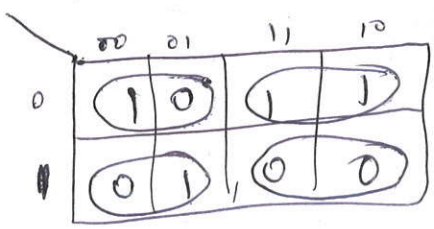
(2)



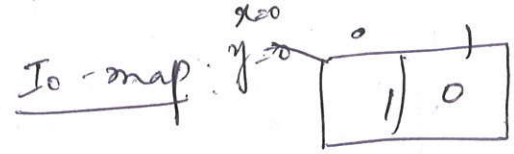
⇒ yz — select lines
x — data line

Eg: First method:

$f(x, y, z) = \Sigma m(0, 2, 3, 5)$

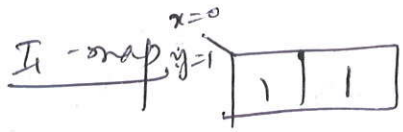


I_0-map:



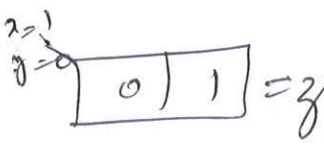
$\bar{z}$

I_1-map:



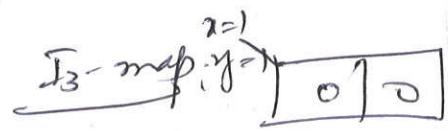
$= 1$

I_2-map:



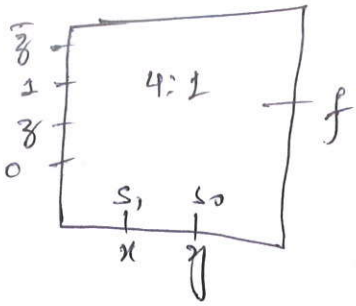
$= z$

I_3-map:

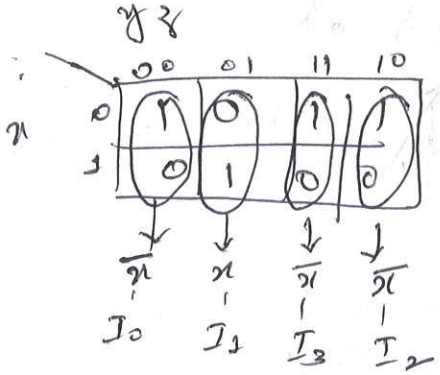


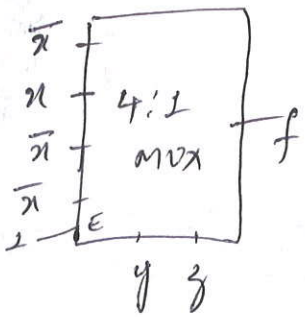
$= 0$

$I_0 = \bar{z}$ $I_1 = 1$ $I_2 = z$ $I_3 = 0$



Second method:





Nibble MUX:

→ In computing (&) digital, nibble means combin" of 4-bit. Eg: 0010 (or) 0001 etc/..

→ Sometimes, we want to select one of 2 i/p nibbles. So use a nibble-MUX.

→ Input on left is $A_3.A_2.A_1.A_0$ & on right is $B_3.B_2.B_1.B_0$. The control signal labeled SELECT det which i/p-nibble is transmitted to the o/p.

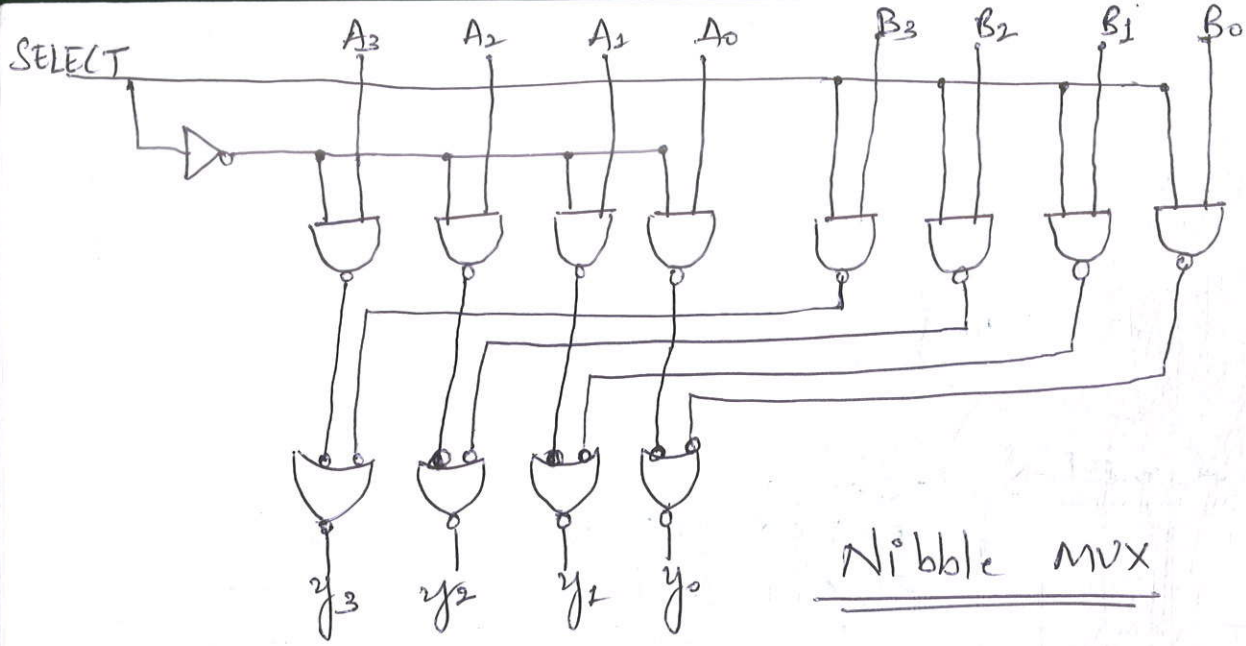
→ When SELECT = 0, 4-NAND gates on LEFT are activated.

i.e., $y_3.y_2.y_1.y_0 = A_3.A_2.A_1.A_0$

→ When SELECT = 1, 4-NAND gates on RIGHT are activated.

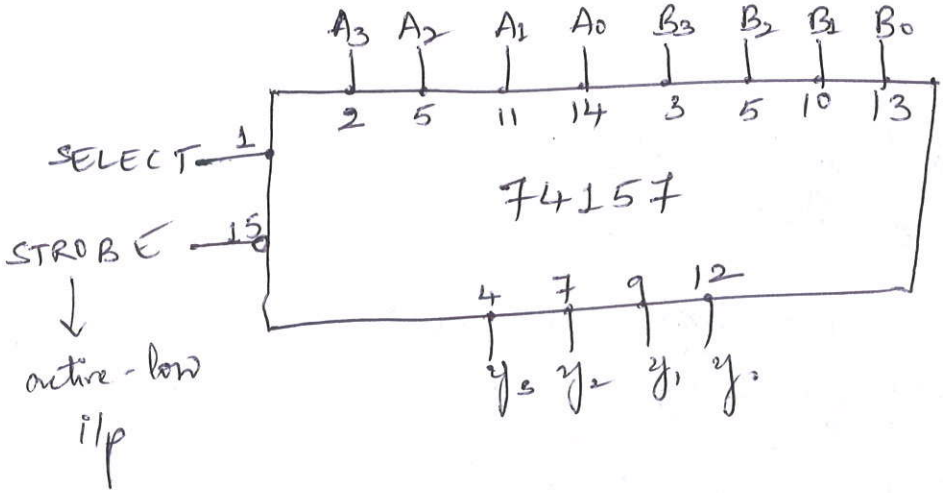
i.e., $y_3.y_2.y_1.y_0 = B_3.B_2.B_1.B_0$

→ i.e., when SELECT = 0, left nibble is steered to the o/p, else right nibble is steered when it is high.



Nibble MUX

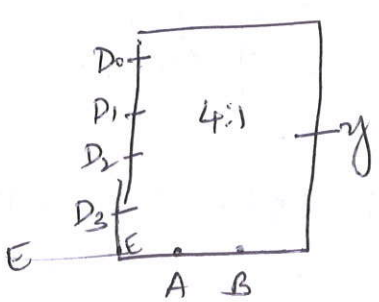
Pin diagm: 74157

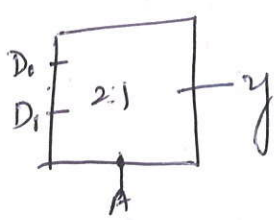


Worked Examples:

1) Show how 4:1 MUX can be obtained using only 2:1 MUX

Soln:





Logic eqn for 2:1 MUX : $y = \bar{A} \cdot D_0 + A \cdot D_1 \rightarrow (1)$

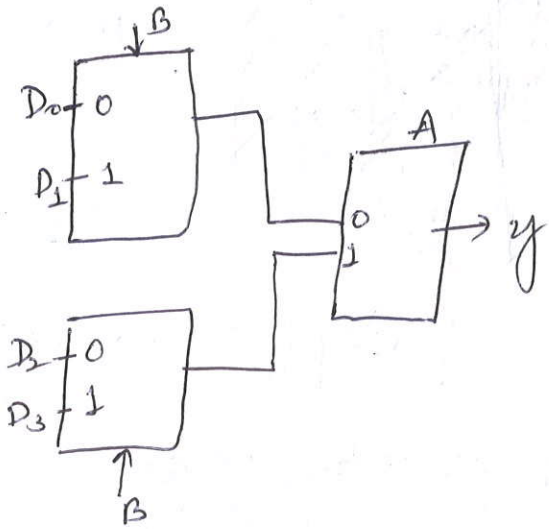
Logic eqn for 4:1 MUX : $y = \bar{A} \cdot \bar{B} \cdot D_0 + \bar{A} \cdot B \cdot D_1 + A \cdot \bar{B} \cdot D_2 + A \cdot B \cdot D_3$

The logic eqn of 4:1 MUX can be re-written as

$$y = \bar{A}(\bar{B}D_0 + BD_1) + A(\bar{B}D_2 + BD_3) \longrightarrow (2)$$

Compare eqn (1) & (2).

We need ~~a~~ two 2:1 MUX to realize the two bracketed terms where B serves as select i/p. The o/p these 2 MUXs can be sent to a 3rd MUX as data i/ps where A serves as select i/p & we get 4:1 MUX.

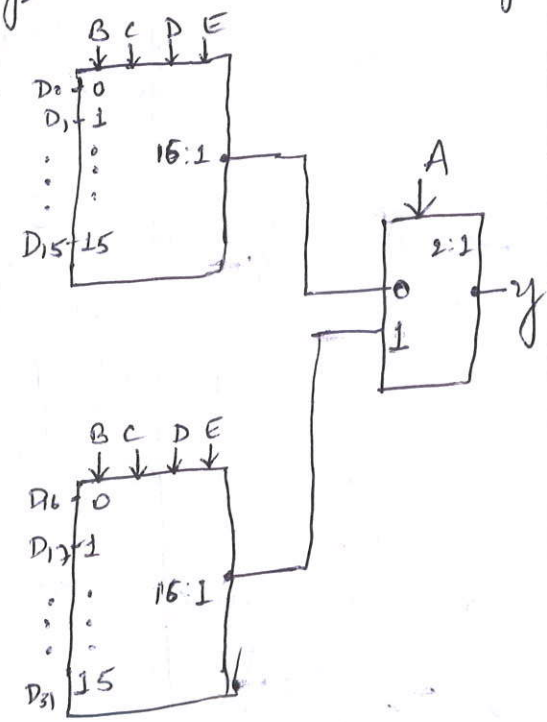


(2) Design a 32-to-1 MUX using two 16:1 MUX & one 2-to-1 MUX.

Soln: A 32-to-1 MUX requires $\log_2 32 = 5$ select lines say, ABCDE.

⇒ 4 select lines BCDE chose 16-to-1 MUX o/ps.

⇒ 2-to-1 MUX chooses one of the o/p of 2 16-to-1 MUX depending on what appears in the 5th select line. A



using

→ (1)

B·D_1 +

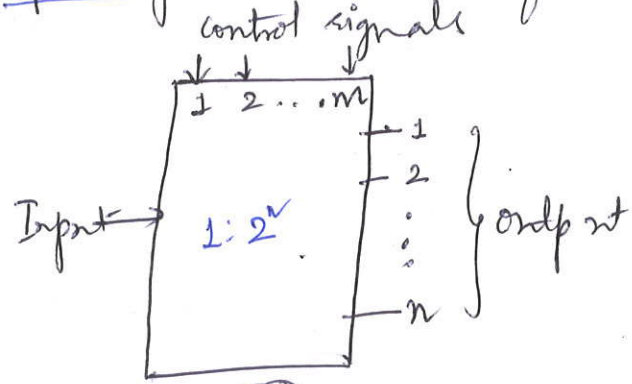
B·D_3

Demultiplexers :

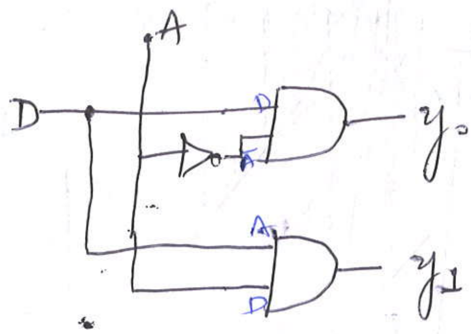
→ Demultiplex means one into many.

→ A demultiplexer is a logic circuit with one i/p & many o/ps.

→ By applying control signals, we can steer the i/p signal to one of the o/p lines.



(a) Block diag^m



$1:2^1 \Rightarrow \bar{A}D + AD$

(b) Logic ckt of 1-to-2 MUX

~~For ... truthtable? ICs//~~

<u>1-to-16 MUX</u> : 74154 is a 1-to-16 DeMUX

- The ckt has 1 i/p signal
  m - control (or) select signals
  n - o/p signals,
  where $n \leq 2^m$

→ The i/p bit is labeled - D
This data bit (D) is transmitted to the data bit of the o/p lines. This o/p depends on the value of ABCD, the control i/p.

→ When ABCD = 0000, the upper AND gate is enabled while all other AND gates are disabled.
∴ D is transmitted to y_0 o/p, giving $y_0 = D$.

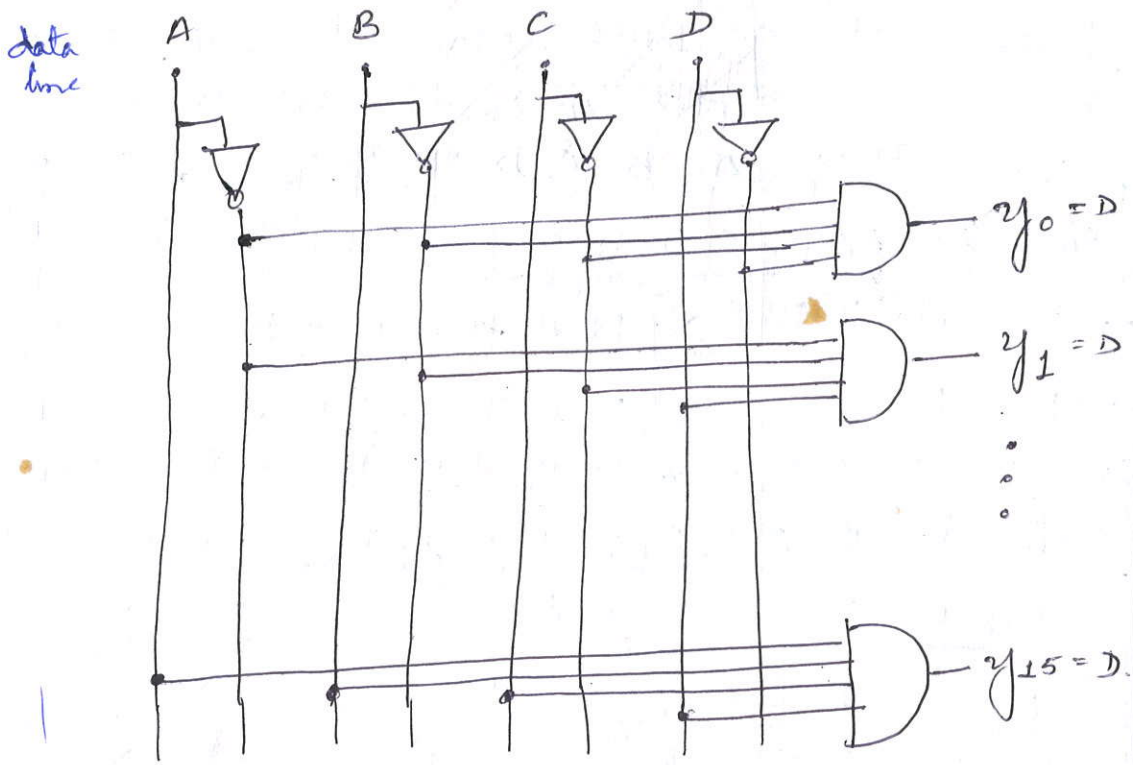
→ If
if
i.e.
All

→ If th
all g
& n

List of

→ If D is low, y_0 is low
if D is high, y_0 is high.
i.e., the value of y_0 depends on the value of D.
All other o/ps are in the low-state.

→ If the control-nibble is changed to ABCD = 1111$_2$,
all gates are disabled except bottom AND gate,
& resulting y_{15} o/p, i.e., $y_{15} = D$.



1 to 16 decoder

List of commercially available DeMUX ICs:

IC NO	DEMUX Type	Decoder Type
74154	1-to-16	4-to-16
74138	1-to-8	3-to-8
74155	1-to-4	2-to 4

The 74154 : is A 1-to-16 deMux.

Pin 18 – i/p data D → active low ✓

Pins 20 to 23 – control bits A B C D.

Pin 1 to 11 and 13 to 17 – o/p lines.

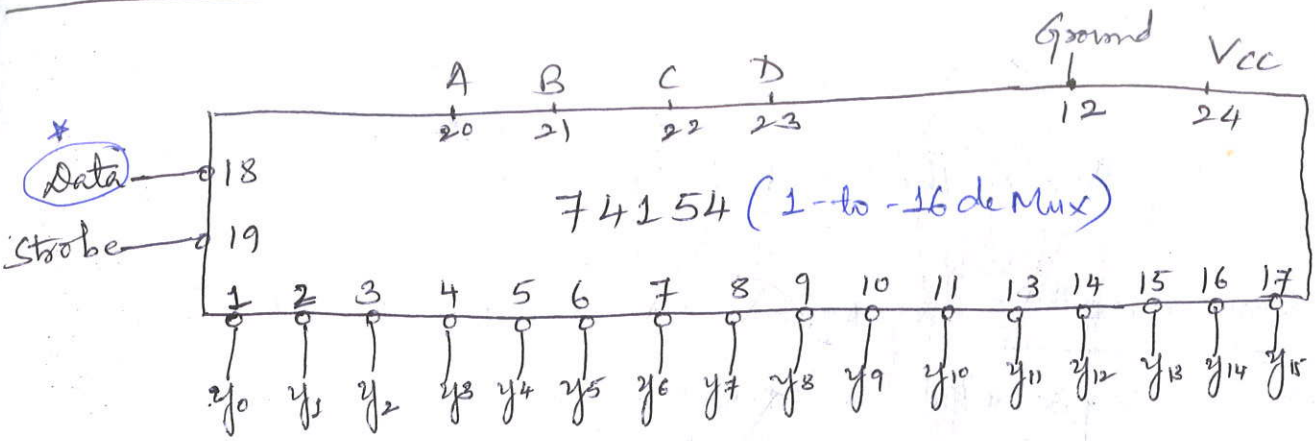
Pin 19 - STROBE – active-low i/p. ✓

Pin - 24 & pin 12 – Vcc & ground resp.

Truth Table of 1 to 16 DeMux : When the strobe is low, the control-i/p det. which o/p lines are low, similarly with high state.

Strobe	Data	A	B	C	D	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}
→ L	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
→ H	L	x	x	x	x	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
→ H	H	x	x	x	x	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L/D	L/D	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0	0	0	0	1		0														
	0	0	0	1	0			0													
	0	0	0	1	1				0												
	0	0	1	0	0					0											
	0	0	1	0	1						0										
	0	0	1	1	0							0									
	0	0	1	1	1								0								
	0	0	1	0	0									0							
	0	0	1	0	1										0						
	0	0	1	0	1											0					
	0	0	1	0	1												0				
	0	0	1	1	0													0			
	0	0	1	1	0														0		
	0	0	1	1	1															0	
	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Pin / logic diagm of 74154 ÷ deMux



Worked Example: Show how two 1-to-16 MUX can be connected to get 1-to-32 deMUX.

Soln: 1-to-32 deMUX has 5 select inputs ABCDE.

Four of them BCDE — fed to two 1-to-16 deMUX.

Fifth A — used to select one of these two deMux through strobe-i/p.

If A = 0, the top 74154 is choosen.

& BCDE directs data to one of the 15 o/p's of the IC.

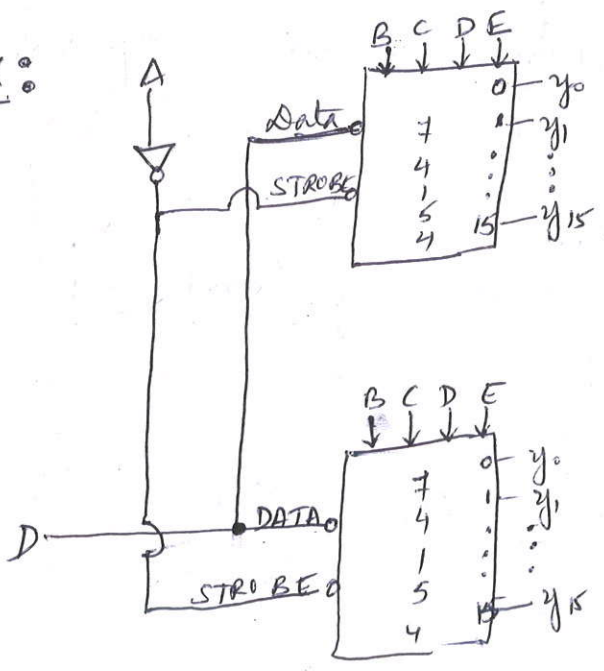
If A = 1, the top 74154 is choosen;

& depending on the value of BCDE, the data is directed to one of the 15 o/p's of the IC.

1 to 32 MUX:

from

1 to 16 MUX



H	H	H	H
H	H	H	H
H	H	H	H
	1	1	1
0			
	0		
1	1	0	

1 of 16 decoder : 74154 – with No data i/p

* A decoder is similar to deMUX, with one exception there is (NO) data input.
* Inputs are the control bits – ABCD

Logic Diagm : is same as DeMUX except D-line.

* The logic ckt is called 1 of 16 decoder because only 1 of the 16 o/p lines is high.

* **Binary to Decimal decoder :** ✓

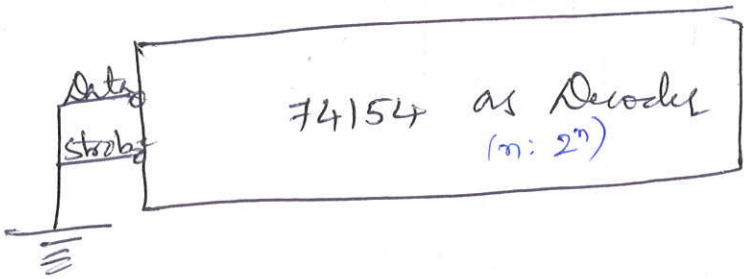
ABCD – possibilities from 0000 to 1111.

The subscript of the high o/p always equal to the decimal equivalent of ABCD. For this reason, the ckt is sometimes called a binary-to-decimal decoder.

4-line to 16-line decoder : ✓ Since it has 4-i/p lines & 16-o/p lines.

Decoder-demultiplexer : 74154 ✓ is called a decoder-deMUX, because it can be used either.

To use 74154 as decoder, just ground the i/ps DATA & STROBE. Then, the selected o/p line is in low state.



Using 74154 – DeMUX as Decoder

Worked Example:

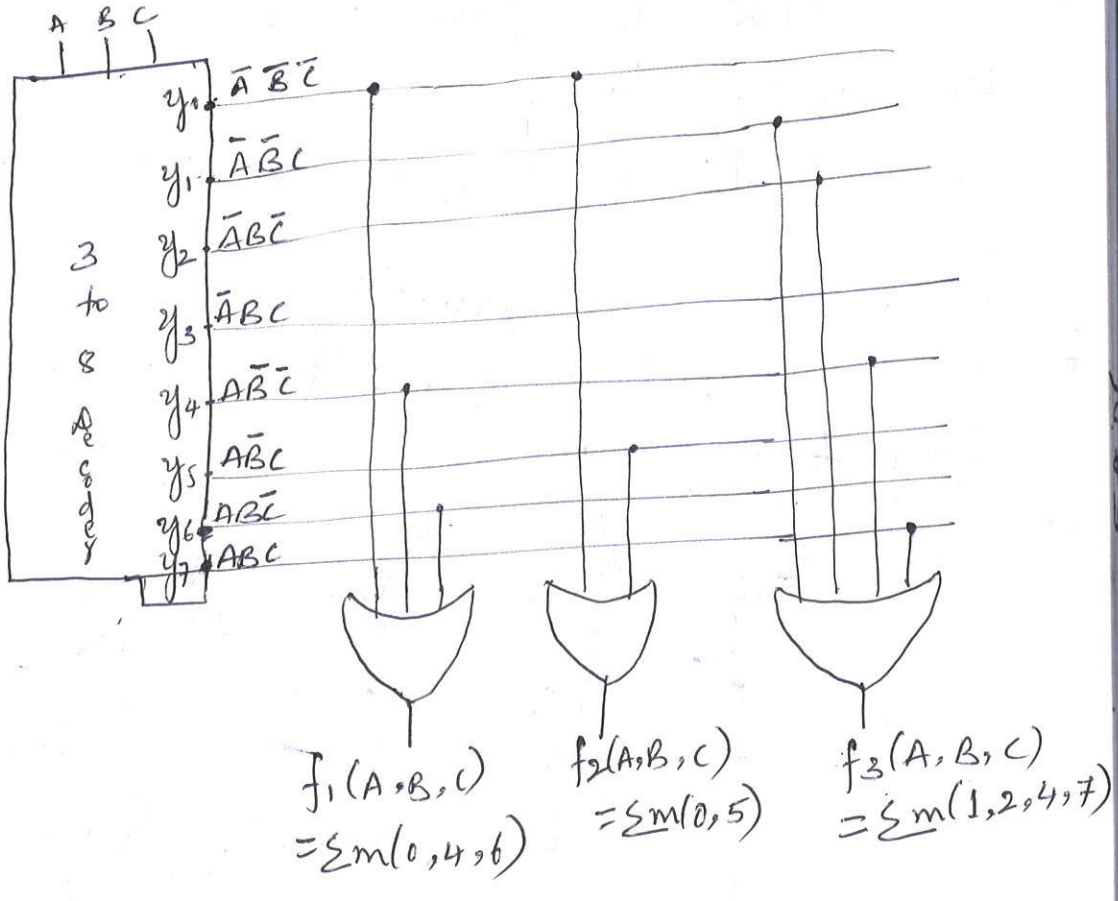
(1) Realize the f(y) Boolean f^n using 3-to-8 Decoder & multi-i/p OR gates.

$f_1(A, B, C) = \Sigma m(0, 4, 6)$

$f_2(A, B, C) = \Sigma m(0, 5)$

$f_3(A, B, C) = \Sigma m(1, 2, 3, 7)$

Soln: Since @ the decoder o/p, we'll get all the min-terms, we use them as shown to get req^d boolean f^n's.



BCD - to - Decimal Decoder:

BCD - binary coded decimal.

- The BCD code expresses each digit in a decimal no. by its nibble equivalent.

Eg: The decimal (429) is changed to BCD as (4) 0100 (2) 0010 (9) 1001

∴ BCD code 0100 0010 1001 is equivalent to $429_{(10)}$.

⇒ Convert the decimal no 8963 to its BCD form.

8	9	6	3
↓	↓	↓	↓
1000	1001	0110	0011

⇒ Converting to decimal form ~~BCD~~ BCD.

0101	0111	1000
↓	↓	↓
5	7	8

⇒ BCD digits are from 0000 to 1001 (0 through 9). High decimal being coded is 9. All combinations above this (1010 to 1111) can NOT exist in BCD code/form.

BCD-to decimal decoder (7445): 1 of 10 - decoder

Only 1 of the 10 o/p lines is high.

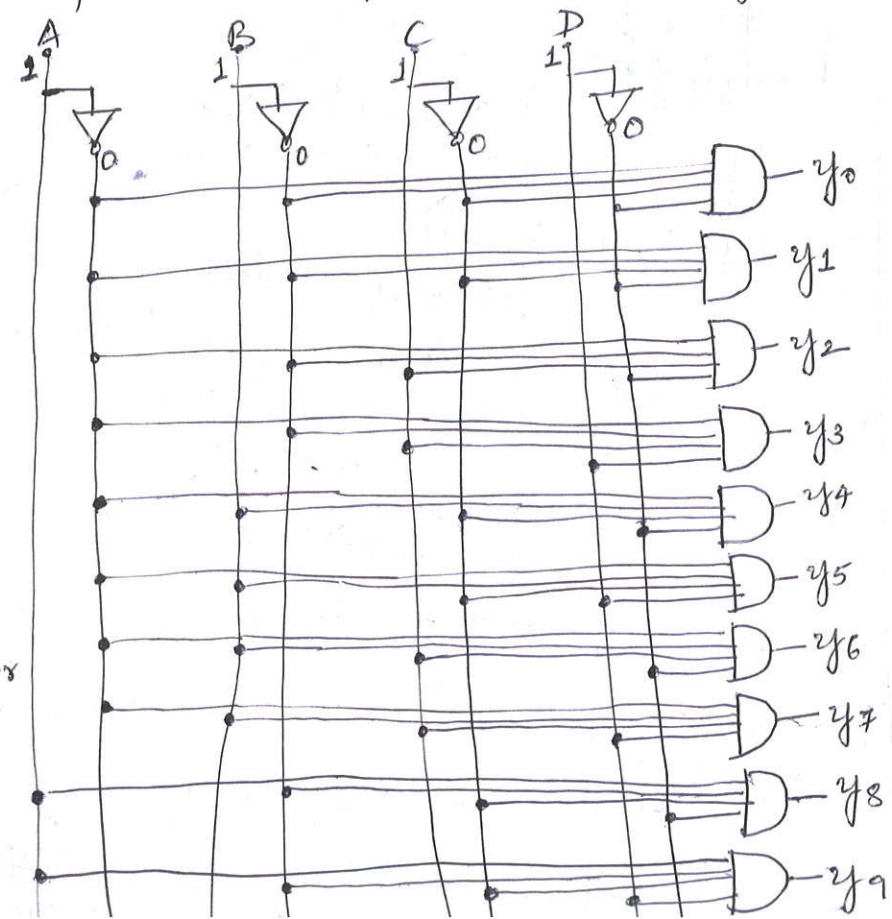


fig:
(1) of (10) decoder

The 74

Pin
Pin

Note:

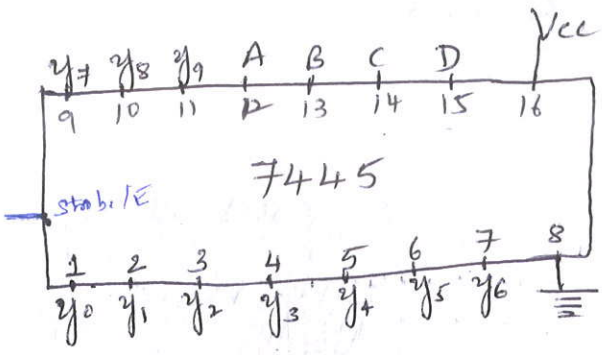
7445

Decimal NO	
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

These combinations forces o/p lines to HIGH state

The 7445 :

Pin - 16 — to Vcc
Pin - 8 — Ground
Pin 12 to 15 — BCD i/ps ABCD
Pin 1 to 7 and 9 to 11 — BCD o/ps. y_0 to y_9.



Note : An invalid BCD i/p (1010 to 1111) forces all o/p lines into the high state.

7445 Truth Table : BCD to decimal decoder

Decimal NO	Inputs A	B	C	D	Outputs y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1	1	1
2	0	0	1	0	1	1	0	1	1	1	1	1	1	1
3	0	0	1	1	1	1	1	0	1	1	1	1	1	1
4	0	1	0	0	1	1	1	1	0	1	1	1	1	1
5	0	1	0	1	1	1	1	1	1	0	1	1	1	1
6	0	1	1	0	1	1	1	1	1	1	0	1	1	1
7	0	1	1	1	1	1	1	1	1	1	1	0	1	1
8	1	0	0	0	1	1	1	1	1	1	1	1	0	1
9	1	0	0	1	1	1	1	1	1	1	1	1	1	0
These combinations forces o/p lines to HIGH state	1	0	1	0	1	1	1	1	1	1	1	1	1	1
	1	0	1	1	1	1	1	1	1	1	1	1	1	1
	1	1	0	0	1	1	1	1	1	1	1	1	1	1
	1	1	0	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	0	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1	1	1	1

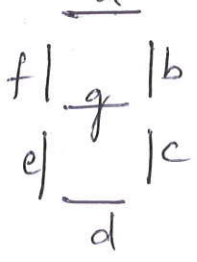
SEVEN-Segment Decoder:

The 7446 TTL - a 7-segment decoder driver

- In most practical applns, 7-segment displays are used to give visual indication of o/p states of digital ICs such as decade counters, latches etc.
- These o/ps are usually in 4-bit BCD form, & are thus NOT suitable for directly driving 7-segment displays. The special BCD-to-7-segment decoder/driver ICs are used to convert the BCD signal into form suitable for driving these displays.

Digit	Segments Activated	Display
0	a, b, c, d, e, f	
1	b, c	
⋮ 4 ⋮	b, c, f, g	
8	a, b, c, d, e, f, g	
9	a, b, c, d, f, g	

fig (b)

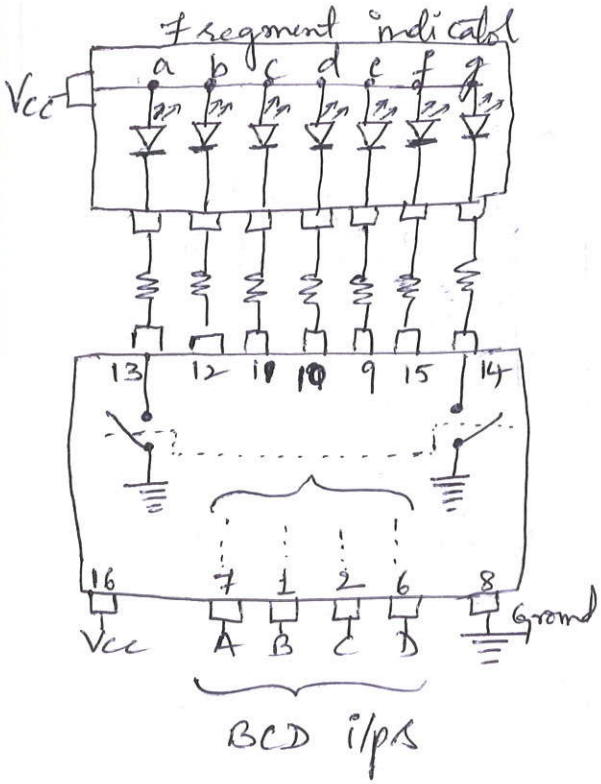


(a)

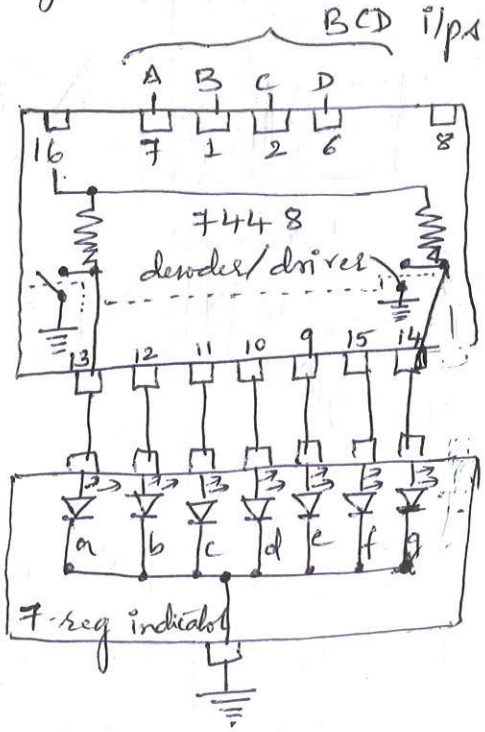
fig (a) & (b) Seven-segment indicator

The 7446 & the 7448

- Two types of decoder drivers, corresponding to the common-anode & common-cathode indicators.
- Each driver has 4 i/p pins (the BCD i/p) and 7-o/p pins (the a through g segments).



(a) 7446 - decoder driver



(b) 7448 - decoder driver

⇒ Logic ckts inside the 7446 convert the BCD input to the required output.

For instance, if the BCD i/p is 0111, the internal logic of 7446 will force LEDs a, b, c to conduct. As a result, dig-7 will appear on 7-segment indicator

⇒ Same with this as well.

⇒ 7446 requires external-current-limiting resistors

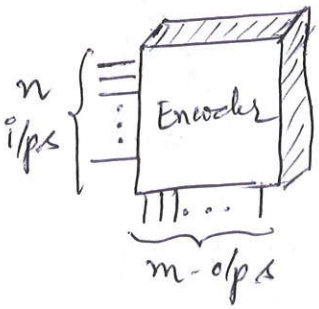
⇒ 7448 - has its own current-limiting resistors on chip

⇒ Switch-symbol is used to illustrate the oper^n of 7446 and 7448 in fig (a) & (b).

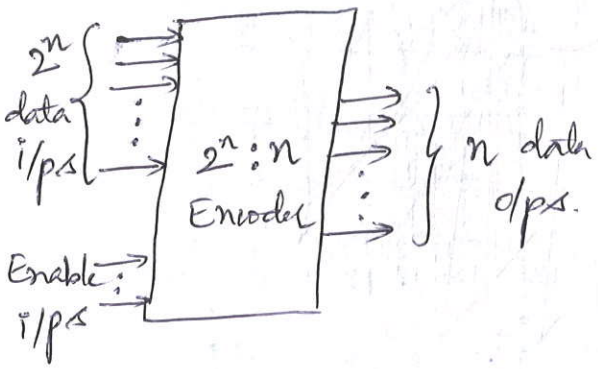
Encoders:

- An encoder is a digital circuit that performs the inverse operation of a decoder.
- An encoder converts an active i/p signal into a coded o/p signal.

General idea:



OR



⇒ An encoder has 2^n (or fewer) i/p lines & n-o/p lines. In encoder, the o/p lines generate the binary code corresponding to the i/p value.

Decimal to BCD Encoder: 74157 TTL.

→ 10-i/p lines
4-o/p lines

→ The switches are push-button switches like those of pocket calci.

→ When button-3 is pressed, C and D-OR gates have i/ps, therefore the o/p is

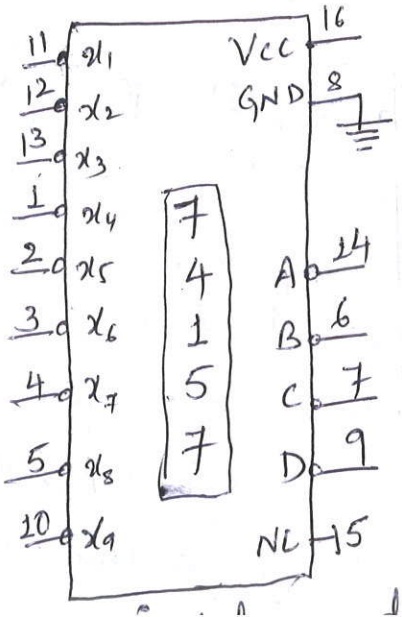
ABCD = 0011

→ For button 5, ABCD = 0101

→ For switch 9, ABCD = 1001

→ Decimal i/p x_1 to x_9 are connected to pins 1 to 5 and 10 to 13.

→ BCD o/ps A, B, C, D are



→ Pin 1
→ Pin

74157

Decimal value	
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

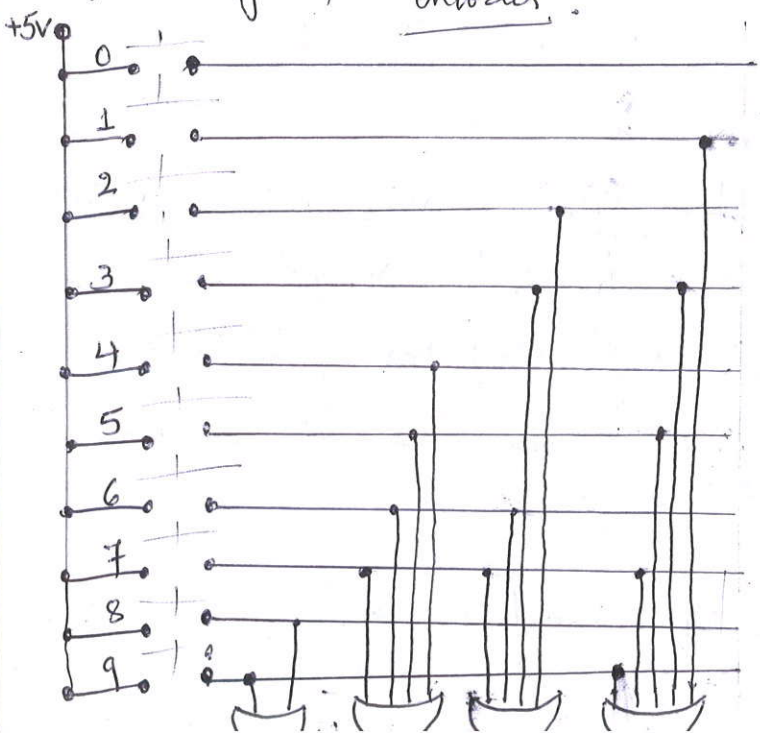
Logic di

→ Pin 16 - Vcc and pin-8 is grounded.
→ Pin 15 - NC - No connection - the pin is NOT used.

7415? Truth table : Active-low-inputs
active-low-outputs

Decimal value	Inputs									Outputs			
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	A (8)	B (4)	C (2)	D (1)
0	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0												0
2		0										0	
3			0									0	0
4				0							0		
5					0						0		0
6						0					0	0	
7							0				0	0	0
8								0		0			
9									0	0			0

Logic diagram of Decimal to BCD encoder:



Incidentally, the 74147 is called a "priority encoder", because it gives priority to the higher-order input. From the TT, if all the i/ps from x_1 through x_9 are low, the highest of these x_9, is encoded to get o/p 0110/LHHL. In other words, x_9 has priority over all

⇒ When x_9 is high, x_8 is in next line of priority & gets encoded if it is low.

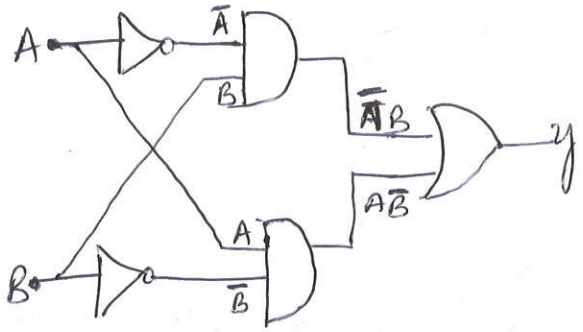
→ Working the way through TT, we can see that the highest active-low from x_9 to x_0 has priority & will control the encoding.

Exclusive-OR gates:

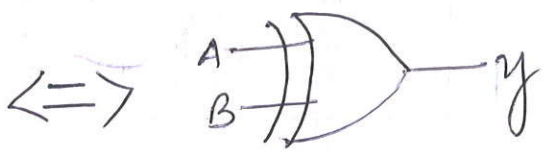
→ X-OR gate has high o/p only when odd no of i/ps is high.

→ Fig. shows how to build an X-OR gate. The upper AND gate forms the product $\bar{A}B$, while the lower one produces $A\bar{B}$. ∴ The o/p of OR gate is

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



Exclusive-OR gate



Logic symbol of OR-gate

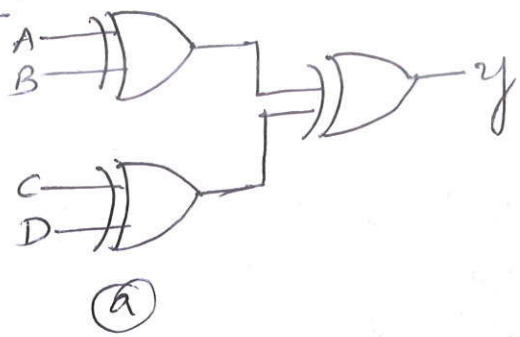
TT of 2 i/ps X-OR gate

A	B	y
0	0	0
0	1	1
1	0	1
1	1	0

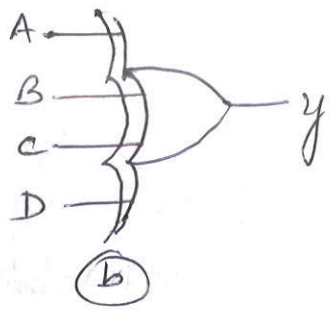
⇒ The o/p is high when A (or) B is high, but not when both are high (or) both are low. This is why the ckt is known as Ex-OR gate.

⇒ In other words, the i/ps must be different to get high (logic 1) output.

Four Inputs:



(a)



(b)

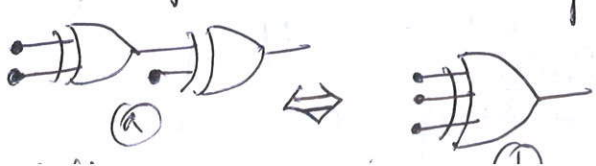
Four i/ps X-OR gate

Truth Table: When ABCD i/ps has an odd no of 1s, o/p=1.

Comment	A	B	C	D	y
Even	0	0	0	0	0
Odd	0	0	0	1	1
Odd	0	0	1	0	1
Even	0	0	1	1	0
Odd	0	1	0	0	1
Even	0	1	0	1	0
Even	0	1	1	0	0
Odd	0	1	1	1	1
Odd	1	0	0	0	1
Even	1	0	0	1	0
Even	1	0	1	0	0
Odd	1	0	1	1	1
Even	1	1	0	0	0
Odd	1	1	0	1	1
Odd	1	1	1	0	1
Even	1	1	1	1	0

First ABCD entry to product an o/p 1 is 0001; it has odd no of 1s. The next ABCD entry to product an o/p 1 is 0010; again an odd no of 1s. An o/p-1 also occurs for these ABCD i/ps 0100, 1000, 1011, 1101, and 1110, each having an odd no of 1s.

Any no of i/ps: Using 2-i/ps X-OR gates as building blocks, you can produce X-OR gates with any no



Parity Generators and checkers:

→ A parity-bit is used for the purpose of detecting errors during transmn of binary inform". It is an extra-bit included with a binary message to make the no. of 1s either odd or even.

→ The message, including the parity bit is transmitted & then checked at the receiving end for errors. An error is detected if the checked parity does NOT correspond with the one transmitted.

→ Parity generator: Circuit that generates the parity bit in the transmitter.

→ Parity checker: Circuit that checks the parity bit in the receiver.

→ Even parity: n-bit binary input has an even no of 1s.
Eg: 110011 ⇒ has even parity
∵ contains 4-1s.

→ Odd parity: n-bit input has an odd no of 1s.
Eg: 110001 ⇒ has odd parity.
∵ contain three-1s.

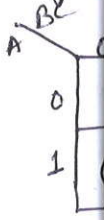
→ Two examples
1111 0000 1111 0011 ⇒ Even parity
1111 0000 1111 0111 ⇒ Odd parity

→ Table shows the 3-bit message with even parity and odd parity.

Problem:
→ Design an even-parity generator with 3-message bits

Sol":

⇒ Design



PG =
=
=
PG =

Sol:

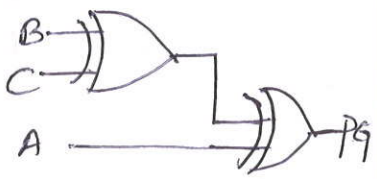
3-bit message			Odd parity bit	Even parity bit
A	B	C		
0	0	0	1	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	0	1

← parity generator truth table for even and odd parity

⇒ Design an even-parity generator with 3 message bits.

A \ BC	00	01	11	10
0	0	(1)	0	(1)
1	(1)	0	(1)	0

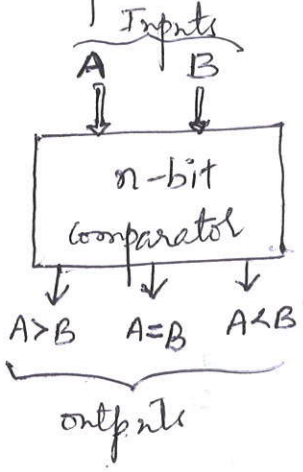
$$PG = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$
$$= \bar{A}(\bar{B}C + B\bar{C}) + A(\bar{B}\bar{C} + BC)$$
$$= \bar{A}(B \oplus C) + A\overline{(B \oplus C)}$$
$$\boxed{PG = A \oplus (B \oplus C)}$$



Magnitude Comparators:

Def^n: A comparator is a special combinational circuit designed primarily to compare the relative magnitude of 2-binary nos.

Block diag^m:



It receives two n-bit nos A and B as inputs and the outputs are $A>B$, $A=B$ and $A<B$. Depending upon the relative magnitude of the 2 nos, one of the o/ps will be 'high'.

⇒ (1) Design 1-bit comparator using basic gates.

Sol^n: Consider two 1-bit no A and B.

Inputs		Outputs		
A	B	$Y_{A>B}$	$Y_{A=B}$	$Y_{A<B}$
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

$Y_{A>B}$:- k-map

A \ B	0	1
0	0	0
1	(1)	0

$Y_{A>B} = A\bar{B}$ ✓

$Y_{A=B}$ - k-map:

A \ B	0	1
0	(1)	0
1	0	(1)

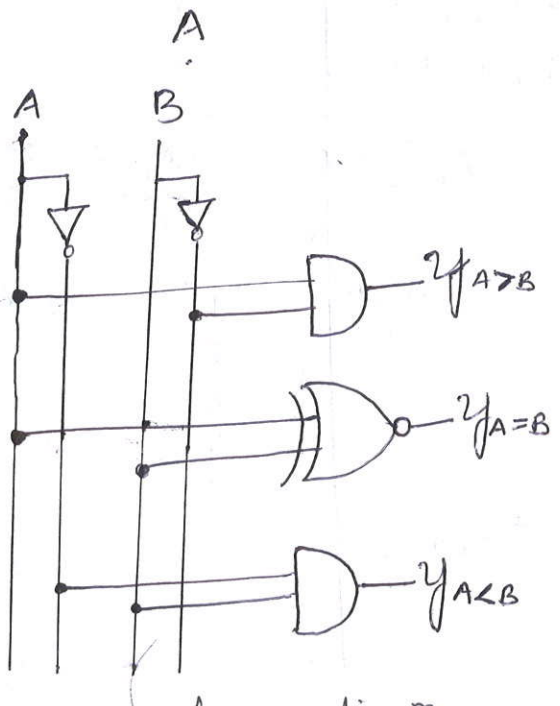
$Y_{(A=B)} = \bar{A}\bar{B} + AB = \overline{A \oplus B}$

$Y_{(A=B)} = A \odot B$ ✓

$Y_{A<B}$ - k-map:

A \ B	0	1
0	0	(1)
1	0	0

$Y_{(A<B)} = \bar{A}B$ ✓



Logic diag^m.

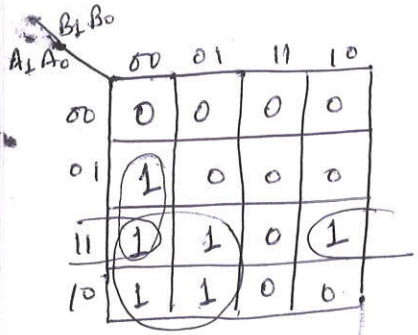
(2) Design 2-bit comparator using gates.

Soln:

Inputs				Outputs		
A_1	A_0	B_1	B_0	A>B	A=B	A<B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	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	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

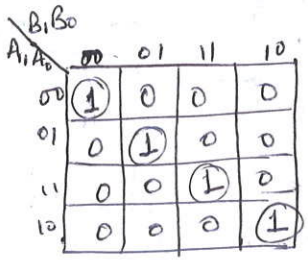
K-map simplification:

(i) A>B



$\therefore A>B = A_1\bar{B}_1 + A_0\bar{B}_1\bar{B}_0 + A_1A_0\bar{B}_0$

(ii) A=B



$\therefore (A=B) = A_1A_0B_1B_0$

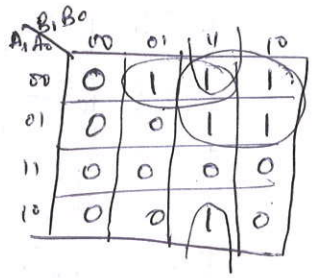
$\Rightarrow \bar{A}_1\bar{A}_0\bar{B}_1\bar{B}_0 + \bar{A}_1A_0\bar{B}_1B_0 + A_1A_0B_1B_0 + A_1\bar{A}_0B_1\bar{B}_0$

$= \bar{A}_1\cdot\bar{B}_1(\bar{A}_0\bar{B}_0 + A_0B_0) + A_1B_1(A_0B_0 + \bar{A}_0\cdot\bar{B}_0)$

$= \bar{A}_1\bar{B}_1(\overline{A_0 \oplus B_0}) + A_1B_1(\overline{A_0 \oplus B_0})$

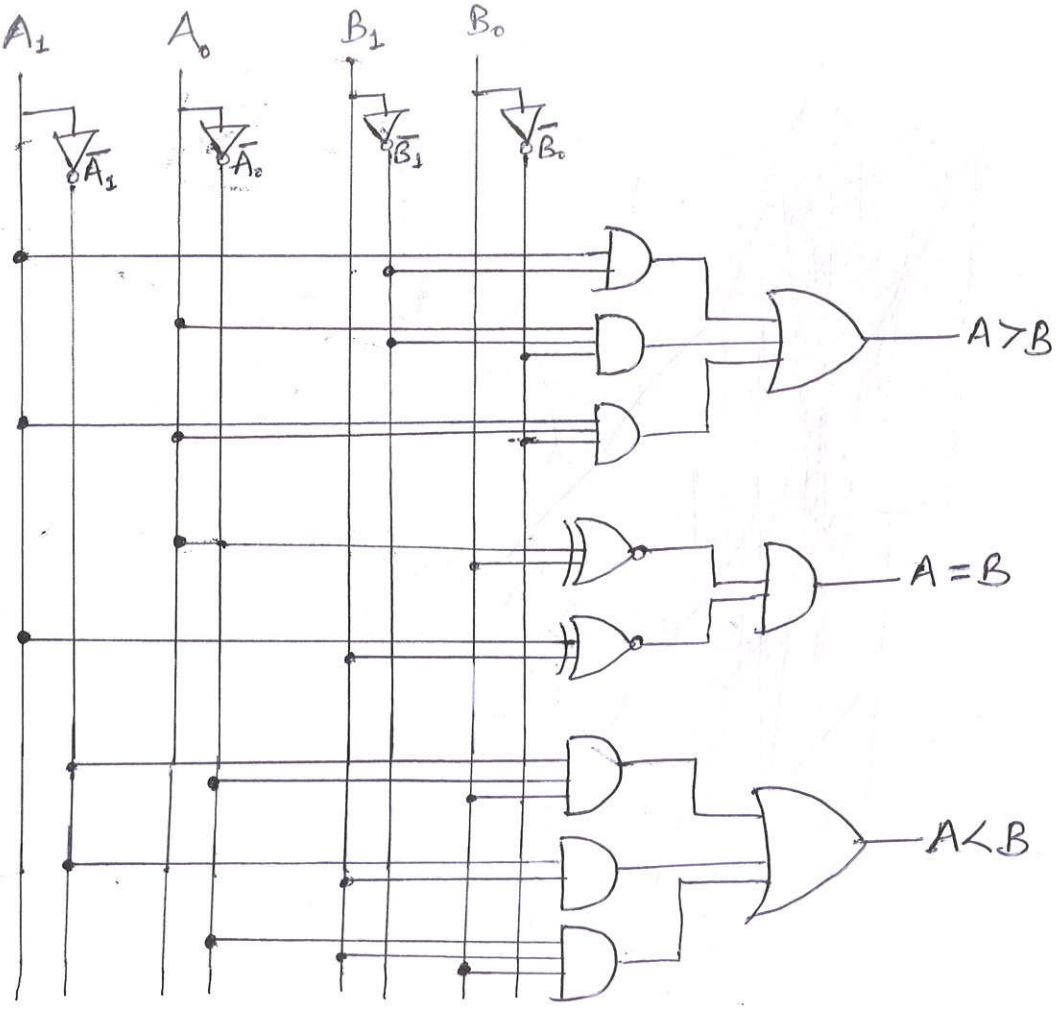
$= (\overline{A_0 \oplus B_0})(\bar{A}_1\bar{B}_1 + A_1B_1) = \overline{(A_0 \oplus B_0)}\cdot\overline{(A_1 \oplus B_1)}$

(iii) A<B



$\therefore (A<B) = \bar{A}_1\bar{A}_0B_0 + \bar{A}_1B_1 + \bar{A}_0B_1B_0$

Logic Diagram of 2-bit comparator



1	
1	
0	
0	

$\bar{A}_1 B_1 +$

$\bar{A}_0 B_1 B_0$

$B_1 B_0 +$

$B_1 \bar{B}_0$

$_1 (A_0 B_0 +$

$\bar{A}_0 \cdot \bar{B}_0)$

$\bar{B}_0)$

$\overline{\oplus B_0}) \cdot \overline{(A_1 \oplus B_1)}$