23 FRIDAY — Day 144-222 Week 21 — MAY 2008 — IMPORTANT

Digital Electronics

Study of systems known as digital systems.

A system that works with the digits (0-9) is called digital system.

It was found to be advantageous to use only 2 digits (0 & 1) rather than 10 digits in the system. This was called binary digital system.

Real world prob (Easy understood by us) → Binary digital system (Easy in 0 & 1) → This is the pre-requisite to convert a prob. from our understandable language to system understandable lang. i.e 0&1

→ Truth table is the most basic form understable by B.D.S.

→ Digital system is basically a I/p which is constructed with I/p of 3 logic gates.

x	y	z	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

x, y, z (input) → B.D.S → f (o/p) — implemented by use of logic gates.

APRIL 2008

Su	Mo	Tu	We	Th	Fr	Sa
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

08 MAY — Week 21 Day 145-221 — IMPORTANT — SATURDAY 24

→ Now, this B.D.S can be constructed by implementing logic gates for each of the I/p combin° but that design won't be cost-effective & easy. ∴ what we do is:

① Either implement those I/p combin° which gives the o/p 1, the rest I/p combin° will automatically produce 0 in o/p & vice-versa.

→ Now, we implement the system using such I/p combin° that are B.D.S. produces o/p of 1 then it is called min terms, sum of pdt., sum of min terms.

→ When we implement the system using I/p combin° that produces o/p 0, then it is called max terms, pdt of sum, pdt of max terms.

Repres° of minterms:

$$f = (\bar{x}\bar{y}\bar{z})_{m_0} + (\bar{x}y\bar{z})_{m_2} + (\bar{x}yz)_{m_3} + (x\bar{y}z)_{m_5}$$

→ minterms

Day 146-220 — SUNDAY 25

→ obtained by ANDing all the variables of the f° in such a way

a) bit 0 is represented by complemented variable.

b) bit 1 is represented by uncomplemented variable.

→ Also represented as m_j, where j is the decimal equivalent of the binary I/p.

MAY 2008

Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

→ f → is said to be in the sum of minterms form.

Also represented as → $f = m_0 + m_2 + m_3 + m_5$

or $f = \Sigma(0,2,3,5)$

Three representn for f:

① Variable : No. of input variable (3).

② Literals : No. of time variable is repeated (12).

③ Terms : 4.

Simplifn techniques for a f:

① using laws of boolean algebra.

② using K-map.

Simplifn is a technique in which we try to reduce the no of terms & literals used in the representn of f.

→ Simplified f is s.t.b in the sum of products form

eg. $f = \underbrace{(x+y+\bar{z})}_{M_1} \cdot \underbrace{(\bar{x}+y+z)}_{M_4} \cdot \underbrace{(\bar{x}+\bar{y}+z)}_{M_6} \cdot \underbrace{(\bar{x}+\bar{y}+\bar{z})}_{M_7}$

↓ maxterms

→ obtained by ORing all the variable

→ s.t.b in the products of maxterms form.

→ represented as → $f = \Pi(1,4,6,7)$.

→ Simplified f is s.t.b in products of sum form.

eg. $f(x,y,z) = (\bar{x}+\bar{y})(\bar{x}+\bar{z})$

variables = 3, terms = 2, literals = 4.

→ $m_0 = \bar{x}\cdot\bar{y}\cdot\bar{z}$

$\overline{m_0} = \overline{\bar{x}\cdot\bar{y}\cdot\bar{z}}$

$= x+y+z$

$= M_0$

⇒ In general, $\boxed{\overline{m_j} = M_j}$

→ minterms & maxterms r complements of each other.

Canonical Forms

→ A f which is in the unsimplified form is s.t.b in canonical form (i.e. should be either in sum of minterms or the pdt. of maxterms).

Conversion to canonical form

① Interchange the symbols Σ & Π.

② Write the missing numbers.

28 Day 149-217 Week 22 MAY 2008

WEDNESDAY

IMPORTANT

eg.

x y	f
0 0	1
0 1	0
1 0	1
1 1	0

$f = \Sigma(0,2)$
$= \pi(1,3)$

Q Express the given f in canonical sum of minterms form.

$f(x,y,z) = x + y\bar{z}$
$= x \cdot 1.1 + 1 \cdot y \cdot \bar{z}$
$= x(y+\bar{y})(z+\bar{z}) + (x+\bar{x})y\cdot\bar{z}$

① $A \cdot 1 = A$
$\Rightarrow A = A \cdot 1$

② $A + \bar{A} = 1$
$\Rightarrow 1 = A + \bar{A}$

$= x\left(yz + y\bar{z} + \bar{y}z + \bar{y}\bar{z}\right) + xy\bar{z} + \bar{x}y\bar{z}$
$= xyz + xy\bar{z} + x\bar{y}z + x\bar{y}\bar{z} + \bar{x}y\bar{z} + \bar{x}\bar{y}\bar{z}$
$= m_7 + m_6 + m_5 + m_4 + m_6 + m_2$

canonical sum → $= m_2 + m_4 + m_5 + m_6 + m_7$

min → $= \Sigma(2,4,5,6,7)$

Q Express the given f in canonical pdt of maxterm form.

$f(x,y,z) = (\bar{x}+\bar{z}) \cdot (\bar{y}+z)$
$= (\bar{x}+0+\bar{z}) \cdot (0+\bar{y}+z)$
$= (\bar{x}+(y\cdot\bar{y})+\bar{z}) \cdot (x\cdot\bar{x}+\bar{y}+z)$
$= (\bar{x}+y+\bar{z})(\bar{x}+\bar{y}+\bar{z})(x+\bar{y}+z)(\bar{x}+\bar{y}+z)$

① $A + 0 = A$
$\Rightarrow A = A + 0$

② $A \cdot \bar{A} = 0$
$\Rightarrow 0 = A \cdot \bar{A}$

APRIL 2008

Su	Mo	Tu	We	Th	Fr	Sa
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

2008 MAY Week 22 Day 150-216

IMPORTANT

THURSDAY 29

Distributive law:

ordinary algebra: ✓ $x \cdot (y+z) = xy + x\cdot\bar{z}$

Boolean algebra: ✓ valid

$\times$ ⊗ $x + y\bar{z} = (x+y)(x+z)$ ✓

$= M_5 M_7 M_2 M_6$
$= \pi(2,5,6,7)$

Q Minterm expansion of $f(P,Q,R) = PQ + Q\bar{R} + P\bar{R}$ is

a) $m_2 + m_4 + m_6 + m_7$ ✓
b) $m_0 + m_1 + m_3 + m_5$
c) $m_0 + m_1 + m_6 + m_7$
d) $m_3 + m_5 + m_4 + m_6$

$= PQ(R+\bar{R}) + (P+\bar{P})Q\bar{R} + P(Q+\bar{Q})\cdot\bar{R}$
$= PQR + PQ\bar{R} + PQ\bar{R} + \bar{P}Q\bar{R} + PQ\bar{R} + P\bar{Q}\bar{R}$
$= m_7 + m_6 + m_6 + m_2 + m_6 + m_4$
$= m_2 + m_4 + m_6 + m_7$

Total Possible f"

x	y	f_0	f_1	f_2	f_3	- - -	f_{15}
0	0	0	0	0	0		1
0	1	0	0	0	0		1
1	0	0	0	1	1		1
1	1	0	1	0	1		1

MAY 2008

Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

i.e. Total no. of possible $f' = 2^{(2^n)}$

2^n → possible i/p combination
n → no. of i/p variables.

⟹ With n variables, total no. of 2^{2^n} different boolean f' can be formulated.

No. of variables	Total possible f'
2	16
3	256
4	65,536 = 64×1024 = 64K

16 boolean f' with 2 i/p variables

→ All the boolean f' made by using 2 i/p variables are standard boolean f'. Each of them have a unique name.

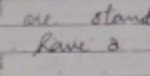
$f_0 = 0$ null
$f_1 = x \cdot y$ AND
$f_2 = x \cdot \bar{y}$ Inhibition
$f_3 = x$ Transfer
$f_4 = \bar{x} \cdot y$ Inhibition
$f_5 = y$ Transfer
$f_6 = x\bar{y} + \bar{x}y$ EX-OR
$f_7 = x + y$ OR
$f_8 = \overline{x+y}$ NOR
$f_9 = \bar{x}\bar{y} + xy$ X-NOR
$f_{10} = \bar{y}$ Complement
$f_{11} = x + \bar{y}$ Implication
$f_{12} = \bar{x}$ Complement
$f_{13} = \bar{x} + y$ Implication
$f_{14} = \overline{x \cdot y}$ NAND
$f_{15} = 1$ Identity

Logic Gates

Basic gates:
1. NOT
2. AND
3. OR

Compact notation:
4. NAND } 2 universal gates (cont any combination).
5. NOR
6. EX-OR } Exclusive gates
7. EX-NOR

AND gate



x	y	f
0	0	0
0	1	0
1	0	0
1	1	1

$f = x.y$ (minterm approach)
$= (x+y)(x+\bar{y})(\bar{x}+y)$

→ obeys the commutative law
i.e. $x.y = y.x$

→ obeys the associative law
i.e. $(xy)z = x(yz)$

IMPORTANT

∴ Total no. of possible f = $2^{(2^n)}$

2^n → possible i/p combinations
n → no. of i/p variables.

→ With n variables, total no. of 2^{2^n} different boolean f can be formulated.

No. of variables	Total possible f
2	16
3	256
4	65,536 = 64×1024 = 64K

16 boolean f with 2 i/p variables

→ All the boolean f made by using 2 i/p variables are standard boolean f. Each of them have a unique name.

$f_0 = 0$	null	
$f_1 = x \cdot y$	AND	
$f_2 = x \cdot \bar{y}$	Inhibition	
$f_3 = x$	Transfer	
$f_4 = \bar{x} \cdot y$	Inhibition	
$f_5 = y$	Transfer	
$f_6 = \bar{x}y + x\bar{y}$	EX-OR	
$f_7 = x + y$	OR	
$f_8 = \overline{x+y}$	NOR	
$f_9 = \bar{x}\bar{y} + xy$	X-NOR	
$f_{10} = \bar{y}$	Complement	
$f_{11} = x + \bar{y}$	Implication	
$f_{12} = \bar{x}$	Complement	

IMPORTANT

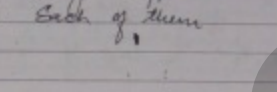
$f_{13} = \bar{x} + y$ Implication
$f_{14} = \overline{x \cdot y}$ NAND
$f_{15} = 1$ Identity

Logic Gates

Basic gates:
1. NOT
2. AND
3. OR

Compact other gates:
4. NAND — universal gates (don't obey associative law).
5. NOR
6. EX-OR — Exclusive gates
7. EX-NOR

AND gate



x	y	f
0	0	0
0	1	0
1	0	0
1	1	1

$f = x \cdot y$ (minterm approach)
$= (x+y)(\bar{x}+\bar{y})(\bar{x}+y)$ → taking only 0's of f.

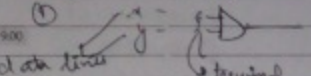
→ obeys the commutative law
⇒ $x \cdot y = y \cdot x$

→ obeys the associative law
i.e. $(\dot{y}z) = (x \cdot y) \cdot z$

Day 153·213 SUNDAY 01

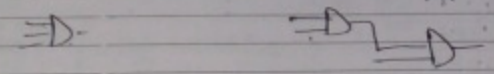
02 Day 154 212 Week 23 JUNE 2008
MONDAY IMPORTANT

Physical significance of these bars:

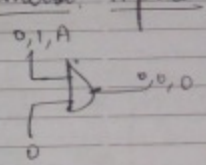
① 

The connection with these terminals can be made in any order if the logic gate follows the commutative law.

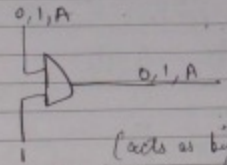
② If 3 i/p AND gate is not available, then we will have to use 2 AND gate.



Control Inputs



Disable i/p
(doesn't allow the other i/p to show its effect)



(acts as buffer).

Enable i/p
(allows the other i/p to show its effect).

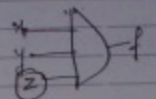
Control i/p

* Buffer is a circuit which gives the same o/p as that of i/p.

→ disable i/p is not always 0 or enable 1.

→ Enable i/p may make the circuit to act as an inverter.

2008 JUNE Week 23 Day 155 211 03
IMPORTANT Unused i/p TUESDAY



If we want to leave one of the i/p as unused i/p then we should apply the enable i/p of that unused i/p.

→ If the logic gate (AND) is implemented using TTL (bipolar transistor logic) then this is called i/p can be left & unconnected (i.e. open) then this is the property of TTL logic family that an open i/p act as logic 1.

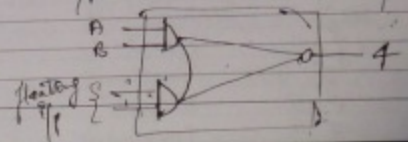
→ For AND gate, logic 1 acts as an enabled i/p.

→ In order to use only 2 i/p of a 3 i/p AND gate, the unused i/p can be connected in one of the following ways:
→ logic 1 // pull up method (Best method).
→ any of the used i/p.
→ In case of TTL logic, it can be left open/floating.

Note: In case of TTL logic, open/floating i/p acts as logic 1.

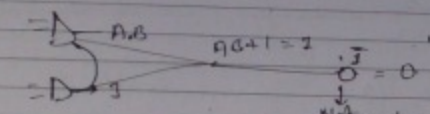
→ If the potential/voltage of a open i/p is raised using high voltage supply, then it is called pull up method.

The fig shows the internal schematic of a TTL AND-OR Invert (AOI) gate for the i/p shown, the output f is.

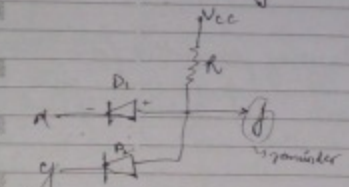


a) 0
b) 1
c) AB
d) $\overline{A\,B}$

Q5 (ADI) → And → OR → NOT.



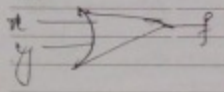
AND gate using Diodes



I/p x	I/p y	Status of Diode D_1	Status of Diode D_2	O/p
0	0	On	On	0
0	1	On	Off	0
1	0	Off	On	0
1	1	Off	Off	1 (Vcc)

Diode will be off when:
(a) no voltage is applied
(b) = " " " to both ends (as potential diff is needed).
(c) high on -ve side, low on +ve side

OR gate



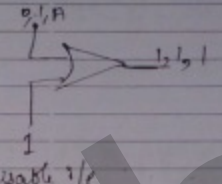
x	y	f
0	0	0
0	1	1
1	0	1
1	1	1

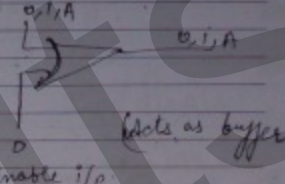
$f = x + y$ (maxterm)

$= \bar{x}y + x\bar{y} + xy$

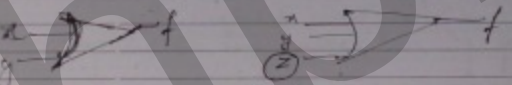
→ obeys commutative law
$\Rightarrow x+y = y+x$

→ obeys associative law
$x+(y+z) = (x+y)+z$





Unused inputs



$x+y, x+\bar{y}, \bar{x}+y$ $\quad x+x+y = x+y$

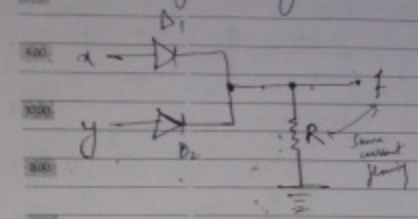
∴ unused i/p can be connected to any of the used i/p.

→ In order to use only 2 i/p of 3 i/p OR gate, the unused i/p can be connected in one the following ways:

① Logic 0 / Pull down method (Best method)

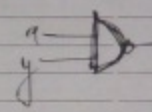
② One of the used i/p.

OR gate using diode



x	y	D_1	D_2	f
0	0	off	off	0
0	1	off	on	1
1	0	on	off	1
1	1	on	on	1

NAND gate.



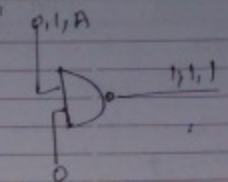
x	y	f
0	0	1
0	1	1
1	0	1
1	1	0

$f = \overline{x \cdot y}$
$= \bar{x} + \bar{y}$ } (using inverse demorgan law) (neither max nor minterm)

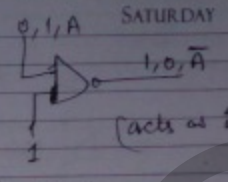
→ obeys commutative law
⇒ $\overline{xy} = \overline{yx}$

→ does not obey associative law.
⇒ $\overline{x \cdot (y \cdot z)} \neq \overline{(\overline{xy}) \cdot z}$

NOTE: Universal gates do not obey associative law.



Disable i/p

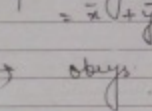


Enable i/p (acts as inverter).

unused i/p of NAND gate is connected in the same way as the unused i/p of AND gate.

→ As NAND gate don't obey associative law, we have to use the 3-i/p nand gate only (no use of 2-i/p nand gate for making 3 i/p nand gate).

NOR gate



x	y	f
0	0	1
0	1	0
1	0	0
1	1	0

$f = \bar{x} \cdot \bar{y}$ (minterm approach)
$= \overline{x+y}$ (Inverse demorgan)

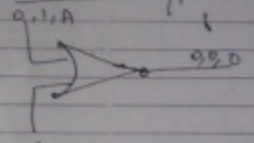
obeys → (i) commutative law
$\overline{x+y} = \overline{y+x}$

(ii) does not obey associative law
$\overline{x+(y+z)} \neq \overline{(x+y)+z}$

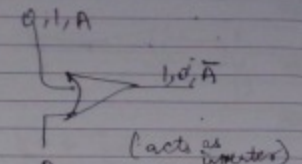
09 Day 161-205 Week 24 JUNE 2008

MONDAY IMPORTANT

Control i/p



Disable i/p



enable i/p

Unused i/p → of NOR gate is connected in the same way as the unused i/p of OR gate.

Which of the foll. is not equivalent to $\bar{x}$?

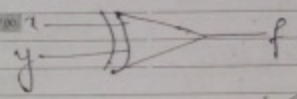
a) x NAND x $\quad \overline{x \cdot x} = \bar{x}$

b) x NOR x $\quad \overline{x+x} = \bar{x}$

c) x NAND 1 $\quad \overline{x \cdot 1} = \bar{x}$

d) x NOR 1 $\quad \overline{x+1} = \bar{1} = 0$

EX-OR



x	y	f
0	0	0
0	1	1
1	0	1
1	1	0

$f = x\bar{y} + \bar{x}y$

= x OR y but not both

→ One of the cond. of OR gate is excluded & then remaining logic gate is called Ex-OR

2008 JUNE Week 24 Day 162-204 10

IMPORTANT TUESDAY

$f = \bar{x}y + x\bar{y}$ (minterms)

$= (x+y)(\bar{x}+\bar{y})$ (maxterms)

→ $f = 1$, when $x \neq y$

$= 0$, when $x = y$

→ EX-OR f° is an odd f°

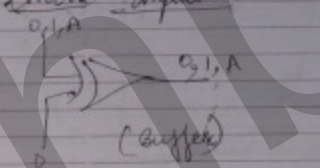
→ obeys commutative law

$x \oplus y = y \oplus x$

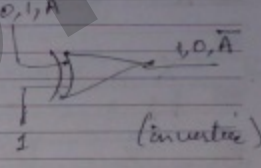
→ obeys associative law

$x \oplus (y \oplus z) = (x \oplus y) \oplus z$

Control Inputs



(Buffer)



(inverter)

Both are enable inputs

→ $A \oplus A = 0$

$A \oplus \bar{A} = 1$

$A \oplus 0 = A$

$A \oplus 1 = \bar{A}$

If $A \oplus B = C$

then $A \oplus C = B$

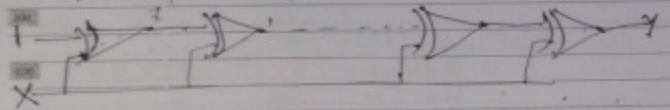
11 Dec 163/203 Week 24 — JUNE 2008 — WEDNESDAY — IMPORTANT

$A \oplus A \oplus A = A$

$A \oplus A \oplus A \oplus A = 0$

$\Rightarrow A \oplus A \oplus A \ldots\ldots$ — n times
$= A$ when n is odd
$= 0$ when n is even

Q) If the i/p to the digital ckt consisting of a cascade of 20 XOR gates is X, then o/p Y is equal to.



a) 0 b) 1 c) X d) $\bar{X}$

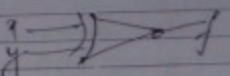
Q) X is interchanged by 1
1 " " X, find Y.

Ans → X (as it acts as inverter).
as even no. of inverter = X
odd = $\bar{X}$

→ NAND ≅ XOR (as evalled i/p is 1 it acts as inverter).
∴ Even NAND gates = X
odd = $\bar{X}$

→ AND acts as buffer for logic 1
∴ any no. of gates has o/p = X

EX-NOR



x	y	f
0	0	1
0	1	0
1	0	0
1	1	1

$f = \bar{x}\bar{y} + xy$ (minterm)
$= (x+\bar{y})(\bar{x}+y)$ (maxterm)

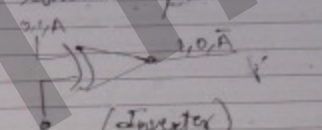
→ $f = 1$, when $x = y$
$= 0$, when $x \neq y$

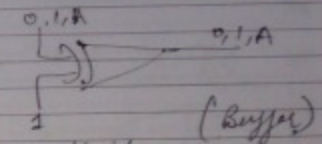
→ because of this reason, also known as
→ Equivalence logic
→ Coincidence logic

→ used in equality detector circuits.

→ obeys Commutative & associative

Control dependent





→ $A \odot A = 1$
$A \odot \bar{A} = 0$
$A \odot 0 = \bar{A}$
$A \odot 1 = A$

13 Day 165 201 Week 24 JUNE 2008

FRIDAY

IMPORTANT

→ $A \oplus B = \overline{A \oplus B}$

$= \bar{A} \oplus B$

$= A \oplus \bar{B}$

→ $A \odot B = \overline{A \odot B}$

$= \bar{A} \odot B$

$= A \odot \bar{B}$

→ $A \oplus A \oplus A = A$

→ $A \oplus A \oplus A \oplus A = 1$

∴ $A \oplus A \oplus A \ldots$ — n times

$= A$ when n is odd

1 when n is even.

But $A \oplus B \oplus C \neq \overline{A \oplus B \oplus C}$

b/c XNOR is not even f^n.

Back $A \oplus B \oplus C = \overline{A \oplus B} \oplus C$

Let $X = A \oplus B$ & $Y = C$

$= \bar{X} \odot Y$

$= X \oplus Y$

$= A \oplus B \oplus C$

→ When no of I/P variable is even, XNOR acts as an even f^n.

→ When no. of input variable is odd, XNOR acts as an odd func.

$A \odot B \odot C \odot D = A \oplus B \oplus C \oplus D$

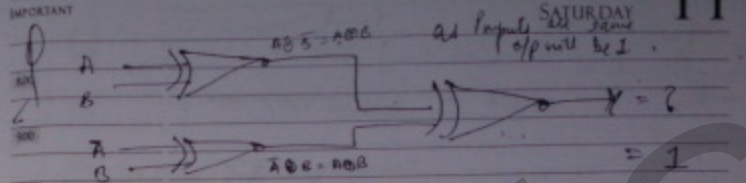
MAY 2008

Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

2008 JUNE Week 24 Day 166 200 14

IMPORTANT

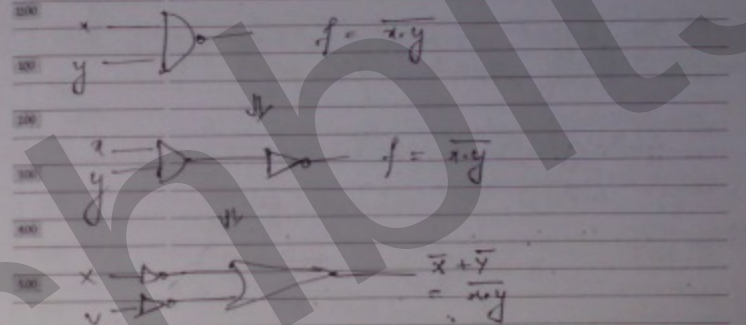
SATURDAY



As Inputs are same o/p will be 1.

= 1

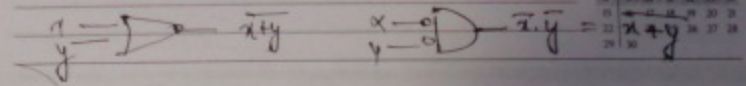
Bubbled logic



Day 167 199 SUNDAY 15

→ NAND ≡ Bubbled OR

→ NOR ≡ Bubbled AND



JUNE 2008

Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

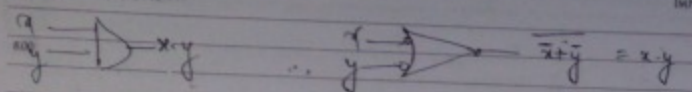
$\bar{x} + y$

16 DAY 168/198 WEEK 25

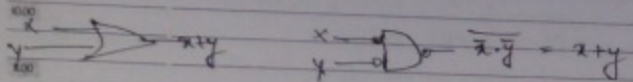
JUNE 2008

MONDAY

IMPORTANT



AND = Bubbled NOR

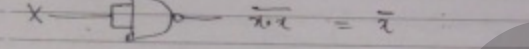


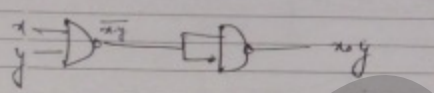
OR = Bubbled NAND

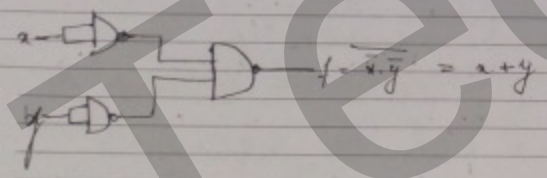
The bubbled equivalent of a logic gate is the diagonally opposite logic gate.

Universal Gates

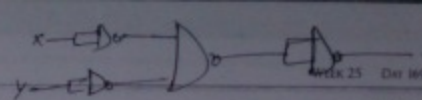
a) NAND as universal gate

i) NOT 

ii) AND 

iii) OR 

NOR:



2008 JUNE

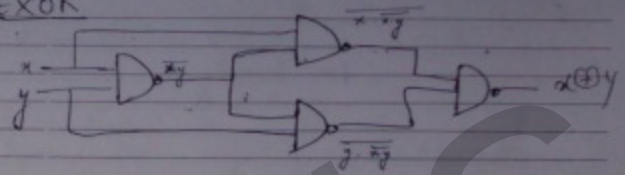
IMPORTANT

WEEK 25 DAY 169/197

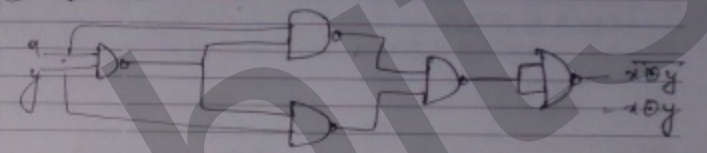
TUESDAY

17

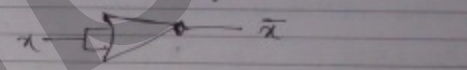
a) EXOR

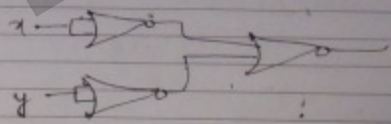


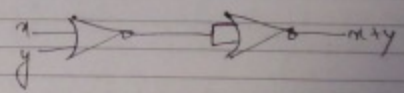
b) ENOR



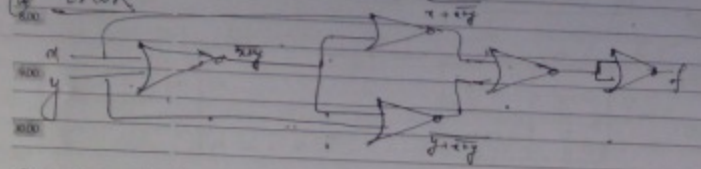
b) NOR as universal gate

i) NOT 

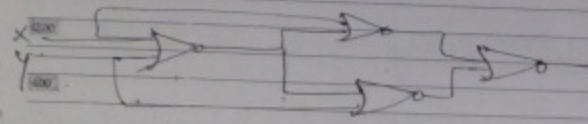
ii) AND 

iii) OR 

7. EXOR

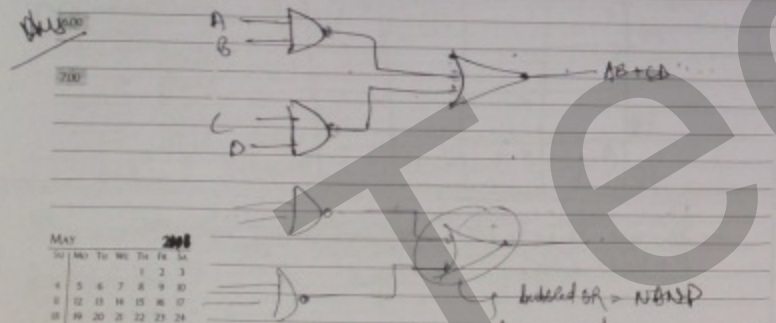


8. EX-NOR



9. Boolean $Y = A.B + C.D$ is to be realized using only 2 i/p NAND gate. Min. no. of gates req:

a) 2 b) 3 c) 4 d) 5

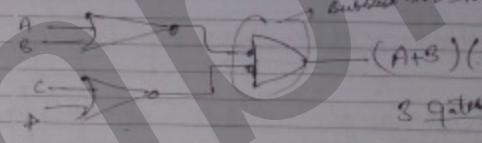


∴ 3 gates reqd

It is a 2 level AND-OR implementation which is equivalent to 2 level NAND-NAND implementation.

→ When circuit is implemented in SOP form, then only no. of NAND gates can be found very easily.

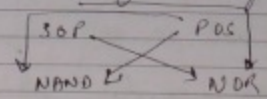
9. $Y = (A+B).(C+D)$ to be implemented using only 2 i/p NOR gate.



3 gates reqd.

→ 2 level OR-AND implementation ≡ NOR-NOR

→ Product of Sum form.



→ If no. of NAND gate are asked for POS, then we need to convert POS to SOP.

20 Date 172-194 Week 25

FRIDAY JUNE 2008

IMPORTANT

→ If we use duality or complement method to convert POS to SOP (or vice versa) then it will change the value of its f^n also.

⟹ SOP is converted into POS (& vice versa) by using the distributive prop.

eg $f = AB + CD \rightarrow$ SOP

$= (AB + C)\,.\,(AB + D)$

$= (A+C)(B+C)(A+D)(B+D) \rightarrow$ POS

eg ⟹ $f = (A+B)(C+D) \rightarrow$ POS.

$= AC + AD + BC + BD \rightarrow$ SOP.

NOT gate

Karnaugh Map Simplification

→ It is a graphical approach of boolean fn Simplification.

⟹ Reduction in no. of literals & terms

→ Sol^n may not be unique.

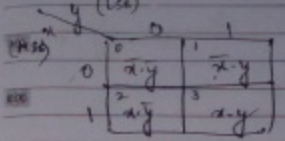
→ Simplified exp may be in SOP or POS form.

→ When we move from one row/column to next then only 1 bit diff should be there.

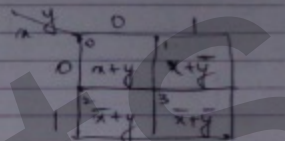
2008 JUNE Week 25 Date 173-193

IMPORTANT SATURDAY 21

2-variable K-Map

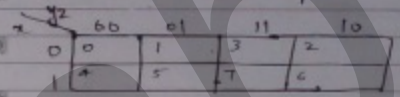


Minterm map



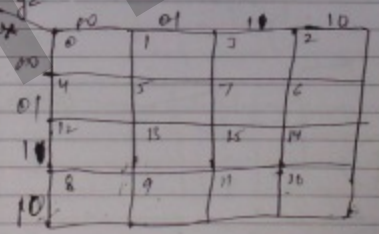
Maxterm map

3-variable K-Map



adjacents cells can be grouped together. (as they differ by only 1 bit)

4-variable K-map



→ If all the cells are filled with 1

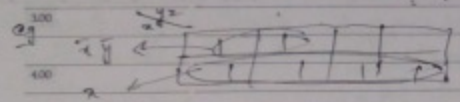
Date 174-192

SUNDAY 22

→ The no. of cells grouped is always a power of 2.

→ If all the cells in a k-map are grouped,
→ the o/p is always 1 (minterm map), or
→ " " " 0 (maxterm map).

No. of cells grouped	No of variables eliminated
2 (pair)	1
4 (quad)	2
8 (octet)	3
16	4
2^n	n

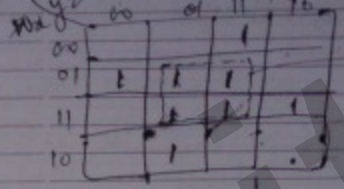


Priority of grouping

→ Octet
→ Quad
→ Pair
→ Single terms
→ Remove (redundant groups)

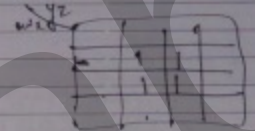
Redundant group → A group whose all the minterms are also grouped by adjacent groups.

Q Find the minimized logic expression



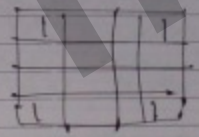
→ Redundant group

$= \bar{y}\bar{w}x + yz\bar{w} + \bar{y}\bar{z}w + yxw$

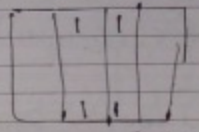


$w\,x\,y\,z$

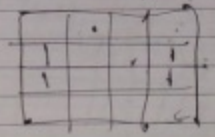
$x\,z$



$\bar{x}\,\bar{z}$

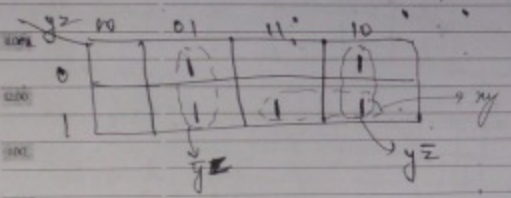


$\bar{x}\,z$

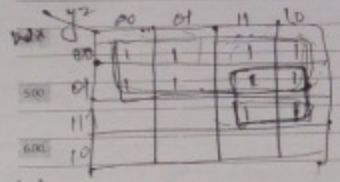


$x\,\bar{z}$

Boolean exp $\bar{x}yz + \bar{x}y\bar{z} + xy\bar{z} + x\bar{y}z + xyz$ can be simplified to
a) $x\bar{z} + xz + yz$
b) $xy + \bar{y}z + y\bar{z}$
c)
d)

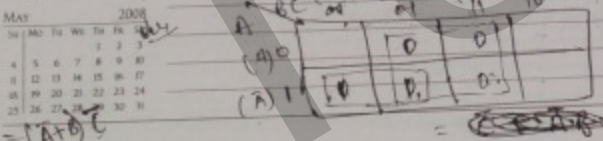


$f(w,x,y,z) = \bar{w}\bar{y} + wxy + \bar{w}y$ → most of minterms filled so may or may not be fully simplified.



Simplified form $= \bar{w} + xy$
minimal form $= \sum(0,1,2,3,4,5,6,7,12,15)$

Exp → $(A+B)(A+\bar{C})(B+\bar{C})$ Simplifies to



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

Don't Care Cond^n: (DCC's)

x	y	z	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1/0/x
1	1	1	1/0/x

x, y, z → [B.D.S] → f

→ Func^n that do not have the o/p specified for some i/p cond^n are known as incompletely specified functions.

→ As the digital system never enters these states, we may assume that o/p be 0 or 1.

→ These states are known as DCC's

→ DCC's provide further simplifica^n on the K-map

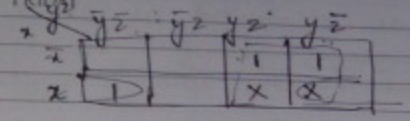
→ DCC's remain the same for both SOP & POS simplifica^n.

→ It is not necessary to group the DCC's.

A circuit implements boolean f = $xy + x\bar{y}\bar{z}$. It is found that the i/p combination $x = y = 1$ can never occur. Taking this into account, a simplified expression for f is given by:

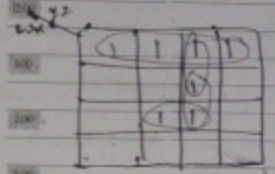
27 Day 179-187 Week 26 JUNE 2008

FRIDAY

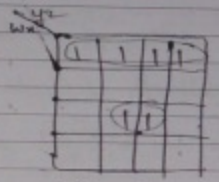


$= y + x\bar{z}$

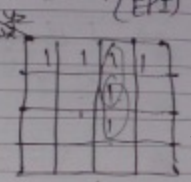
Prime Implicants (PI) and Essential Prime Implicants (EPI)



PI = 4 (All possible)
EPI = 2



EPI = 2
$f_{epi} = \bar{w}\bar{x} + wxz$
$f = f_{epi} + f_{pi}$
$= \bar{w}\bar{x} + wxz + \bar{w}yz$
or
$= \bar{w}\bar{x} + \bar{w}yz + wxz$



$f_{pi} = \bar{w}yz$
or
xyz

→ The pdt. terms obtained by grouping all the minterms in a K-map are known as PI.

→ PI's are obtained by considering all the biggest possible groupings in a k-map.

→ If a minterm is covered by only one PI, then that PI is known as EPI

MAY 2008

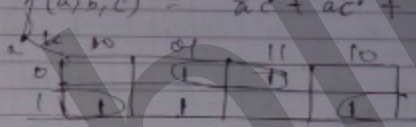
Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

2008 JUNE Week 26 Day 180-186 28

IMPORTANT SATURDAY

→ The simplified expression is obtained by including
(a) all the EPI's
+
(b) PI's covering those minterms that are not covered by EPI.

Q what are the EPI's of the foll
$f(a,b,c) = a\bar{c} + a\bar{c} + \bar{b}c$



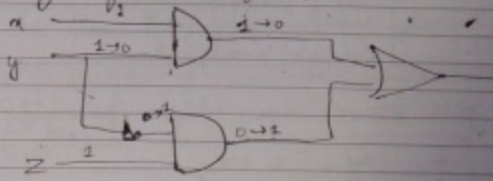
$= ac, a\bar{c}$

Hazards

→ It is a condition in which the o/p of circuit changes momentarily, whereas ideally it must not change.

Day 181-185 SUNDAY 29

→ It occurs b/c of the finite propagation delay of the logic gates.



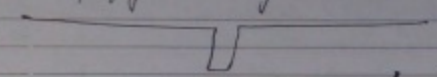
when y changes to 0 ...

JUNE 2008

Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

30 Day 182-184 Week 27 JUNE 2008
MONDAY IMPORTANT

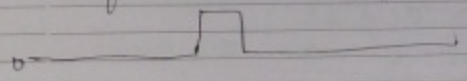
but due to propagation delay:



→ This type of hazard is "static 1" hazard.

→ If y goes from 1 to 0, the output 1 must remain at logic 1, but it momentarily goes to 0.
This is known as static-1 hazard.

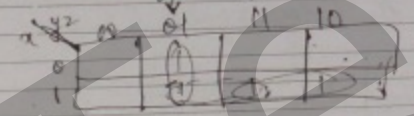
→ It occurs when ckt is implemented in the SOP form.



→ Input must remain at logic 0, but momentarily goes to 1, known as static-0 hazard.
POS form.

→

Elimination of Hazards

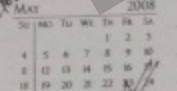


$= xy + \bar{y}z$ (static-1 hazard)

pros occurs when y goes from 1→0.
(add that expalso)

$f = xy + \bar{y}z + xz$

(b) hazard free circuit



2008 JULY Week 27 Day 183-183
IMPORTANT TUESDAY 01

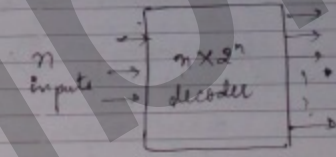
→ Hazards are eliminated by including the redundant groups in the ckt.

Combinational Circuits

→ A decoder is a combinational ckt that converts binary coded information into other codes (eg octal, decimal, hexadecimal, etc).

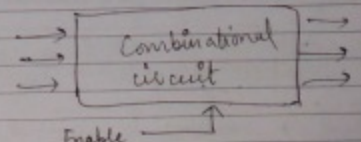
→ A binary code of n bits can represent $\leq 2^n$ distinct elements of information.

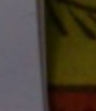
∴ a decoder has n inputs and $\leq 2^n$ outputs.



eg 3×8 decoder
4×10 "
4×16 "

ENABLE INPUTS





02 Day 184 052 Week 27 JULY 2008
WEDNESDAY IMPORTANT

→ Enable I/p has the same properties as any other input of the combinational ckt.

→ It is an optional feature which increases the flexibility of the circuit by becoming active or inactive.

→ Active high enable (E)
E = 1 ⇒ Active
E = 0 ⇒ Inactive

→ Active low enable (Ē)
Ē = 0 ⇒ active
Ē = 1 ⇒ inactive.

A 2×4 Decoder with active high enable

E	x	y	D_0	D_1	D_2	D_3
0	X	X	0	0	0	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	0	1

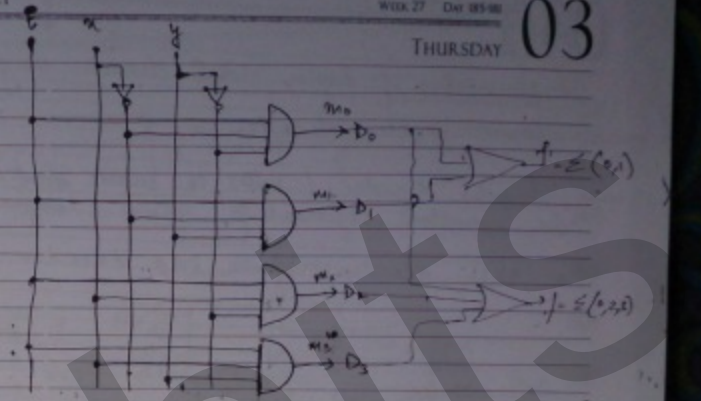
$D_0 = E \cdot \bar{x} \cdot \bar{y}$

$D_1 = E \cdot \bar{x} \cdot y$

$D_2 = E \cdot x \cdot \bar{y}$

$D_3 = E \cdot x \cdot y$

2008 JULY Week 27 Day 185 00 IMPORTANT
THURSDAY 03



→ An n × 2^n decoder generates all the 2^n possible minterms for n input variables.

→ At any instant, only 1 output is at logic 1. This output represents the minterm equivalent of the binary coded input.

Boolean Func" Implementn" using Decoder

→ Any Boolean f" can be represented in the sum of minterms form.

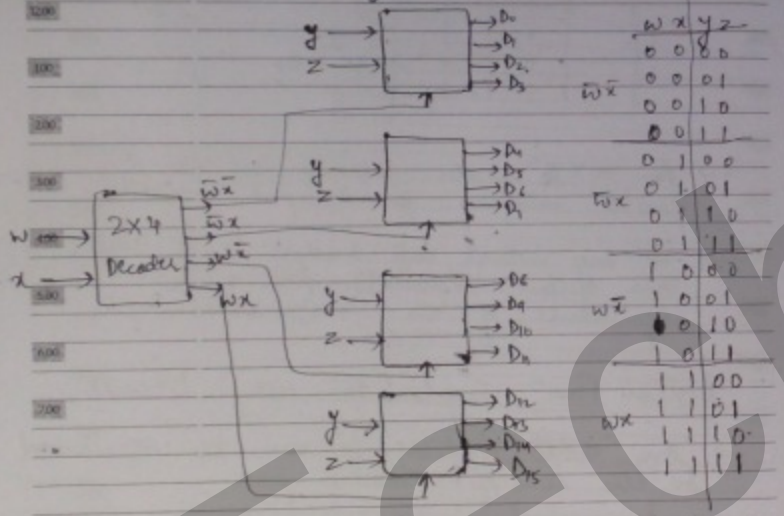
→ ∴ Boolean f" can be implemented by summing (ORing) the desired minterms.

04 Day 186-180 Week 27 JULY 2008
FRIDAY IMPORTANT

Implementation of higher order decoder using lower order decoders

→ Smaller decoder (with the enable input) can be connected together to obtain larger decoder.

→ 4X16 decoder from 2x1 decoder



w	x	y	z	
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	$\bar{w}\bar{x}$
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	$\bar{w}x$
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	$w\bar{x}$
1	1	0	0	
1	1	0	1	
1	1	1	0	wx
1	1	1	1	

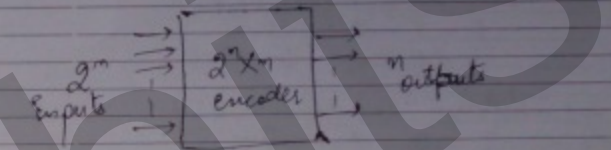
2008 JULY Week 27 Day 187-179
IMPORTANT 05
SATURDAY

ENCODER

→ It performs the inverse operation of decoder.

→ It generates the binary code corresponding to the input value.

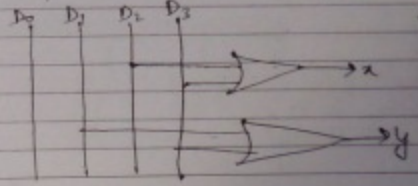
→ It has $\leq 2^n$ inputs & n outputs.



4 X 2 Encoder

D_0	D_1	D_2	D_3	x	y
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

$x = D_2 + D_3$, $y = D_1 + D_3$

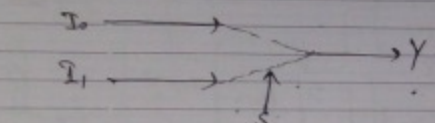


Day 188-178 06
SUNDAY

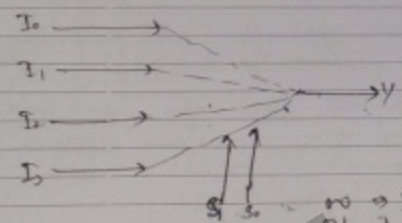
07 Day 189-177 Week 28 JULY 2008
MONDAY IMPORTANT

→ Encoder is a network of OR gates.
→ No of OR gate used = No. of outputs.

MULTIPLEXER



S = 0 ⇒ y = I0
= 1 ⇒ y = I1

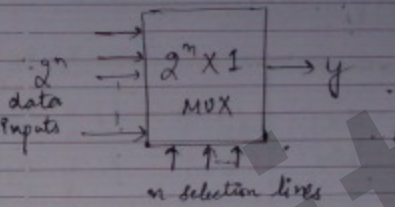


00 ⇒ I0
01 ⇒ I1
10 ⇒ I2
11 ⇒ I3

⇒ for n input lines we require $\log_2 n$ selection lines.

⇒ or for 2^n input lines, we require n selection lines.

2008 JULY Week 28 Day 190-176
IMPORTANT TUESDAY 08



Size of MUX : $2^n \times 1$
Input lines
output lines
But not the selection lines

Also known as
→ Data selector
→ Parallel to serial converter
→ Many to one circuit
→ Universal logic circuit.

x	y	f
0	0	A
0	1	B
1	0	C
1	1	D

$f = \bar{x}\bar{y} + x\bar{y}$

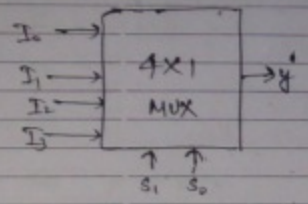
$f = \bar{x}\bar{y}A + \bar{x}yB + x\bar{y}C + xyD$

If output not given instead A, B, C, D all given then write in form of minterms all the input variables & multiply their values A, B, C, D

09 Day 191-175 Week 28 JULY 2008

WEDNESDAY IMPORTANT

4X1 MUX

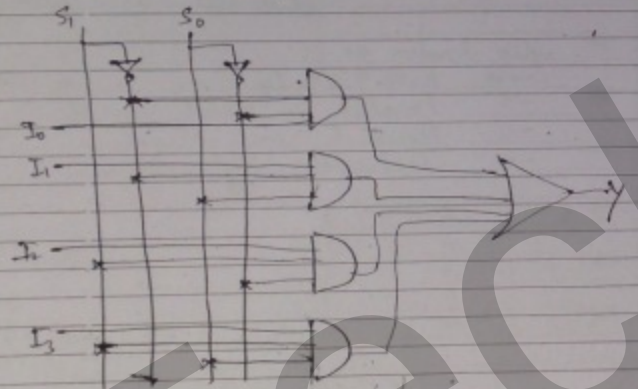


S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

function table

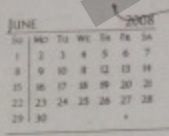
Boolean exp:

$$y = \bar{S_1}\,\bar{S_0}\,I_0 + \bar{S_1}\,S_0\,I_1 + S_1\,\bar{S_0}\,I_2 + S_1 S_0\,I_3$$



structure of selection bits

complete circuit

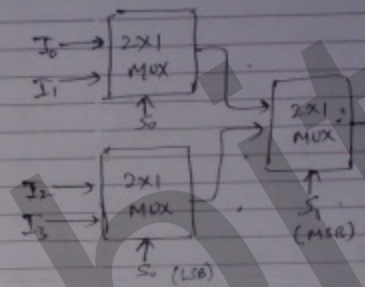


2008 JULY Week 28 Day 192-174

IMPORTANT THURSDAY 10

Implementation of higher order MUX from lower order MUX.

4X1 MUX using 2X1 MUX.



S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

S_1 = 0 ⇒ upper i/p to o/p
S_1 = 1 ⇒ lower i/p to o/p
} Illy for S_0

⇒ 3 2X1 MUX reqd for 4X1 MUX.

Illy

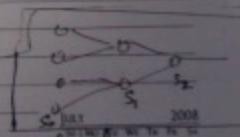
8X1	7	→ 2X1
16X1	15	→ 2X1
64X1	63	→ 2X1
2^nX1	$2^n - 1$	→ 2X1

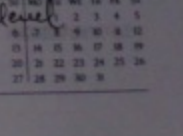
→ MUX required

S_1 S_0

MSB LSB

S_1 → more importance, ∴ will be applied to level where less no. of multiplexers are reqd.



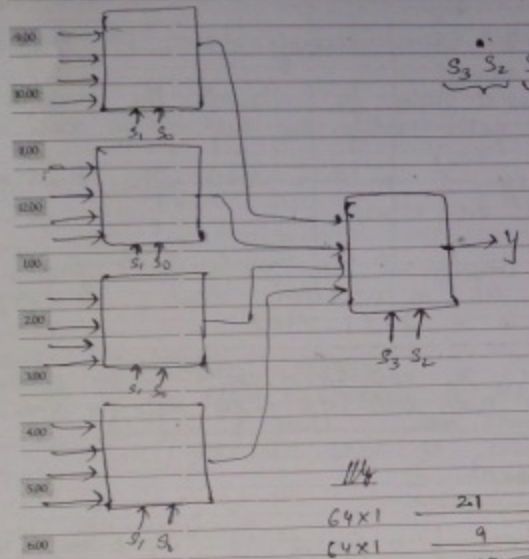


11 FRIDAY JULY 2008

Day 193-173 Week 28

IMPORTANT

16X1 MUX using 4X1 MUX:



S_3 S_2 S_1 S_0

11/4

64X1 → 21 → 4X1

64X1 → 9 → 8X1

256X1 → 17 → 16X1

To implement BX1 MUX → required AX1 MUX

$B/A = K_1$

$K_1/A = K_2$

$K_2/A = K_3$

$\frac{K_{n-1}}{A} = K_n = 1$

Never for 8X1 MUX using 4X1 MUX → Take MSB selection lines & extend.

2008 JULY Week 28 Day 194-172

IMPORTANT

12 SATURDAY

BX1 $\xrightarrow{K_1 + K_2 + \cdots + K_n}$ AX1

64X1 → 4X1

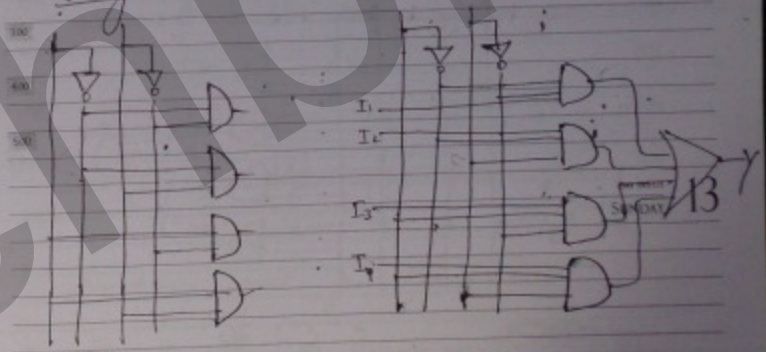
$\frac{64}{4} = 16$

$\frac{16}{4} = 4$

$\frac{4}{4} = 1$

→ 16+4+1 = 21 MUX req.

Implementation of Boolean func of N variable using MUX.



2X4 Decoder

A decoder is exactly same as the selection network of a multiplexer.

→ Decoder is a minterm generator circuit,
∴ In mux, the work of generation of minterm is done by selection network.

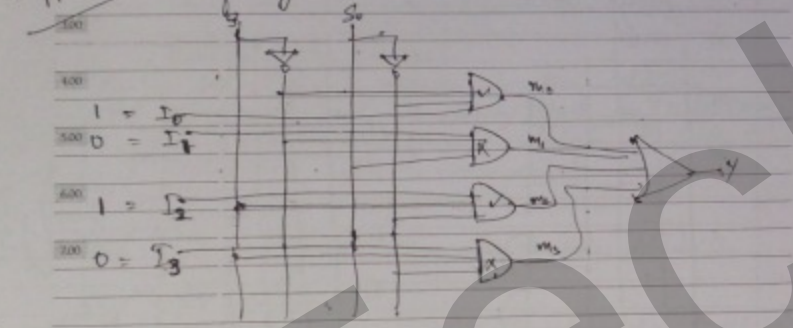
→ Size of the MUX can also be determined by using the n select lines.

→ $2^n \times 1$ if n is no. of select lines.
↳ Size of mux.

Boolean f^n implementation using MUX

Approach 1

$f(x,y) = \Sigma(0,2)$



→ Table

S1	S0	o/p
0	0	1
0	1	0
1	0	1
1	1	0

variable of f^n on selection line → data i/p line

→ The no. of variables in the f^n is equal to the
a) no. of i/p lines in decoder (for decoder)
b) no. of selection lines in mux

→ In MUX, we already have OR gate at the end,
∴ we'll need to disable the unreqd AND gates with the help of i/p data lines.

Apply i/p = 1 on those AND gates that are reqd, 0 on others.

→ To implement an n variable func^n, a MUX with n selection lines is reqd, that is a $2^n \times 1$ MUX.

Apply → The variables of the f^n on the selection line
→ truth table o/p on the data i/p lines.

Approach 2

$n \to 2^n \times 1$	$2 \to 2^2 \times 1 = 4 \times 1$	$3 \to 2^3 \times 1 = 8 \times 1$
$n \to 2^{n-1} \times 1$	$2 \to 2^{2-1} \times 1 = 2 \times 1$	$3 \to 2^{3-1} \times 1 = 4 \times 1$

An n variable f^n can also be implemented using a MUX having only (n-1) selection lines, that is $2^{n-1} \times 1$ MUX

16 Day 198 168 Week 29 JULY 2008

WEDNESDAY IMPORTANT

Q $f(x, y, z) = \Sigma(0,1,3,6)$

x	y	z	f	f ↔ z
0	0	0	1	f = 1 = I_0
0	0	1	1	
0	1	0	0	f = z = I_1
0	1	1	1	
1	0	0	0	f = 0 = I_2
1	0	1	0	
1	1	0	1	f = $\bar{z}$ = I_3
1	1	1	0	

Limitation in approach 2

→ If the term $\bar{z}$ appears in the relation between f and z, then an additional NOT gate is reqd to obtain $\bar{z}$ from z.

Summarizing

Approach 1 → A 2^n X1 MUX can implement ALL n variable f°.

Approach 2 with limitation → A 2^{n-1} X1 MUX can implement only SOME n variable f°.

Application → A 2^{n-1} X1 MUX and a NOT gate can implement ALL n variable f°.

JUNE 2008

Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

2008 JULY Week 29 Day 199 167

IMPORTANT THURSDAY 17

Trick (short method) for finding sele° I/p for f°z.

(1) I_0 I_1 I_2 I_3 — x y (z)

$\bar{z}$ | 0 | 2 | 4 | 6 |
z | 1 | 3 | 5 | 7 |
1 z 0 $\bar{z}$

(2) I_0 I_1 I_2 I_3 — x (y) z

y | 0 | 1 | 4 | 5 |
y | 2 | 3 | 6 | 7 |
$\bar{y}$ 1 y 0

(3) I_0 I_1 I_2 I_3 — (x) y z

$\bar{x}$ | 0 | 1 | 2 | 3 |
x | 4 | 5 | 6 | 7 |
$\bar{x}$ $\bar{x}$ x $\bar{x}$

Numberisy concl from truth table

	x	y	z
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

JULY 2008

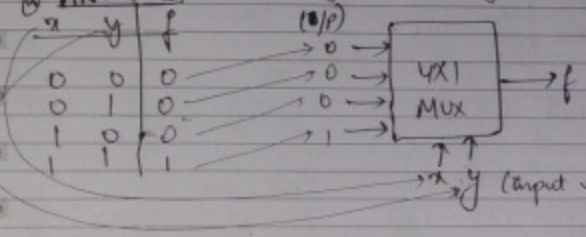
Su	Mo	Tu	We	Th	Fr	Sa
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

Implementation of standard logic gates using MUX:

(1) Approach 1 (Using MUX having n-selection lines)

(a) AND Gate

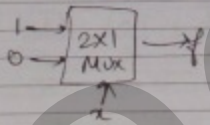
x	y	f
0	0	0
0	1	0
1	0	0
1	1	1



Hly the OR, NAND, NOR, EXOR & EXNOR gates are implemented.

(b) NOT Gate

x	f
0	1
1	0



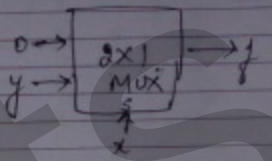
(2) Approach 2 (Using MUX having n-1 selection lines)

→ This approach is not applicable for NOT gate.

Important & Must

(a) AND gate

x	y	f	f → y
0	0	0	f = 0
0	1	0	
1	0	0	f = y
1	1	1	



→ Remaining logic gates can be Hly implement

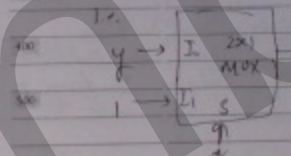
Must

(b) → For OR Gate

$f = x + y$

$f = I_0 = y$

$f = I_1 = 1$



Now we analyze it

$f = \bar{S} I_0 + S \cdot I_1$

$= \bar{x} \cdot y + x \cdot 1$

$= (x + \bar{x})(x + y)$

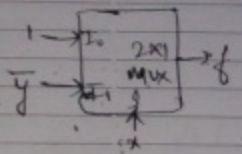
$= 1 \cdot (x + y)$

$= x + y$

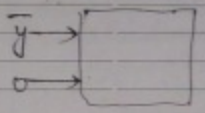
- So verified, it is implementing OR operation.

21 Day 203 163 Week 30 JULY 2008
MONDAY IMPORTANT

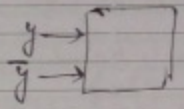
(a) NAND gate



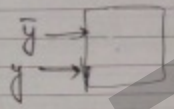
(b) NOR gate



(c) XOR gate



(d) XNOR



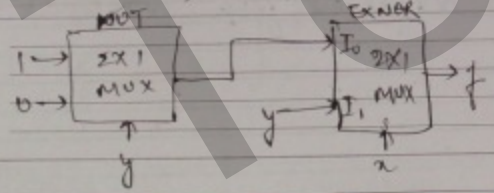
Note:

To obtain the NAND, NOR, EXOR & EXNOR gates, two 2x1 muxes are reqd.

The first 2x1 MUX is reqd to obtain $\bar{y}$ from y

But it requires two 2x1 mux to implement it, as we don't have NOT gate.

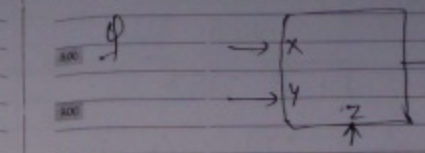
To implement it:



JUNE 2008

Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

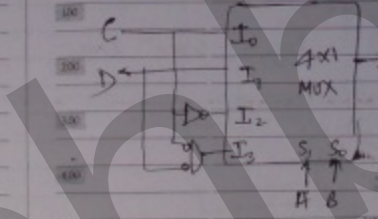
2008 JULY Week 30 Day 204 162
IMPORTANT TUESDAY 22



x	y	z	
T	T	I	T
T	T	R	T
T	R	D	T
R	O	T	

Compare T+R with OR gate & get ans

Boolean f realized by this logic ckt shown is:



$\Rightarrow f = \bar{S}_1\bar{S}_0 I_0 + \bar{S}_1 S_0 I_1 + S_1\bar{S}_0 I_2 + S_1 S_0 I_3$

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

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

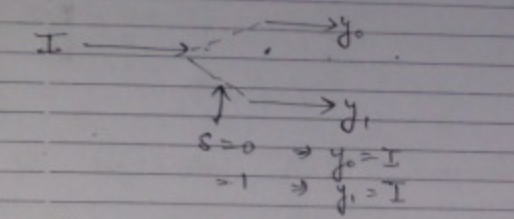
$= \bar{A}\bar{B}CD + \bar{A}\bar{B}C\bar{D} + \bar{A}BCD + \bar{A}B\bar{C}D + AB\bar{C}D + AB\bar{C}\bar{D} + AB\bar{C}\bar{D}$

$= 3, 2, 7, 5, 9, 8, 12$

JULY 2008

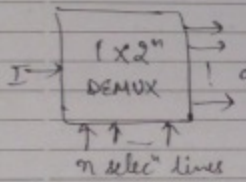
Su	Mo	Tu	We	Th	Fr	Sa
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

DEMULTIPLEXER



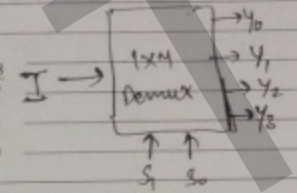
→ If 2^n → output lines
Then n selection lines.

n → output ⇒ $\log_2 n$ selⁿ lines



Also known as;
→ Data distributer

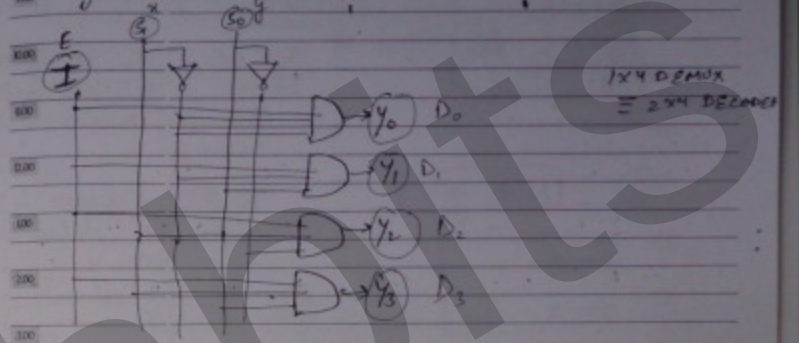
1 X 4 DEMUX



S_1	S_0	y_0	y_1	y_2	y_3
0	0	I	0	0	0
0	1	0	I	0	0
1	0	0	0	I	0
1	1	0	0	0	I

$y_0 = \overline{S_1} \cdot \overline{S_0} \cdot I$ $y_1 = \overline{S_1} \cdot S_0 \cdot I$

$y_2 = S_1 \cdot \overline{S_0} \cdot I$ $y_3 = S_1 \cdot S_0 \cdot I$



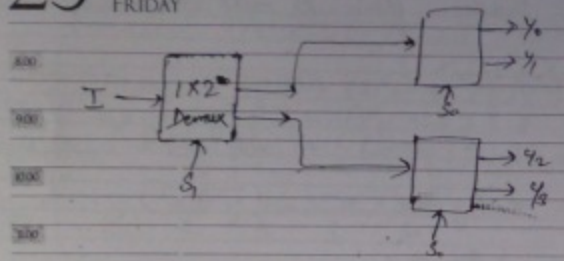
→ Selection network of MUX & DEMUX is same.

→ A decoder (with enable input) is same as DEMUX.
∴ the ckt is known as Decoder demultiplexer

→ An $n X 2^n$ decoder is equivalent to a $1 X 2^n$ DEMUX.

Implementation of higher order Demux using lower order Demux

① 1X4 Demux from 1X2 Demux.



- AXB decoder → using → CXD decoder
- ⇒ 1XB demux → 1XD demux
- ⇒ BX1 mux → DX1 mux

Q How many 3X8 decoders with an enable i/p are needed to construct a 6X64 decoder w/o using any other logic gate.

Sol 6X64 → using → 3X8

1X64 demux → 1X8 demux

8X1 mux → 8X1 mux

⇒ $\frac{64}{8} = 8$

$\frac{8}{8} = 1$

8+1 = 9

Half Adder

→ Adds 2 bits.

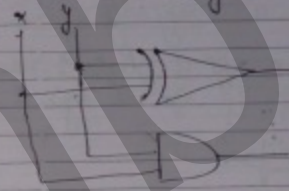
(Augend) x → Half Adder → S (Sum)

(Addend) y → Half Adder → C (Carry)

x	y	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

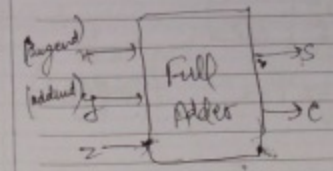
⇒ $S = \bar{x}y + x\bar{y}$ $\quad C = x \cdot y$

$= x \oplus y$



Full Adder

→ Adds 2 bits, considering a carry from the previous stage.



no. of 1's (out)	x	y	z	S	C
0	0	0	0	0	0
1	0	0	1	1	0
1	0	1	0	1	0
2	0	1	1	0	1
1	1	0	0	1	0
2	1	0	1	0	1
2	1	1	0	0	1
3	1	1	1	1	1

(x+y)+z

→ z is the carry from the previous stage
→ C . . . to the next stage.

→ $S = \Sigma(1,2,4,7)$
$= \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z} + xyz$

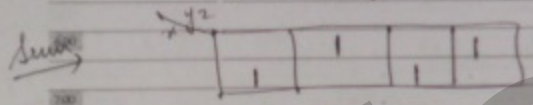
∴ Sum is an odd f^n

→ $C = \Sigma(3,5,6,7)$
$= \bar{x}yz + x\bar{y}z + xy\bar{z} + xyz$

Carry is a majority func^n

Majority f^n → The o/p is logic 1 when majority no. of i/p are at logic 1.

K-Map Simplifica^n



Simplifica^n is not possible for the sum.

Carry

$C = xy + yz + xz$ (Simplifica^n)

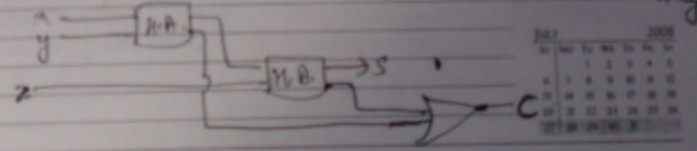
Full Adder from 2 half Adders

$S = \Sigma(1,2,4,7)$
$= \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z} + xyz$
$= \bar{x}(\bar{y}z + y\bar{z}) + x(\bar{y}\bar{z} + yz)$
$= \bar{x}(y \oplus z) + x(\overline{y \oplus z})$
$= x(y \oplus z) + x(\bar{y} \oplus z)$
$= x \oplus y \oplus z$

} manipulation not Simplifica^n

$C = \Sigma(3,5,6,7)$
$= \bar{x}yz + x\bar{y}z + xy\bar{z} + xyz$
$= z(\bar{x}y + x\bar{y}) + xy(z + \bar{z})$
$= z(x \oplus y) + xy$

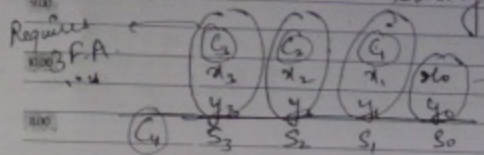
] Manipulation



Important

Parallel Adder / Binary Adder

→ Adds 2 n-bit binary numbers.



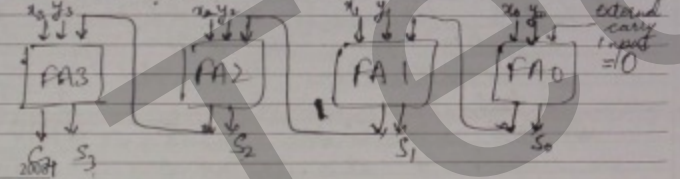
Requirements for 4-bit addition

→ 3 FA & 1 H.A., or
→ 4 F.A. or
→ 7 H.A., 3 OR gates

∴ In general, for n-bit add^n :

→ (n-1) F.A. & 1 H.A
→ n F.A.
→ (2n-1) H.A & (n-1) OR gate

4-bit Parallel Adder



Output = C_4 S_3 S_2 S_1 S_0

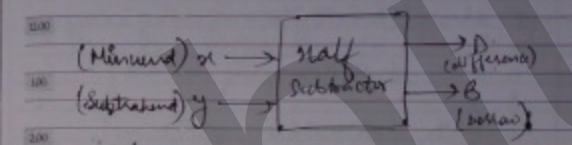
Important

→ Cost is higher for a carry lookahead adder compared to ripple carry adder.

→ The carry bit propagate from the L.S.B. to the M.S.B position in a ripple manner, therfore the ckt. is known as Ripple carry adder.

Half Subtractor

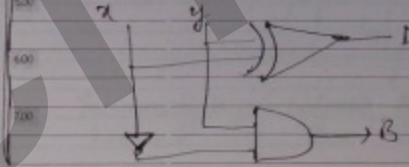
→ Subtracts 2-bits



x	y	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

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

$B = \bar{x}y$



Full Subtractor

Subtracts 2 bits, considering a borrow given to the previous stage.

01 FRIDAY — AUGUST 2008 — IMPORTANT

(Minuend) x, (Subtrahend) y, z (borrow given to previous stage) → Full Subtractor → D (Difference), B (Borrow taken from next stage)

x	y	z	D	B
0	0	0	0	0
1	0	0	1	1
2	0	1	1	1
3	0	1	1	1
4	1	0	0	1
5	1	0	0	0
6	1	1	0	0
7	1	1	1	1

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

$D = \bar{x}\bar{y}z + x\bar{y}\bar{z} + \bar{x}y\bar{z} + xyz$

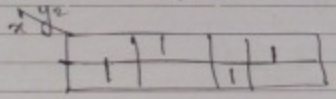
∴ difference is an odd fn

$B = \Sigma(1,2,3,7)$

$= \bar{x}\bar{y}z + x\bar{y}z + \bar{x}yz + xyz$

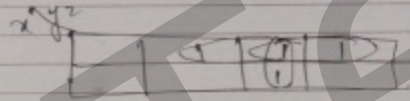
K-map Simplifica"

Diffe"



∴ Simplifica" not possible in difference

Borro"

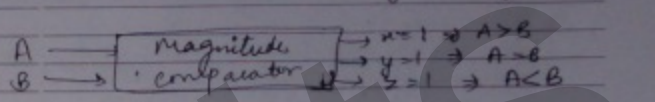


$B = \bar{x}y + yz + x\bar{z}$

2008 AUGUST — Week 31 Day 215/151 — IMPORTANT — SATURDAY 02

Magnitude Comparator

→ Determines the relative magnitude of 2 numbers

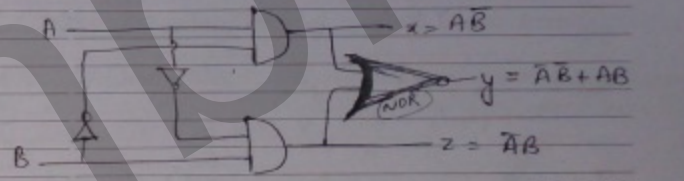


A	B	x	y	z
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

$x = A\bar{B}$

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

$z = \bar{A}B$



Day 216/150 — SUNDAY 03

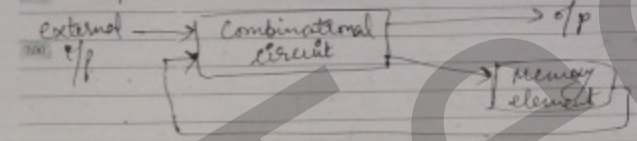
Q The output 4 of a 2 bit comparator is logic 1 whenever the 2 bit i/p A is greater than the 2 bit input B. The no. of combinations for which of i/p is logic 1.

a) 4 b) 6 c) 8 d) 10

A		B		y
a_0	a_1	b_0	b_1	
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

6 tyms

Sequential Circuits



→ The binary data stored in the memory elements is known as the state.

→ Previous state → Q_{n-1}
→ Present state → Q_n
→ Next state → Q_{n+1}

→ The o/p depends upon
  → External input i/p or
  → present state

→ The next state depends upon
  → external i/p i/p or
  → present state

Types of Sequential Circuits

① Synchronous Seq. Ckt.

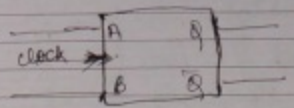
All the memory elements are triggered simultaneously by a common activating signal known as the clock pulse.

② Asynchronous Seq. Ckt.

The memory elements operate at random instants of time independent of each other.

Flip flop

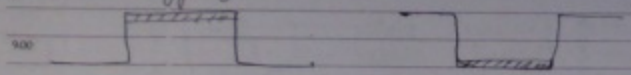
(Binary cell)



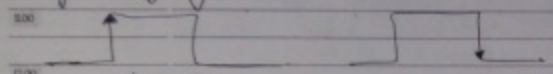
Combinational → only level triggering
Seq. → both edge & level triggering
flipflop

06 Day 219/147 Week 32 AUGUST 2008
WEDNESDAY IMPORTANT

Level triggering



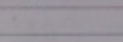
Edge triggering



+ve edge −ve edge

→ Set = 1
Reset = 0

→ Binary cell that stores 1 bit
→ May have one or two data inputs.
→ Triggered by a clock pulse.
→ +ve edge triggering
→ −ve "

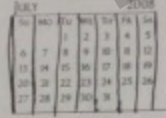


→ Has 2 outputs
→ Q ⇒ Normal o/p
$\overline{Q}$ ⇒ Complement o/p

→ Under normal operations, Q and $\overline{Q}$ are complements of each other.

→ Has 2 stable states
∴ the ckt acts as bistable multivibrator.

→ Set state / 1- state
→ Q = 1 & $\overline{Q}$ = 0



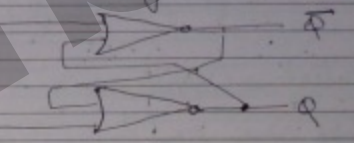
2008 AUGUST Week 32 Day 220/146 07
IMPORTANT THURSDAY

→ Reset state / 0-State / Clear state
→ Q = 0 & $\overline{Q}$ = 1.

SR latch

→ Consists of 2 cross-coupled NOR or NAND gates
→ Has 2 inputs
→ S ⇒ Set
→ R ⇒ Reset
→ Has 2 o/p's
→ Q
→ $\overline{Q}$
→ Different truth table for NOR & NAND gates.

① SR LATCH using NOR gates



when S=0 & R=0

$\overline{Q}$	0 → 0	1 → 1
Q	1 → 1	0 → 0

when S=0 & R=1

$\overline{Q}$	0 → 1	1 → 0 / 1
Q	1 → 0	0 → 0

S=1, R=0

$\bar{q}$	0 → 0	1 → 0
q	1 → 1	0 → 1

S=1, R=1

$\bar{q}$	•	$q = \bar{q} = 0$ = Invalid state
q		

S	R	Q_{n+1}	
0	0	Q_n	(Hold State)
0	1	0	(Reset)
1	0	1	(Set)
1	1	(Invalid)	$Q = \bar{Q} = 0$

→ Invalid state does not mean that there is no output at the Q and $\bar{Q}$ terminals. It only means that $Q = \bar{Q}$, a situation to avoid under normal condition.

Hold State After Invalid State

S	R	Q	$\bar{Q}$	
0	1	0	1	
0	0	0	1	
1	0	1	0	
0	0	1	0	
1	1	0	0	
0	0	0	0	×

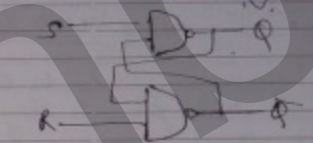
will not be able to find the exact state i.e. either 0,1 or 1,0
So, system is in Indeterminate state.

S	R	Q	$\bar{Q}$
①	1	0	0
0	②	0	1
0	0	0	1

S	R	Q	$\bar{Q}$
1	①	0	0
①	0	1	0
0	0	1	0

→ If SR = 00 is applied after SR = 11, then either $Q = 0$ & $\bar{Q} = 1$ or $Q = 1$ & $\bar{Q} = 0$ the FF enters an Indeterminate state

SR latch using NAND Gates



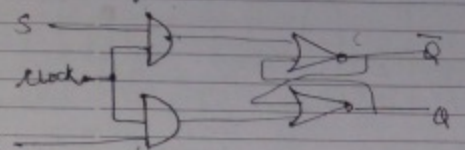
S	R	Q_{n+1}	
0	0	$Q = \bar{Q} = 1$	(Invalid)
0	1	1	(Set)
1	0	0	(Reset)
1	1	Q_n	(Hold)

→ To get a particular o/p, the i/p reqd by NAND latch are complements of those reqd by NOR latch.

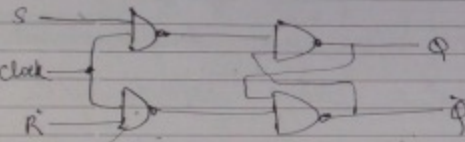
In an SR latch made by cross-coupling two NAND gate, if both S & R i/p are at 0, then it will result in

a) $Q = 0$ $\bar{Q} = 1$
b) $Q = 1$ $\bar{Q} = 0$
c) $Q = 1$ $\bar{Q} = 1$
d) Indetermined state

SR - flip flop



using NOR latch



using NAND latch

why using AND gate

This is b/c when clock is applied to 0, then we want the NOR gate to stay in the idle state.
Now the idle state or hold state of NOR is (0, 0) ∴ we require a gate which irrespective of the I/p applied makes the o/p 0 when clock is 0.

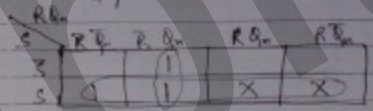
clock	S	R	Q_{n+1}
0	x	x	Q_n
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	Invalid

Function Table

Characteristic Table

S	R	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	X
1	1	1	X

Characteristic Eqn



$Q_{n+1} = S + \bar{R} Q_n$ when $S + R \neq 1$

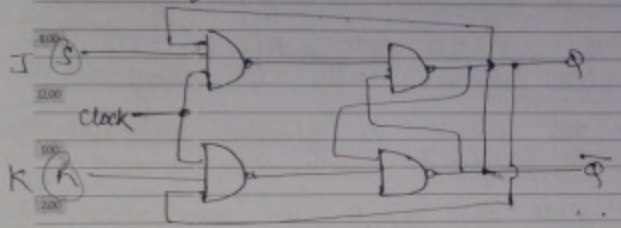
Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Excitation Table

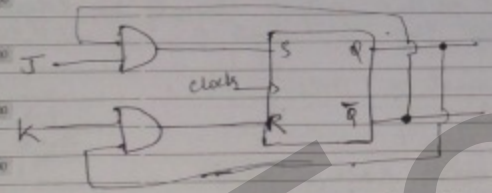
→ Derived from characteristic table

J K flip flop

Resolves the problem of invalid state of SR flip flop.

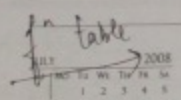


using logic gates

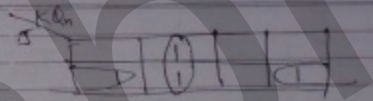


using SR flip flop

Clock	J	K	Q_{n+1}
0	X	X	Q_n
1	0	0	Q_n
↓	0	1	0
↓	1	0	1
↓	1	1	$\bar{Q}_n$ → Toggle state



J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



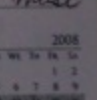
$Q_{n+1} = J\bar{Q}_n + \bar{K}Q_n$

Q_n	Q_{n+1}	J	K
0	1	1	X
0	0	1	X
1	1	X	1
1	0	↓ X	0

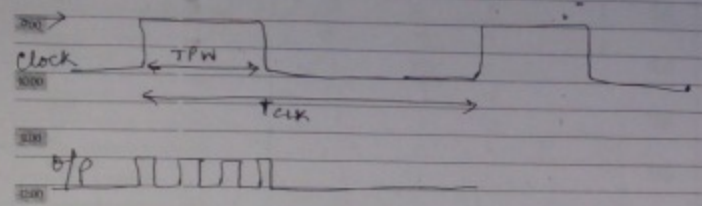
Race Around Cond. in a JK flip flop

→ The o/p of the flip flop changes more than once whereas ideally it must change only once

→ It occurs when,
  → J = K = 1
  → Propagation delay of flip flop > Clock Pulse width



$\Rightarrow T_{PD} < T_{PW}$



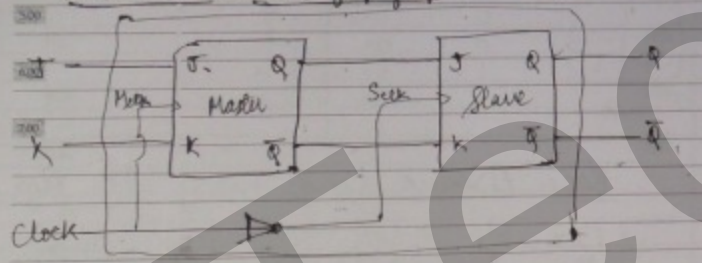
→ This problem is resolved by ensuring the following conditions,

$T_{PW} < T_{PD} < T_{clk}$

Or in worst case

$T_{PW} \leq T_{PD} < T_{clk}$.

Master Slave JK flip flop



→ The overall ckt acts as a single flip flop & stores 1 bit.

→ The master is triggered when the clock goes to 1.

→ The slave is " " " " goes to 0

→ At any instant, only one flip flop is triggered

→ The external data enters the MS flip flop when the master is triggered.

→ The output is generated when the slave is triggered.

→ Both the inputs are never equal to 1. ∴ the problem of race around cond. is not present in a MS JK FF.

Q. The present output Qn of an edge triggered JK flip flop is logic 0. If J=1, then Qn+1

a) cannot be determined.
b) logic 0
c) logic 1
d) ~~write~~ race around

J	K	Q_{n+1}
1	0	1
1	1	$\bar{Q}_n$

D flip flop will retain the value while buffer not.

Buffer eg

D flip flop

18 MONDAY August 2008

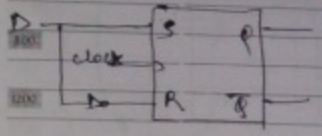
Day 231-135 Week 34

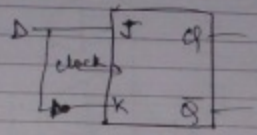
action controlled clkt

So D flip flop is a memory element

D flip flop

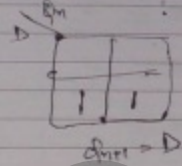
→ output follows the input





Clock	D	Q_{n+1}
0	X	Q_n
1	0	0
1	1	1

D	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1



$Q_{n+1} = D$

Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

2008 August

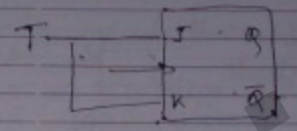
Important

Week 34 Day 232-134

Tuesday 19

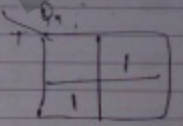
toggle

T flip flop



Clock	T	Q_{n+1}
0	X	Q_n
1	0	Q_n
1	1	$\bar{Q}_n$

T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

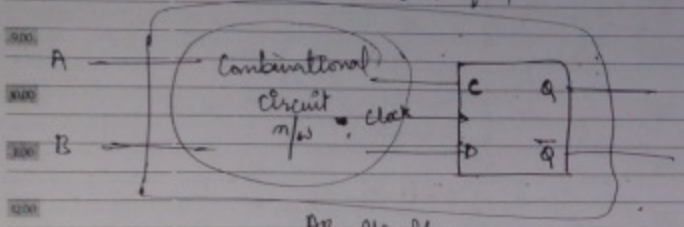


$Q_{n+1} = T\bar{Q}_n + \bar{T}Q_n$

$= T \oplus Q_n$

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Conversion b/w flip flops



AB flip flop

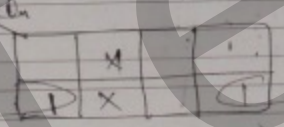
→ Obtain JK from SR flip flop

J	K	Q_n	Q_{n+1}	S	R
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	X	0
1	1	0	1	1	0
1	1	1	0	0	1

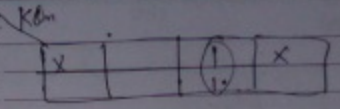
Char. table of JK flip flop

clear of JK

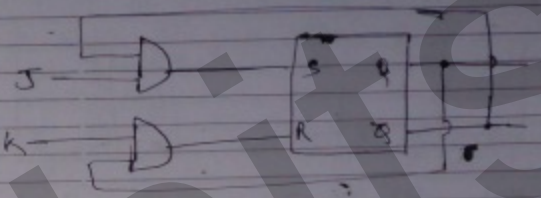
excat eq SR

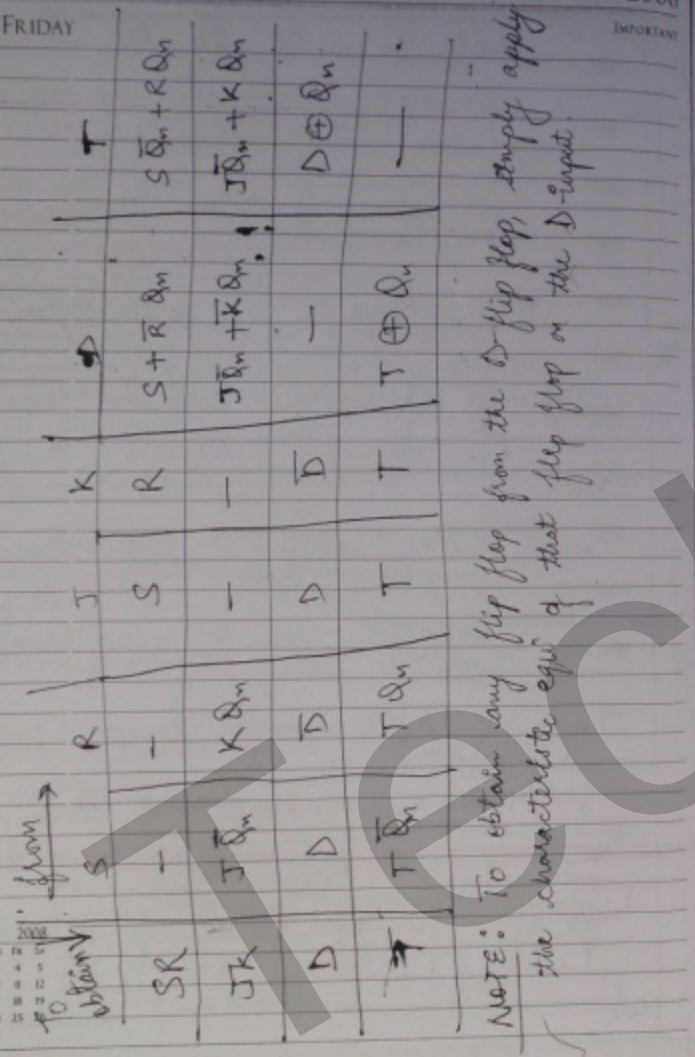


$S = J\bar{Q}_n$



$R = KQ_n$





Registers & Counters

Registers

→ A register is a group of flip flops.

→ A single flip flop stores 1-bit.
∴ a register containing n flip flops (n-bit register) stores n bits.

→ Register are essentially synchronous systems.

→ Feeding of new data is known as Loading/Updating the register.
  → SERIAL loading
  → PARALLEL loading

Classification of Registers

(i) Based on Application
  → Storage Register
  → Shift register: a) It shifts the data from one flip flop to its adjacent flip flop.
  b) The output of one flip flop is connected to the I/p of the adjacent flip flop.

Day 237-128 Sunday 24

(ii) Based on modes of inputs & outputs
  → SISO (Serial in Serial out)
  → SIPO
  → PISO
  → PIPO

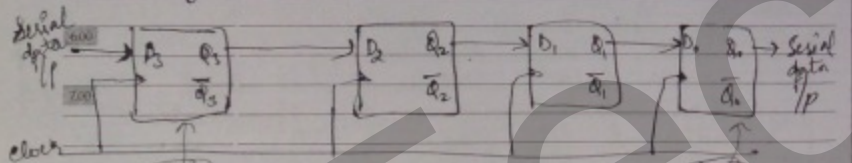
(B) Based on Direction Of Operation
(a) Unidirectional (left shift, right shift)
(b) Bidirectional

Clock Pulse Reqt. for an n-bit register

→ for serial data
→ to input n bits — n clock pulses reqd.
→ to output n bits — n-1 clock pulse
– SR, LS to always available at o/p & clock pulse not reqd to read out
→ for parallel data
→ to input n bit — 1 clock pulse
→ to output n bit — 0 clock pulse

SISO Register

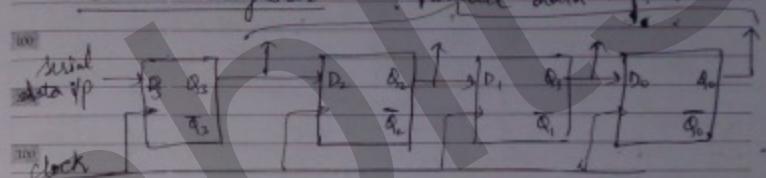
A 4-bit, unidirectional right shift, SISO register, using D flip flops is shown



If 1011 is to be transfer
So first B is transfered inside

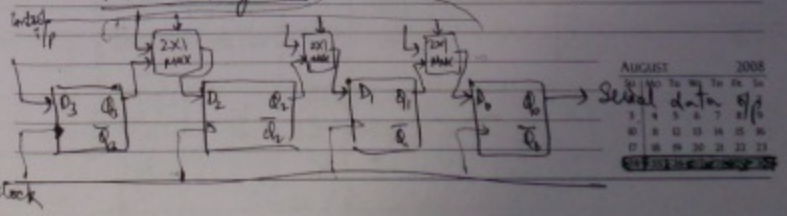
Clock	Q_3	Q_2	Q_1	Q_0
0	1	0	1	1
1	0	1	0	1
2	0	0	1	0
3	0	0	0	1
4	0	0	0	0

SIPO Register



→ Serial data is also known as temporal code.
→ Parallel data is also known as spatial code.
→ ∴ SIPO converts temporal code into spatial code.

PISO Register

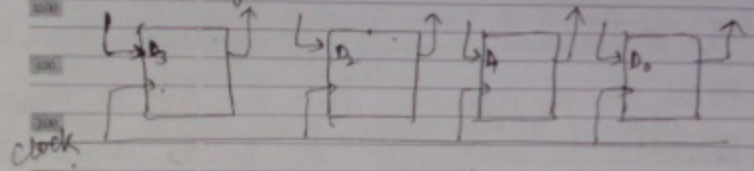


27 Day 240-026 Week 35 AUGUST 2008
WEDNESDAY IMPORTANT

Control	Operation
0	Parallel data i/p
1	Serial data o/p.

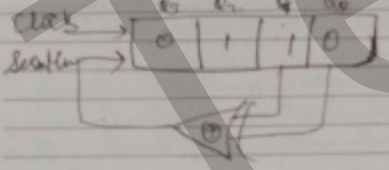
PISO converts spatial code to temporal code.

PIPO register



→ It can act only as a storage register but not shift register.
∴ mostly used as storage register in microprocessor due to its high speed.

The initial contents of the 4-bit SISO right shift register is 0110. After 3 clock pulses are applied the contents of the shift register will be:



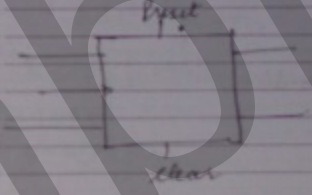
a) 0000
b) 0101
c) 1010
d) 1111

2008 AUGUST Week 35 Day 240-025 28
IMPORTANT THURSDAY

clock	Q_3	Q_2	Q_1	Q_0	
0	0	1	1	0	(1 0 0)
1	1	0	1	1	(1 0 1)
2	0	1	0	1	(0 1 0)
3	1	0	1	0	(1 0 0)

COUNTERS

Preset & clear inputs



(When we want to forcibly set or reset the flip flop then we have preset & clear inputs)

→ Also known as,
→ Direct Inputs
→ Asynchronous Inputs

→ Sets / Resets the flip flop irrespective of the clock inputs & the data inputs.

Preset / Direct Set	Clear / Direct Reset
→ Sets the output.	→ Resets the output
→ Active high preset (by applying logic 1)	Active high clear (by applying logic 1)
→ Active low preset ($\overline{\text{Preset}}$) (by applying logic 0)	Active low clear ($\overline{\text{clear}}$) (by applying logic 0)

29 Day 242·124 Week 35 AUGUST 2008

FRIDAY

COUNTERS

→ A counter is a special type of register that follows a predetermined sequence of states upon the application of input pulses applied on the clock input terminals of the flip flops.

→ The input pulses may either be the clock pulse or they may originate from some other external source.

→ In a counters,
→ if the no. of flip flops = n
& no. of states = N
then $N \leq 2^n$
→ MOD N counter
→ Divide-by-N counter

Applications of Counters

→ To count the no. of clock pulses
→ as a freq. divider

f → [MOD N] → f/N

→ In general

[MOD A] → [MOD B] →
↓
f → [MOD AB] → f/AB

2008 AUGUST Week 35 Day 243·123

IMPORTANT

30 SATURDAY

The min. no. of D flip flops needed to design a MOD-258 counter is:

a) 9 b) 8 c) 512 d) 258

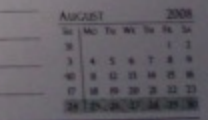
$2^9 = 512$ ✓ $\times 2^8 = 256$ states

258 used, 254 unused

Types of Counters

Synchronous	Asynchronous / Ripple Counter
→ All the flip flops are triggered simultaneously	The flip flops are triggered either by a clock pulse or by the outputs of adjacent flip flops.
→ Faster (as in 3 FFs, all the FF are provided as clock is applied to each flip flop)	Slower
→ Up & down & random counters possible	Only up & down counters possible.
→ Decoding errors are not possible.	Decoding errors are present

Day 244·122 SUNDAY 31



01 DAY 245 (21) WEEK 36 SEPTEMBER 2008

MONDAY IMPORTANT

Ripple Counter (Asynchronous counter)

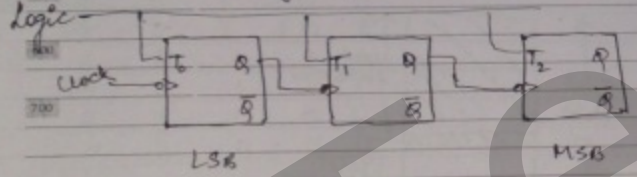
→ It consist of a series connection of T flip flops (or other flip flops converted into T flip flops).

→ All the T flip flops always operate in the toggle mode (T=1).

→ The clock pulse is applied to only one flip flop. This flip flop represents the LSB of the counting sequence.

→ The output of each flip flop is connected to the clock input of the next higher order flip flop.

3-bit Binary Up Counter



State Transition Conditions

(a) Q_0 toggles on every clock pulse.
(b) Q_1 toggles when Q_0 goes from 1 to 0.
(c) Q_2 toggles when Q_1 goes from 1 to 0.

AUGUST 2008

Su	Mo	Tu	We	Th	Fr	Sa
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

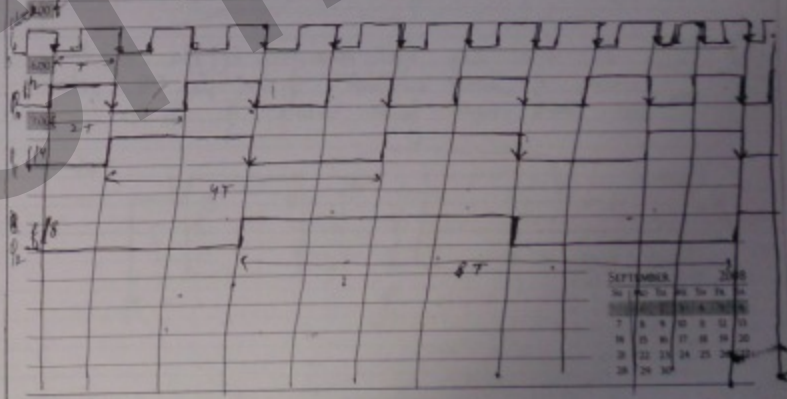
2008 SEPTEMBER WEEK 36 DAY 246 (20) 02

IMPORTANT TUESDAY

State Transition Table / State Table

Clock	Q_2	Q_1	Q_0	
0	0	0	0	0
1	0	0	1	1
2	0	1	0	2
3	0	1	1	3
4	1	0	0	4
5	1	0	1	5
6	1	1	0	6
7	1	1	1	7
8	0	0	0	
9	0	0	1	
10	0	1	0	
11	0	1	1	

Timing Diagram



SEPTEMBER 2008

Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

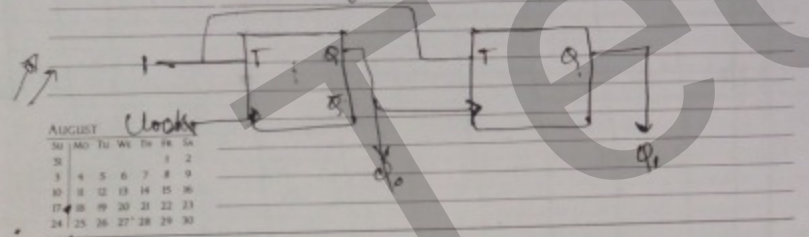
→ freq. of MSB pulse is getting 1/8.
→ where freq. is the freq. of the clock.

Flip flop o/p	Clock triggering	Counter operation
Q / $\bar{Q}$	+ve / -ve	Up / Down
0 / 0	0	0
0 / 1	1	1
1 / 0	0	1
1 / 1	1	0

EX-OR

Q → +ve, $\bar{Q}$ → -ve
+ve → -ve, -ve → +ve
-2 -1 0 1 2 (up / down)

Q. In the seq. ckt shown, if the initial value of the output Q_1Q_0 is 00, what are the next four values of Q_1Q_0?



Ans. 11, 10, 01, 00
as it is down counter. (2 bit).

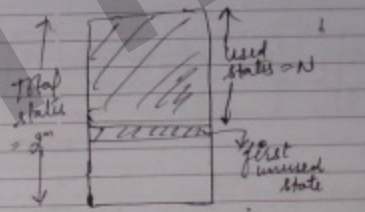
BCD Ripple Counter

→ Also known as : Decimal counter
Decade counter
MOD 10 counter
Divide by 10 counter

→ Follows a sequence of 10 states from 0 to 9.

→ Total no of flip flops = $n = 4$
→ Total no of states = $2^n = 16$
used = $N = 10$
unused = 6

Underline principle in a Design of a counter having unused states

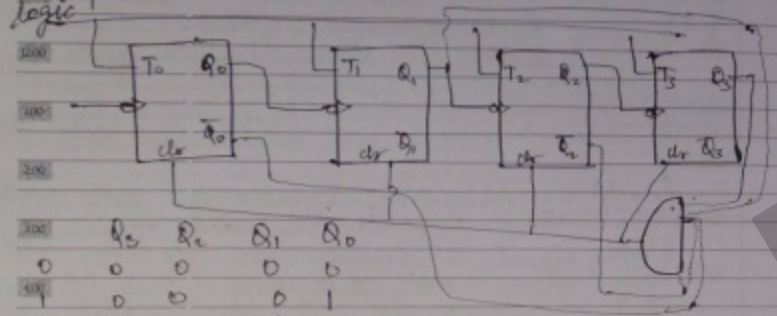


→ for up counter
a) clear all the flip flops of the counter on reaching the first unused state.

This begins the counting sequence from 0000.

→ for down counter
Preset all the flip flops of the counter on reaching the first unused state. This begin the counting sequence from 1111.

logic 1



	Q_3	Q_2	Q_1	Q_0
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
10	1	0	1	0

First unused state = $Q_3 \bar{Q}_2 Q_1 \bar{Q}_0 = 1$

→ If Q_3, Q_1 are set to 1, then also result is verified.

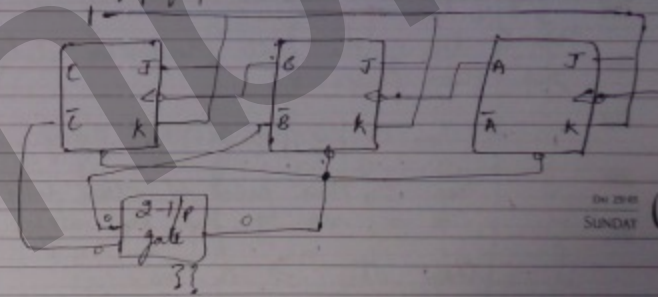
→ Or using active low clear, NAND gate will be used.

→ for finding "mod value

eg→ up → $Q_3 Q_2 \bar{Q}_1 Q_0 \Rightarrow 1101 \Rightarrow 13$ = first unused state = MOD 13

down → $\bar{Q}_3 Q_2 Q_1 \bar{Q}_0 \Rightarrow 0110 \Rightarrow 6$ = first unused state = MOD 9 (i.e. from 15 to 7).

Q In the MOD 6 ripple counter shown, the o/p of the 2-i/p gate is used to clear the JK flip flops;



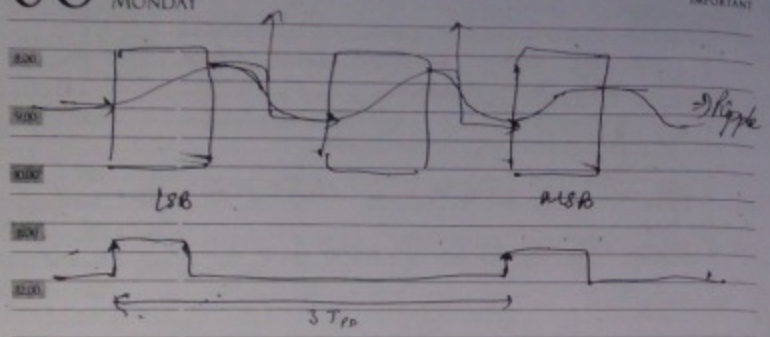
Ans

up counter, MOD 6 – (0 to 5 states)

C	B	A	$\bar{C}$	$\bar{B}$
0	0	0	1	1
0	0	1	1	1
0	1	0	1	0
0	1	1	1	0
1	0	0	0	1
1	0	1	0	1
1	1	0	0	0

→ so either OR or AND as output should be 0. But for OR gate, we getting 1 & 0 for all.

Ans ⇒ OR gate



→ In a ripple counter, the o/p of the flip flop change one bit at a time, starting from the LSB flip flop.

This is b/c of the finite propogation delay of the flip flops.

If T_{PD} ⇒ flip flop propoga" delay

T_{clk} ⇒ time period of clock pulse

n = no. of flip flop stages

then $T_{clk} \geq n \cdot T_{PD}$

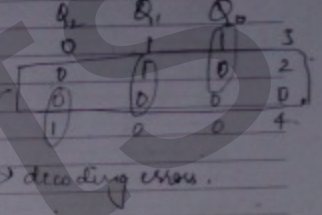
$\Rightarrow f_{clk} \leq \frac{1}{n T_{PD}}$

Decoding Errors

↳ Arises b/c of the finite propoga" delay of the flip flops.

	Q_2	Q_1	Q_0	
3	0	1	1	
4	1	0	0	

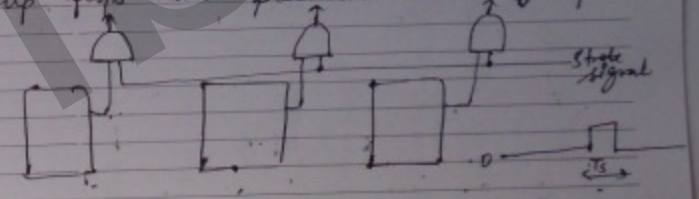
⇒



→ decoding errors.

So it is resolved by the use of strobe signal.

Strobe signal → Samples the o/p when all the flip flops have processed their i/p inputs.



If T_S is the strobe time,

then $T_{clk} \geq n T_{PD} + T_S$

$\Rightarrow f_{clk} \leq \frac{1}{n T_{PD} + T_S}$

DAY 254/112 WEEK 37 SEPTEMBER 2008

WEDNESDAY IMPORTANT

Synchronous Counters

Synchronous up & down counter

→ A common clock triggers all the flip flops simultaneously.

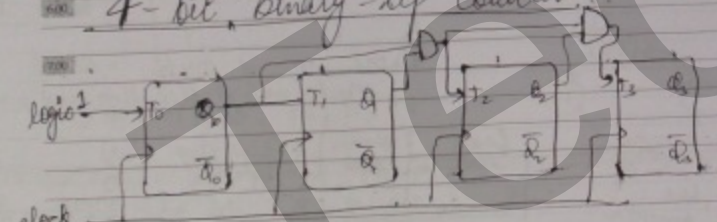
→ The counting sequence is independent of +ve or -ve edge triggering.

→ It consist of a series connection of T flip flop (or other flip flops converted into T flip flop)

→ The T flip flop in the LSB position is always set to the toggle mode ($T=1$).

→ A flip flop in a higher significant position is toggled when the o/p. of all the lower significant flip flops are equal to 1.

4-bit series Binary-up Counter



Clock I/p terminal → Ripple counter
Data I/p → Synchronous counter

2008 SEPTEMBER WEEK 37 DAY 255/110

IMPORTANT THURSDAY 11

State Transition conditions

→ Q_0 toggles on every clock pulse

→ Q_1 toggles when $Q_0 = 1$ & clock is applied.

→ Q_2 toggles when $Q_1 Q_0 = 1$ & clock is applied.

→ Q_3 toggles when $Q_2 Q_1 Q_0 = 1$ & clock is applied.

→ Series refers to the series of AND and not the flip flops.

Clock	Q_3	Q_2	Q_1	Q_0	
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	2
3	0	0	1	1	3
4	0	1	0	0	4
5	0	1	0	1	5
6	0	1	1	0	6
7	0	1	1	1	7
8	1	0	0	0	8
9	1	0	0	1	9
10	1	0	1	0	10
11	1	0	1	1	11
12	1	1	0	0	12
13	1	1	0	1	13
14	1	1	1	0	14
15	1	1	1	1	15
16	0	0	0	0	
17	0	0	0	1	

∴ MOD 16 counter

→ To construct a Down counter, connect $\bar{Q}$ as the i/p to the next stage.

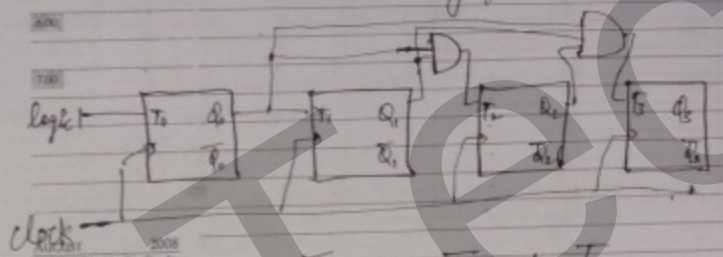
→ In an n bit Series counter

$$T_{clk} \geq T_{PD} + (n-2)\,T_{PD}$$
(flip flop) (AND gate)

$$\Rightarrow f_{clk} \leq \frac{1}{T_{PD} + (n-2)T_{PD}}$$
(FF) (AND)

→ Time period is dependent on the no. of flip flop stages.
This problem is resolved in a parallel counter

4 bit Parallel Binary Up Counter



$$T_{clk} \geq T_{PD} + T_{PD}$$
(FF) (AND)

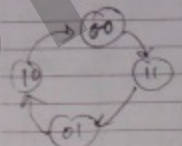
$$\Rightarrow f_{clk} \leq \frac{1}{T_{PD_{FF}} + T_{PD_{AND}}}$$

⇒ Time period is independent on the no. of flip flop stages.

Synchronous Counter Design For Any Random Counting Sequence.

Q Two D flip flops are connected as a synchronous counter that goes through the following $Q_B Q_A$ sequence,
$00 \rightarrow 11 \rightarrow 01 \rightarrow 10 \rightarrow 00 \rightarrow \ldots$
The connections to the i/p D_A & D_B are,

a) $D_A = Q_B$, $D_B = Q_B$
b) $D_A = \bar{Q}_A$, $D_B = \bar{Q}_B$
c) $D_A = (Q_B \bar{Q}_B + \bar{Q}_B Q_B)$ $D_B = Q_A$
d) $D_A = (Q_B Q_B + \bar{Q}_B \bar{Q}_B)$ $D_B = \bar{Q}_B$.



State transition diagram

Total no. of states = N = 4
∴ total no. of flip flop reqd = n = 2

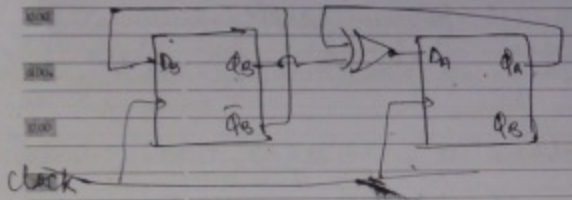
Present state		Next state		Flip flop i/ps	
Q_B	Q_A	Q_B	Q_A	D_B	D_A
0	0	1	1	1	1
0	1	1	0	1	0
1	0	0	0	0	0
1	1	0	1	0	1

15 DAY 259-107 WEEK 38 SEPTEMBER 2008

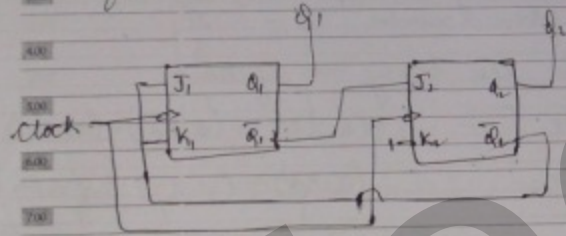
MONDAY IMPORTANT

$D_B = \overline{Q_B}\,\overline{Q_A} + \overline{Q_B}\,Q_A = \overline{Q_B}(\overline{Q_A} + Q_A) = \overline{Q_B}$

$D_A = \overline{Q_B}\,\overline{Q_A} + Q_B Q_A = Q_B \odot Q_A$



Q. What are the counting states ($Q_1 Q_2$) for the counter



(a) 11, 10, 00, 11, 10 ✓

(b) 01, 10, 11, 00, 01

(c) 00, 11, 01, 10, 00

(d) 01, 10, 00, 01, 10

AUGUST 2008

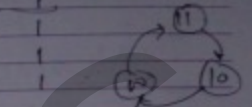
Su	Mo	Tu	We	Th	Fr	Sa
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

2008 SEPTEMBER WEEK 38 DAY Tuesday 16

IMPORTANT TUESDAY

$J_1 = \overline{Q_2}$ $K_1 = Q_2$

Q_1	Q_2	J_1	K_1	J_2	K_2
0	0	1	1	1	1
1	1	0	0	0	1
1	0	1	1	0	1
0	0	1	1	1	1



Ring Counter

→ It is a circular shift register.

→ An n bit counter has n states.

→ If the i/p frequency is f, the o/p freq of each flip flop is f/n.

→ At any instant, only one flip flop is set & all other flip flops are cleared.

Johnson Counter

→ Also known as: twisted ring counter
Switch tail counter

→ An n bit counter has 2n states.

→ If the i/p freq is f, the o/p freq of each flip flop is f/2n

→ The o/p needs to be decoded.

→ A total of 2n decoding gates are required.

SEPTEMBER 2008

Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

Dec 261-105 WEEK 38 — SEPTEMBER 2008

WEDNESDAY — IMPORTANT

→ These gates are either 2-input AND gates or 2-I/p NOR gates.

Number System

→ Radix / Base of a Number System (r)

→ The radix defines the no. of elements in the system.

→ The elements of a no. system are derived from decimal system & the english alphabet in such a way that,

→ $r < 10 \Rightarrow$ the first r decimal digits

→ $r = 10 \Rightarrow$ the decimal system itself

→ $r > 10 \Rightarrow$ digits 0 to 9 & the first $(r-10)$ english letters.

RADIX POINT

→ In the decimal no 110.40 → known as decimal point or Radix point (in general)

→ ~~In general the~~

→ $r = 2 \Rightarrow$ binary point

$r = 8$ octal

$r = 16$ hexadecimal

→ $2006 = 2000 + 00 + 0 + 6$

$= 2 \times 10^3 + 2 \times 10^2 + 0 \times 10^1 + 6 \times 10^0$

AUGUST 2008

Su	Mo	Tu	We	Th	Fr	Sa
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

2008 SEPTEMBER — Week 38 Day 262-104

IMPORTANT — THURSDAY — 18

∴ In general, a decimal no $a_n - - - a_2 a_1 a_0 . a_{-1} - - - a_{-m}$

is a shorthand of

$(a_n \times 10^n) + - - - (a_2 \times 10^2) + (a_1 \times 10^1) + (a_0 \times 10^0) + (a_{-1} \times 10^{-1}) + - - (a_{-m} \times 10^{-m})$

∴ in general for any radix r

$(a_n \times r^n) + - - - (a_2 \times r^2) + (a_1 \times r^1) + (a_0 \times r^0) + (a_{-1} \times r^{-1}) + - - (a_{-m} \times r^{-m})$

eg. $(1101)_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

$= 8 + 4 + 0 + 1$

$= 13$

→ The unique property of the above expression is that it gives the decimal equivalent of the given number.

→ In the binary no. system,

$(a_n \times 2^n) + - - - - + (a_2 \times 2^2) + (a_1 \times 2^1) + (a_0 \times 2^0)$

$(a_n \times 2^{-n}) + - - - - (a_2 \times \frac{1}{4}) + (a_1 \times \frac{1}{2}) + (a_0 \times 1)$

(weights in powers of 2)

→ $a_n - - - a_3 a_2 a_1 a_0$ are either 0 or 1.

∴ decimal equivalent of a binary no. is obtained by simply adding the weights of all those bit positions where a 1 is present

SEPTEMBER 2008

Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

19 Day 263-103 Week 38 SEPTEMBER 2008

FRIDAY

IMPORTANT

Conversion of a decimal no. to radix r

→ Seperate the integer & fraction parts.

Conversion of integer part:

→ Divide the integer & the subsequent quotients by r, until the quotient becomes 0.

→ Write the remainders in the reverse order.

eg ① $(53)_{10} = (\ ?\ )_2$

$= (110101)_2$

2	53	
2	26	1
	13	0
	6	1
	3	0
	1	1
	0	1

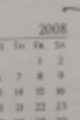
② $(422)_{10} = (\ ?\ )_6$

$= (1542)_6$

6	422	
	70	2
	11	4
	1	5
	0	1

Conversion of fractional part.

→ Multiply the fraction by r until the fractional part becomes 0 or the desired level of accuracy is achieved.



2008 SEPTEMBER Week 38 Day 264-102

IMPORTANT

SATURDAY 20

→ Write the integers in the same order.

eg ① $(0.625)_{10} = (\ ?\ )_4$

$= (0.22)_4$

$0.625 \times 4 = 2.500$

$0.500 \times 4 = 2.000$

② $(0.840)_{10} = (\ ?\ )_2 = (0.110101)_2$

$0.840 \times 2 = 1.680$

$0.680 \times 2 = 1.360$

$0.360 \times 2 = 0.720$

$.720 \times 2 = 1.440$

$.440 \times 2 = 0.880$

Octal & Hexadecimal Conversion

→ To convert from octal to binary, write the 3-bit binary equivalent of each octal digit.

eg $(76.21)_8 = (111\ 110.010\ 001)_2$

Day 265-101 SUNDAY 21

→ To convert from hexadecimal to binary write the 4-bit binary equivalent of each hexadecimal digit

eg $(D6.A2)_{16} = (1101\ 0110.1010\ 0010)_2$

22 MONDAY — DAY 266 (100) WEEK 39 — SEPTEMBER 2008 — IMPORTANT

→ To convert from binary to octal, start from the binary point & form groups of 3 bits. Then write the octal equivalent of each group.

eg $(011\,010.\,010\,110) = (32.26)_8$

→ To convert from binary to hexadecimal, start from the binary point & form group of 4 bits. Then write the hexadecimal equivalent of each group.

eg $(0110\,0110.\,1010\,0010)_2 = (66.A2)_{16}$

Q → The no of 1's in the binary representation of $(3 \times 4096 + 15 \times 256 + 5 \times 16 + 3)$ are

Sol $(3 \times 16^3 + 15 \times 16^2 + 5 \times 16^1 + 3 \times 16^0)$

$= (3F53)_{16}$

$= (0011\ 1111\ 0101\ 0011)_2$

$= 10 \rightarrow 1$'s

Compliments

→ In any system, there are 2 types of compliments
1. Diminished Radix Compliment or $(r-1)$'s complement

AUGUST 2008

Su	Mo	Tu	We	Th	Fr	Sa
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

2008 SEPTEMBER — WEEK 39 DAY 267 (099) — IMPORTANT — TUESDAY 23

2. Radix compliment / r's compliment

→ The r's complement can be obtained by adding 1 to the $(r-1)$'s complement.

$(r-1)$'s compliment

→ The $(r-1)$'s compliment of a no. N, of radix r & having n no. of digits is given by $r^n - 1 - N$

→ For the decimal no system (9's complement) subtract the individual digits from 9

eg $N = 6793$

$\quad\quad 3206$

→ For the binary no. system (1's complement) change all 0's to 1's & 1's to 0's

eg $N = 1010$ ↓ 1's complement 0101

$\quad\quad 0101$

$1111 - 1010 = 0101$

⇒ Hly $(r-1)$ complement in any other radix is obtained by subtracting the individual digits from $(r-1)$.

SEPTEMBER 2008

Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

24 Day 268·098 Week 39 SEPTEMBER 2008
WEDNESDAY IMPORTANT

r's complement

The r's complement of a no. N, of radix r & having n no. of digits is given by:

$$r^n - N \quad , \text{when } N \neq 0$$
$$0 \quad , \text{when } N = 0$$

→ For decimal no system (10's complement)

→ Leave all the least significant 0's unchanged.
→ Subtract the first ~~least~~ significant, non zero digit, from 10.
→ Subtract all higher significant digits from 9.

eg $N = 697300$

9 9 9 10 (10^6)
6 9 7 3 0 0
3 0 2 7 0 0

1000000
697300
302700

→ For binary no. system (2's complement).

① Leave all the least significant 0's unchanged.
② Leave the first least " 1 unchanged
③ Change all higher significant 0's to 1's and 1's to 0's.

eg $N = 101100$

2^6 = 64
1000000
101100
010100

0 1 0 1 0 0

AUGUST 2008

Su	Mo	Tu	We	Th	Fr	Sa
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

2008 SEPTEMBER Week 39 Day 269·097 25
IMPORTANT THURSDAY

Signed Binary Numbers

System of unsigned numbers
5
→ Always (& only) +ve no.
→ Only exception is 0.

System of Signed No
+5, -5
Both +ve & -ve no.

Sign Bit

→ In signed binary no the MSB bit is assigned as the sign bit in such a way that,
→ Sign bit = 0 ⇒ +ve no.
→ Sign bit = 1 ⇒ -ve no.

Note: The above allocation remains the same irrespective of the signed binary representation used.

Signed Binary Representation

+ve no.
Represented in
→ Signed magnitude system

-ve no.
→ Signed mag. sys/ signed complement sys.
→ Signed 1's complement sys
→ Signed 2's complement sys

26 FRIDAY — Day 270-096 Week 39 — 2008 September — Important

Signed Magnitude System (SMS)

→ In the SMS, we write the binary equivalent of the decimal value & then add the sign bit at the MSB position.

eg +5 : 0 101 ; −5 : 1 101

Signed compliment system (SCS)

y —(S.1's CS)→ ? ←(1's comp)— X.

eg (= −5)

→ find the corresponding signed magnitude positive no.

→ +5 → 0101 → (1's compliment) → 1010 = Ans

Why to obtain the S.2's CS, we find the 2's compliment of X.

Q A no. is expressed in Signed 2's compliment as 10011. Its decimal equivalent value is

a) 19 b) 13 c) −19 d) −13

August 2008

27 SATURDAY — 2008 September — Week 39 Day 271-095 — Important

Ans 10011 —(2's comp)→ 01101 ⇒ +13
∴ no. is −13.

→ for finding SCS of +ve no. in both 1's & 2's compliment → same as SMS.

→ A +ve no. is represented in the SCS exactly the same way as it represented in the SMS.

→ To obtain the decimal value of a no. in SCS we first inspect the sign bit of the no.

→ Sign bit = 0 ⇒ the no. is +ve & automatically in the SMS.

sign bit = 1 ⇒ no is −ve
→ find the compliment once again
→ This gives the corresponding signed magnitude +ve no.
→ change the +ve sign to −ve sign

28 SUNDAY

Eg 2 no. X & Y are represented in the S.1's CS as 10100 & 00101. Find decimal value of X & Y

Ans X = 10100 (−ve) —(1's comp)→ 01011 ⇒ +11

Y = 00101 ⇒ +5. (+ve)

September 2008

29 Day — Oct 273-093 Week 40 — SEPTEMBER 2008 — MONDAY — IMPORTANT

Addition of Signed Binary Numbers.

Steps:

1. The two operands are represented in the SMS.
2. The ve operands are represented in the S 2's C.S.
3. The 2 operands, including the sign bit, are added.
4. A carry, if generated, is disguarded.
5. The result, if +ve (sign bit = 0) is automatically in the SMS.
6. The result, if -ve (sign bit = 1) is automatically in the S 2's C.S.

eg

+11	⇒	0 1 0 1 1
+4	⇒	0 0 1 0 0
		0 1 1 1 1 ⇒ +15

+11	⇒	0 1 0 1 1
−4	⇒	1 1 1 0 0
		(1) 0 0 1 1 1 ⇒ +7

(1) disguarded

−11	⇒	1 0 1 0 1
−4	⇒	1 1 1 0 0
		(1) 1 0 0 0 1

(1) disguarded

↓ −ve no. so find 2's complement

0 1 1 1 1 ⇒ +15

dec = −15

30 Day — 2008 SEPTEMBER — Week 40 Oct 274-092 — TUESDAY — IMPORTANT

−11	⇒	1 0 1 0 1
+4	⇒	0 0 1 0 0
		1 1 0 0 1

↓ 2's complement

0 0 1 1 1 ⇒ +7 dec = −7

Ranges of Different Number Representation

→ unsigned : 0 to $2^n - 1$

→ Signed mag. sys : $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$

→ Signed 1's C.S. : $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$

↳ not implemented in real world cmt as 0 has 2 representations, i.e. +0 & −0, i.e. why 2's compliment is implemented.

→ Signed 2's C.S : $-(2^{n-1})$ to $+(2^{n-1}-1)$

[NOTE]: The SMS & 1's C.S. have 2 different representations for 0, i.e. +0 & −0.

x	y	z	2's C.S.	1's C.S.	SMS
0	0	0	+0	+0	+0
0	0	1	+1		
0	1	0	+2		
0	1	1	+3		
1	0	0	−4		
1	0	1	−3		
1	1	0	−2		
1	1	1	−1	−0	−0

Compliment of a fraction no.

→ (r-1)'s compliment

→ Subtract the individual digit from r-1.

eg Find 9's compliment of $(6793.214)_{10}$

$$N = 6793.214$$
$$3206.785$$

→ r's compliment
find the (r-1)'s compliment & then add 1 to the rightmost digit.

eg Find 10's compliment of $(6793.214)_{10}$

$$= 3206.785$$
$$+1$$
$$3206.786$$

Bit Extention in S.C.S.

→ To extend the no. of bits in S.C.S simply re-write the sign bit repeatedly.

eg $1101 = 1111101$

$0101 = 0000101$

Q 4 bit 2's compliment representation of a decimal no. is 1000. The no. is:
(a) +8 (b) 0 (c) −7 (d) −8

1000 —2's→ 1000 = −0 → +0 ✗

Now using bit extension, we write

11000 —2's→ 01000 → +8
ans = −8

Overflow

→ Overflow is said to occur when the sum of 2 n-bit binary numbers crosses the range of n-bits

→ Range of 2's C.S. = $-(2^{n-1})$ to $+(2^{n-1}-1)$

eg for n = 4,
range = −8 to +7

∴ overflow will occur when the sum is <−8 or >+7

→ Overflow may occur only when both operands are +ve or both operands are −ve.

eg +5 0101
+4 0100
1001

∴ overflow

−5 1011
−4 1100
(1)0111 → 7
signed bit

Underflow and overflow

So overflow will occur when $\bar{x}\bar{y}z + xy\bar{z}$.

(Cin)	(Cout)	(Cin)	(Cout)
0 1 0 0		0 0 0	
0 1 0 1		1 0 1 1	
0 1 0 0		1 1 0 0	
1 0 0 1		1 0 1 1 1	

So overflow when = $C_n \oplus C_{n-1}$

⇒ The cond. of overflow may be detected in the following ways:

→ $\bar{x}\bar{y}z + xy\bar{z}$

→ $C_n \oplus C_{n-1}$

Boolean Algebra

For the set of element = $B = \{0, 1\}$, AND, OR & NOT operations are defined as

AND operation

A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

OR operation

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

→ $A \cdot A = A$
$A \cdot \bar{A} = 0$
$A \cdot 0 = 0$
$A \cdot 1 = A$

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

NOT operation

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

Operator Precedence

BRACKET
NOT
AND
OR

Basic Theorem of Boolean Algebra

→ Involution
$\bar{\bar{A}} = A$

→ De-Morgan's
$\overline{A \cdot B \cdot C} = \bar{A} + \bar{B} + \bar{C}$
$\overline{A + B + C} = \bar{A} \cdot \bar{B} \cdot \bar{C}$

→ Absorption
$A + AB = A$
$A(1 + B) = A$

→ Associativity
$A(BC) = (AB)C$
$A + (B + C) = (A + B) + C$

Distribution Theorem

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

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

06 DAY 280-086 WEEK 41 OCTOBER 2008

MONDAY

IMPORTANT

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

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

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

Q $AB + (\bar{A}+\bar{B})CD$ Simplify.

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

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

$= B + CD + \bar{B}CD$

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

$= B + CD + CD$

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

Q The boolean $\bar{x}\bar{y} + xy + \bar{x}y$ is equivalent to

Sol $\bar{x}(\bar{y}+y) + xy$

$\bar{x} + xy$

$\bar{x} + y$ Ans

Q $AB + BC + \bar{A}C$

$= AB + BC.1 + \bar{A}C$

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

$= AB + BCA + BC\bar{A} + \bar{A}C$

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

$= AB + \bar{A}C$

2008 OCTOBER WEEK 41 DAY 281-085 07

IMPORTANT

TUESDAY

Redundancy Theorem / Consensus Theorem

Applicable when,
→ 3 variables are present (eg A,B,C)
→ each variable is used 2 times
→ only 1 literal is complemented (eg $\bar{A}$)

Q Thus the term which contains neither A nor $\bar{A}$ becomes the redundant term.
This term is eliminated.

Illy $\bar{A}B + BC + AC = \bar{A}B + AC$

$A\bar{B} + BC + AC = A\bar{B} + BC$

$AB + \bar{B}C + AC = AB + \bar{B}C$

$AB + B\bar{C} + AC = AC + B\bar{C}$

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

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

Q This theorem is also applicable for the POS term

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

Q The 3rd point of the theorem can be manipulated as 'Each literal is complemented except one'.

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

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

08 Day 282-084 Week 41 OCTOBER 2008
WEDNESDAY IMPORTANT

Transposition Theorem

$(A+B)(A+C) = A\cdot A + A\cdot C + A\cdot B + BC$
$= A + AC + AB + BC$
$= A(1 + C + B) + BC$
$= A + BC$

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

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

Q using trick

$\bar{A}BC + A\bar{B}C + AB\bar{z} + ABC$
$A + A + A = A$
$\Rightarrow A = A + A + A$

$\bar{A}BC + A\bar{B}C + AB\bar{z} + ABC + ABC + ABC$

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

SEPTEMBER 2008

Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

2008 OCTOBER Week 41 Day 283-083
IMPORTANT THURSDAY 09

Q If $x=1$ in the logic eqn:
$([x+z\{\bar{y}+[\bar{z}+x\bar{y})\}])\{\bar{z}+\bar{z}[x+y)\}=1$,
then
(A) $y=z$ (B) $y=\bar{z}$ (C) $z=1$ (D) $z=0$

$\Rightarrow (\quad)\cdot(\quad) = 1$

$\Rightarrow \bar{x} + \bar{z}(x+y) = 1$
$1 \quad 0 \quad 1$
$\Rightarrow \bar{z}(x+y) = 1$
$\Rightarrow \bar{z} = 1$
$\Rightarrow z = 0$

Duality Principle / Dual

→ used to convert true logic into -ve logic & vice versa.

Procedure:
→ interchange the AND & OR operations
→ " " 0's & 1's.

Eg (1) $A\bar{B}C + \bar{A}BC + AB\bar{C}$
↓ dual
$(A+\bar{B}+C)(\bar{A}+B+C)(A+B+\bar{C})$

(2) $A+1=1$ —dual→ $A\cdot 0 = 0$

OCTOBER 2008

Su	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

10 Day 284-082 Week 41 OCTOBER 2008

FRIDAY

IMPORTANT

→ Finding the dual of an expression 2 times gives back the original expression.

→ Self Dual Expressions

→ finding the dual only once gives back the original expression.

⇒ → With 'n' variables a total of $2^{2^{n-1}}$ self dual expressions can be formulated.

Inspect the following exp. for self duality.

(i) $AB + BC + AC$ SOP

↓ dual

$(A+B)(B+C)(A+C)$ POS ← convert to SOP for compare

$= (B+AC)(A+C)$

$= AB + BC + AC$

∴ Self dual

(ii) $\bar{A}B + BC + A\bar{C}$

↓ dual

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

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

$= BA + B\bar{C}$

∴ Not self dual

SEPTEMBER 2008

Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

2008 OCTOBER Week 41 Day 285-081

IMPORTANT

SATURDAY 11

Compliment

→ Obtain using the De Morgans theorem.

→ Procedure:

→ Interchange the AND & OR operations

→ " 0's & 1's.

→ complement each literal

Eg → (i) $A\bar{B}C + AB\bar{C} + \bar{A}BC$

↓ complement

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

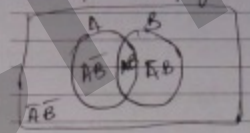
(ii) $A + 1 = 1$ → complement → $\bar{A} \cdot 0 = 0$

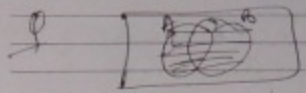
Venn Diagram

Graphical approach of boolean representation & simplification.

Day 286-080 SUNDAY 12

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





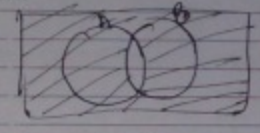
$= A\bar{B} + AB + \bar{A}B$

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

$= A + B$

OCTOBER 2008

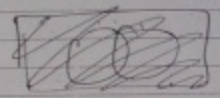
Su	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	



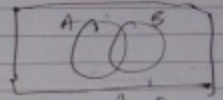
$f = \overline{AB}$

$g = \overline{A}$

Logic operation performed is EXNOR

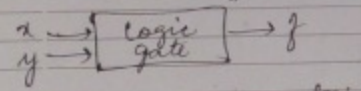


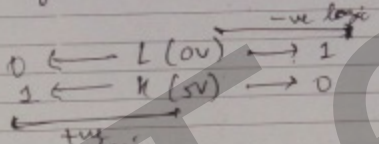
$g = 1$



$g = 0$

Positive & Negative Logic





→ It has nothing to do with polarity of the signals.

	+ve		-ve		+ve	-ve
0	0V	6V	-3	-6	0	
1	5V	1V	-8	-1	-6	

→ The I/p & O/p of a logic gate are specified interms of a low level signal (L) and a high level signal (H).

→ A binary digital system is defined for 2 logic values, logic 0 & logic 1.

→ ∴ 2 assignments are possible.

L ⇒ logic 0 } +ve logic system
H ⇒ logic 1 }

L ⇒ logic 1 } -ve logic system
H ⇒ logic 0 }

Note: +ve & -ve logic does not refer to the polarity of the signals.

x	y	f
L	L	L
L	H	L
H	L	L
H	H	H

Assuming +ve logic

xy	f
0 0	0
0 1	0
1 0	0
1 1	1

↓ AND gate

Assuming -ve logic

a	b	f
1	1	1
1	0	1
0	1	1
0	0	0

↓ OR gate

⇒ +ve logic AND gate ≡ -ve logic OR gate