ElCoM
ELECTRICAL COMPUTER MECHATRONICS

تقدم لجنة ElCoM الاكاديمية

دفتر فاينل لمادة:

منطق رقمي

من شرح:

د.سامر خصاونة

جزيل الشكر للطالب:

تسنيم بركات

Numbering system:

1) A way of representing quantities

2) every N.S has Radix (R) (base)

determine total number of numbers available ⟷ naming

R=10 → عشري   zero → {0,1,2,3,------9} ← r-1

10 items number symbols

R = 5 → خماسي

R = 2 → {0,1}

3) There are so many numbering systems

R = 12 ⇒ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B}

A ↓ 10, B ↘ 11

R = 16 السادس عشر / Hexa decimal

{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}

A = 10, B = 11, C = 12, D = 13, E = 14, F = 15

(most common)

التركيز على ✫ R=2, R=8, R=10, R=16

R=2 ↓ binary / ثنائي {0,1}

R=8 ↓ ثماني octal {0,----,7}

R=10 {0,---,9}

R=16 {0,----,F}

Similarity: all numbering system represent quantity but in different shape.

R = 2 → 1111

R = 8 → 17

R = 10 → 15

R = 16 → F

$(15)_{10} = (17)_{\text{octal } 8} = (1111)_2 = (F)_H$

✫ $(200)_{16} \neq (200)_{10}$

Scanned with CamScanner

Smallest R:

$R = 0 \Rightarrow \{ \}$ ✗

$R = 1 \Rightarrow \{0\}$ ✗

$R = 2 \Rightarrow \{0, 1\}$ ✓ → smallest Radix (R)

1 11 10 01 1011

$R = 3$ ✓

R is integer ≥ 2
and positive

Number stracture

$\_ \ \_ \ \_ \ . \ \_ \ \_$

∈ domain $\in \{0, ---, r-1\}$

$1234 \in r = 5$

$1367 \notin r = 7$

$(\overset{}{6}\ \overset{}{6}\ \overset{r^0}{6}\ .\ 6\ 6)_7$

$7^2 \quad 7^1 \quad 7^0 \quad 7^{-1} \quad 7^{-2}$

$$\hookrightarrow \boxed{6 \times 7^2 + 6 \times 7^1 + 6 \times 7^0} + \boxed{\frac{6}{7} + \frac{6}{49}}$$

$(\overset{r^2}{3}49.9)_{10}$

$\underset{10^2}{300} + \underset{10^1}{40} + \underset{10^0}{9} + \frac{9}{10} \rightarrow 10^{-1}$

Scanned with CamScanner

(binary) 2 → B   8 → Q (octal)

(decimal) 10 → d   16 → H (Hexa decimal)

❁ Conversion between numbering systems

1) from any N.S → decimal

→ multiply the number by it's weight

Ex: convert $(1011)_2$ to decimal

$r^3 \quad r^2 \quad r^1 \quad r^0$

$2^3 \quad 2^2 \quad 2^1 \quad 2^0$

$1\times2^3+0+1\times2^1+1\times2^0$

$8+0+2+1=11_{decimal}$

Ex: convert $(234)_6$ → decimal

$6^2 \quad 6^1 \quad 6^0$

$2\times6^2+3\times6+4\times6^0 = 72+18+4=(94)_{10}$

Ex: convert $(AB.AB)_{16}$ to decimal

A=10 , B=11

$16^1 \quad 16^0 \quad 16^{-1} \quad 16^{-2}$

$10\times16^1+11\times16^0+\frac{10}{16}+\frac{11}{16^2}=(171.668)_{10}$

LSB: Least significant bit

MSB: Most significant bit

$(\underset{MSB}{\underset{\downarrow}{-}}\ -\ -\ -\ -\ -\ \underset{LSB}{\underset{\downarrow}{-}})_2$

digit → all systems   bit → binary

Scanned with CamScanner

② From decimal → Any system → Integer number: Repeated devision
→ fraction: Repeated multiplication

d → (2, 3, 4, 5, ..., 8)
binary a

$(31.5)_{10} \rightarrow (\quad)_3$

* integer: divide by destination Radix (r) until you reach zero, take reminder at each step
* float: multiply by destination Radix (r), take out integer part, continue until you reach zero or repetition occurs

Ex: $(135.75)_{10} \rightarrow (\quad)_2$ R = 2 ∈ {0, 1}

① 135 remainer

2	135	1	* LSB
2	67	1	
2	33	1	
2	16	0	
2	8	0	
2	4	0	
2	2	0	
2	1	1	
	0		* MSB

stop

② 0.75

MSB

$0.75 \times 2 = 1.5$ 1

$0.5 \times 2 = 1.0$ 1 ← LSB

0.0 Stop

$0.75 = 0.11$

$135 = 10000111$
$2^7 \quad 2^2\, 2^1\, 2^0$

verification

$135 = 1 \times 2^7 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
$= 128 + 4 + 2 + 1$
$= 135$

$\therefore (135.75)_{10} = (10000111.11)_2$

Scanned with CamScanner

Ex: $(74.2)_{10} \rightarrow (\quad)_2$

2	74	ϕ	LSB
2	37	1	
2	18	0	
2	9	1	
2	4	0	
2	2	0	
2	1	1	MSB
	0		

$0.2 \times 2 = 0.4$ MSB 0

$0.4 \times 2 = 0.8$ 0

$0.8 \times 2 = 1.6$ 1

$0.6 \times 2 = 1.2$ 1

$0.2 \times 2 = 0.4$ stop LSB

$\therefore (74.2)_{10} \approx (1001010.0011\text{--})_2$

Ex: $(187)_{10} \Rightarrow (\quad)_{16}$

$\boxed{B = 11}$

16	187	11	→ LSB
16	11	11	→ MSB
	0		

$\therefore (187)_{10} = (BB)_{16}$

Scanned with CamScanner

Ex: $(74.2)_{10} \longrightarrow (\quad\quad)_8$

$$\begin{array}{c|cc}
8 & 74 & 2 \\
8 & 9 & 1 \\
8 & 1 & 1 \\
 & 0 &
\end{array}$$

2 → LSB, 1 → MSB; $\in \{0, \ldots, 7\}$

$74 = 112$

$0.2 \times 8 = 1.\underline{\underline{6}}$ $\quad 1 \to$ MSB

$0.6 \times 8 = 4.\underline{\underline{8}}$ $\quad 4$

$0.8 \times 8 = 6.\underline{\underline{4}}$ $\quad 6$

$0.4 \times 8 - 3.\underline{\underline{2}}$ $\quad 3 \to$ LSB

0.2×8

$0.2 = 0.1463$ ---- $\therefore (74.2)_{10} \approx (112.1463\text{----})_8$

Ex: $(201.5)_{10} \longrightarrow (\quad\quad)_{16}$

$$\begin{array}{c|cc}
16 & 201 & 9 \\
16 & C\ (12) & C \to 12 \\
\frac{C^{12}}{16} & \emptyset &
\end{array}$$

$201 = C9$

$0.5_{10} \Rightarrow (\quad\quad)_{16}$

$0.5 \times 16 = 8.0$ $\quad 8$

$0.0 \to$ stop

$(0.5)_{16} = (0.8)_{16}$

10^{-1} $\quad 16^{-1}$

$\therefore (201.5)_{10} = (C9.8)_{16}$

Scanned with CamScanner

Subject   Date   No.

③ fast decimal to binary conversion

* try to decompose the integer number into its components

Ex: $(192)_{10} \xrightarrow{fast} (\quad)_2$

2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
/	/	1	1	0	0	0	0	0	0
256	128	64	32	16	8	4	2	1

$(192)_{10} = (11000000)_2$

Ex: $(31)_{10} \xrightarrow{fast} (\quad)_2$

$$\begin{array}{ccccc} 1 & 1 & 1 & 1 & 1 \\ 16 & \underbrace{8 \quad 4 \quad 2 \quad 1}_{\text{max } 15} \end{array}$$

$(31)_{10} = (11111)_2$

④ Octal ↔ binary

represent each octal number as 3 binary bits or vice versa

$(\underset{x}{\text{—}})_8 \rightarrow (\underset{x}{\text{- - -}})_2$

octah	binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Ex: $(765)_8 \rightarrow (111110101)_2$

$7 \downarrow 111 \quad 6 \downarrow 110 \quad 5 \downarrow 101$

Ex: $(50.12)_8 \rightarrow (101000.001010)_2$

$5 \downarrow 101 \quad 0 \downarrow 000 \quad 1 \downarrow 001 \quad 2 \downarrow 010$

$(\underset{\downarrow}{\underline{110}}\,\underset{\downarrow}{\underline{111}}\,\underset{\downarrow}{\underline{010}}.\underset{\downarrow}{\underline{110}}\,\underset{\downarrow}{\underline{100}})_{binary}$

$(6\ \ 7\ \ 2\ .\ 6\ \ 4)_{octab}$

Ex: $(\underset{\downarrow}{\underline{001}}\ \underset{\downarrow}{011}.\underset{\downarrow}{\underline{010}})_b$

$(1\ \ 3\ .\ 2)_{octan}$

⑤ Hexa decimal ↔ binary

each 4 bits in binary is replaced by one Hexa numbe

$(f21.7)_H \rightarrow (\underbrace{1111}_{F}\ \underbrace{0010}_{2}\ \underbrace{0001}_{1}.\underbrace{0111}_{7})$ binary

$(\underset{\downarrow}{\underline{0010}}\ \underset{\downarrow}{\underline{1011}}.\underset{\downarrow}{\underline{1000}})_2$

$(2\ \ B\ .\ 8)_H$ 10-15 A → f

*Binary arithmetic (+, -, *)

numbers

unsigned (magnitude only)

no sign, not positive

not negative

Ex: 27

32	16	8	4	2	1
X	1	1	0	1	1

range: $(0, 2^n - 1)$

number of values = 2^n

signed

sign	magnitude

-27 → ?

+27 → ?

① 1's complement

② 2's complement

③ signed Magnitude (SM)

① sign-magnitude (SM)

For positive $\overset{S}{\boxed{0}}\boxed{\text{get magnitude}}$

For negative $\overset{S}{\boxed{1}}\overset{M}{\boxed{\text{get magnitude}}}$

* range: $(-(2^{n-1}-1), 2^{n-1}-1)$

Ex: $-13 \rightarrow \boxed{1 \mid 1101}$

* number of values: 2^n

② 1's complement (Diminished radix complement)

For positive $\overset{S}{\boxed{0}}\overset{M}{\boxed{\begin{array}{l}\text{convert to binary}\\ \text{copy paste in M}\end{array}}}$

* positive numbers are never flipped

For negative $\boxed{1 \mid \quad\quad}$

1) convert to binary
2) flip all bits $0110 \rightarrow 1001$
3) store flipped binary value

③ 2's complement (Radix complement)

For positive $\overset{S}{\boxed{0}}\overset{M}{\boxed{\begin{array}{l}\text{convert to binary}\\ \text{paste binary value}\end{array}}}$

For negative $\overset{S}{\boxed{1}}\overset{M}{\boxed{\text{-----}}}$

① get binary value
② flip all binary bits
③ Add (one) to binary bits that are flipped
④ store result in M

Scanned with CamScanner

Ex: represent 58 in binary (unsigned)

32	16	8	4	2	1
1	1	1	0	1	0

$\xrightarrow{\text{copy-paste}}$ $\overset{M}{\boxed{111010}}$

*58_{10} needs at least 6 bits, so 5 bits can't represent 58 and 7 bits can represent 58 → 0111010

not signed

Ex: represent $(+35)_{10}$ → in binary

$\boxed{\overset{S}{0} \mid \overset{M}{100011}}$

32	16	8	4	2	1
1	0	0	0	1	1

$\therefore +35 = (0100011)_{1's}$

$= (0100011)_{2's}$

$= (0100011)_{s/M}$

Ex: represent $(-35)_{10} \to (\quad\quad)_{binary}$

① S/M $\boxed{\overset{S}{1} \mid \overset{M}{100011}}$

② 1's $\boxed{\overset{S}{1} \mid 011100}$

③ 2's $\boxed{\overset{S}{1} \mid \overset{M}{011101}}$

Scanned with CamScanner

Ex: what is the decimal value of $(101011)_2$

① if unsigned => $\underset{32\ 16\ 8\ 4\ 2\ 1}{\overset{M}{\boxed{101011}}} = 43$

if signed

② S/M => $\boxed{\underset{S}{1} \mid \underset{M}{01011}} = -11$

③ 1's => $\boxed{\underset{(-)}{1} \mid \underset{10100}{01011}} = -20$

④ 2's => $\boxed{\underset{(-)}{1} \mid 01011}$

$$\begin{array}{r} 10100\,+ \\ 1 \\ \hline 10101 \end{array} = -21$$

Ex: If $n=6$, $r=2$ what is the range for unsigned numbers?

range: $(0, 2^n-1) = (0, 2^6-1) = (\underset{\text{minimum value}}{0}, \underset{\text{maximum value}}{63})$

Ex: If $n=8$ represent 520? (unsigned)

range: $(0, 2^n-1) = (0, 255)$

→ 520 can't be represented (out of range)

Ex: represent +5 when $n=8$, $r=2$ using SM

$(\underbrace{0}_{\text{sign}}\ \underbrace{0\ 0\ 0\ 0\ 1\ 0\ 1}_{\text{magnitude}})_{SM}$

Scanned with CamScanner

Ex: what is the following number?

1) $(\underline{0}1010)_{1's}$ $8+2 = 10$

+ve 8 4 2 1

2) $(\underline{1}1010)_{1's}$

-ve

convert to positive number (00101)

16 8 4 2 1

00101 = 1+4 = 5

⇒ $(11010)_{1's} = -5$

Ex: find (-15) if n = 6 in 1's complement

+15   0 0 1 1 1 1

16 8 4 2 1

-15   1 1 0 0 0 0

Ex: what is the following number in decimal?

1) $(\underline{0}1101)_{2's} \rightarrow 1+4+8 = +13$

+ve 8 4 2 1

2) $(\underline{1}0110)_{2's}$ ① find 1's complement

-ve   ② Add 1 to the result

$$\begin{array}{r} 10110 \rightarrow 01001 \\ 1+ \\ \hline 01010 \end{array}$$

⇒ 01010 = +10

8 4 2 1

$(10110)_{2's} = -10$

Scanned with CamScanner

Ex: find (-9) in 2's complement if n = 7 ?

+9 = 0001001 → (1110110)

$$\begin{array}{rl} -9 = & 1110110 \\ & \underline{\quad\quad\;\; 1+} \\ & (1110111)_{2's} \end{array}$$

Ex: find (-15) in 2's complement if n = 5 ?

+15 = 01111 → (10000)

$$\begin{array}{rl} -15 = & 10000 \\ & \underline{\quad\;\; 1+} \\ & (10001)_{2's} \end{array}$$

Ex: represent -42.75 using

SIM $\boxed{1 \mid 101010.11}$
(-)

1's $\boxed{1 \mid 010101.00}$
(-)

2's 1 01010100
1 +

= $\boxed{1 \mid 010101.01}$

M = 42.75

101010 ⇒ 42

32 16 8 4 2 1

0.11 ⇒ 0.75

M = 42.75 = 101010.11

Scanned with CamScanner

Ex: convert $(1111)_{1's} \rightarrow (\quad)_{2s}$

1's $\boxed{1 | 111}$ → 4 digits (4 bits)

1 +

2's $\boxed{1 | 1000}$ → 5 bits

Represent Zero as signed number in binary

S/M: S, M $\boxed{0 | 000}$ + zero; S, M $\boxed{1 | 000}$ - zero

∴ in S/M there are 2 zero's

1's: S, M $\boxed{0 | 0000}$ + zero; S, M $\boxed{1 | 1111}$ - zero

∴ 1's there are 2 zeros

2's: S, M $\boxed{0 | 0000}$ + zero; S, M $\boxed{ | }$ No negative zero

∴ only one zero in 2's

Scanned with CamScanner

system	Range	total numbers represented
unsigned	$[0, 2^n - 1]$	2^n
S/M	$[-(2^{n-1}-1), (2^{n-1}-1)]$	2^n
1's	$[-(2^{n-1}-1), (2^{n-1}-1)]$	2^n
2's	$[-(2^{n-1}), (2^{n-1}-1)]$	2^n

Ex: How many bits are required to represent $(-121)_{decimal}$ in 2's complement?

8 bits $[-128, 127] \rightarrow$ at least 8 bits

* Extension: enlarge the number size without affecting the sign or magnitude

① unsigned

$\underline{1}\,\underline{0}\,\underline{1} \Rightarrow \underline{0}\,\underline{0}\,\underline{0}\,\underline{1}\,\underline{0}\,\underline{1}$

3bits 6 bits (Zero extension)

(extend by adding zeros)

② S/M

$\boxed{1 \mid 101} \Rightarrow \boxed{1 \mid 0000101}$ (zero extinsion)

③ 1's , 2's $\boxed{0 \mid 101} \Rightarrow \boxed{0 \mid 000\ 101}$

(sign extinsion)

$\boxed{1 \mid 101} \Rightarrow \boxed{1 \mid 111\ 101}$

(extend the sign bit)

Scanned with CamScanner

1's 2's $\Rightarrow r=2$

7's 8's $\Rightarrow r=8$

14's 15's $\Rightarrow r=15$

R-1 R R

$$B + \underbrace{(-B)}_{\text{complement}} = r^n$$

B: number , r: radix , n: # of digits

Find 8's complement $B+(-B) = r^n \rightarrow$ max

$\underline{7}\,\underline{7}\,\underline{7}_8$

for $(160)_8$?

r: 8 ocatal , n: 3digits , B=160

$160 + \square = 8^3$

8's complement of 160 is $8^3 - 160$

Find R's complement of $(123)_4$

$123 + (-B) = 4^3$ B=123 n=3 r=4

$-B = 4^3 - 123$

Scanned with CamScanner

(R-1) complement:

$$B + (-B) = \underset{\text{max}}{r}^{n} - 1$$

find (R-1) complement for $(511)_{10}$

$B = 511, \ r = 10, \ n = 3$

$511 + (-B) = 10^{3} - 1$

$10^{3} - 1 - 511 = 1000 - 1 - 511 = 488_{10}$

★ r complement ⇒ $B + (-B) = r^{n}$

★ (r-1) complement ⇒ $B + (-B) = r^{n} - 1$

Scanned with CamScanner

* Overflow: The number is too large or too small to be represented using n

Ex: perform $10_d + 4_d$ in 4 bits (binary airthmatic)

$10_d + 4_d = 14_d$ → Can't be represented in 4 bits

X range 4 bits → [0,15]

Ex: perform in binary $-13_d + -12_d$ n = 5 bits use 1's complement

$n = 5$ bits → $[-(2^{n-1}-1), (2^{n-1}-1)] = [-15, +15]$

$-13_d + -12_d = -25_d$ → Can't be represented in 5 bits

Binary Arithmetic: A+B , A−B, n: given

⊛ unsigned Addition: (A+B)

A is unsigned , B is unsigned

① represent A, B in binary unsigned format

$(\overset{\text{check}}{*}, \overset{\text{check}}{*})$

② perform addition extension

③ Check overflow

if cout = 1 → overflow

else result is correct

Scanned with CamScanner

⊛ unsigned subtraction (A − B)
A, B are both unsigned

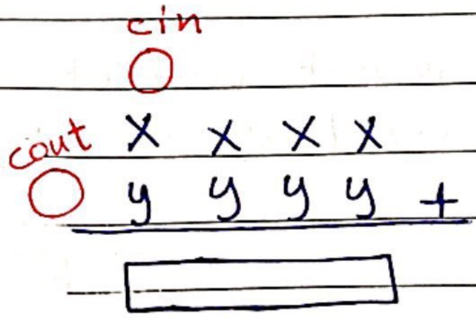
① Get A, B in binary given that
$A - B = \underset{\text{positive}}{A} + \underset{\text{negative}}{(-B)}$

- B could be represented in 1's comp or 2's comp

→ For 1's comp: A and B are represented in 1's
(* , *)
check check

perform addition
- if cout = 1 → ignore cout and add 1 to result
else final result = −(1's of answer)

→ For 2's comp:
perform addition
- if cout = 1 → ignore cout, result as it is
else final answer = − (2's complement of binary answer)



Scanned with CamScanner

⊛ signed addition, signed subtraction,
(A+B), (A−B) ⇒ A, B both signed.

① convert subtraction into addition
A+B , A−B → A+(−B)

② convert numbers (A,B) into signed
representation (1's or 2's)
A − B = A + (−B)
↳ mix sign and magnitude

③ perform addition

⇒ for 1's complement:
− if Cin = Cout → no overflow.
→ add 1 to result if cout = 1

else (cin ≠ cout) → overflow (result is not correct)

⇒ for 2's complement:
− if cin = cout → result is correct (No overflow)

else (cin ≠ cout) → result is not correct (overflow)

Scanned with CamScanner

Ex: perform the following using unsigned $(n = 6)$

$60_d + 35_d$

$60_d = (\underset{32}{1}\underset{16}{1}\underset{8}{1}\underset{4}{1}\underset{2}{0}\underset{1}{0})_{unsigned}$

$35_d = (100011)_{unsigned}$

□ cin=0

$$\begin{array}{lr} \text{cout} & 111100 \\ \boxed{1} & \underline{100011} + \\ & 011111 \end{array}$$

for 6 bits : $[0, 2^6 - 1]$

$[0, 63]$

$\therefore$ cout = 1 → overflow (incorrect)

for $n = 7$: $[0, 2^7 - 1]$

$[0, 127]$

$$\begin{array}{r} 1 \\ 0111100 \\ \underline{0100011} + \\ 1011111 \end{array}$$

cin = 1

cout = 0 ⇒ no overflow

Ex: perform the following using unsigned $(n = 7)$

$51_d - 23_d$

① $51_d + (-23)$

② convert to binary:

using 1's

$51 = 110011$ (unsigned)

$23 = 10111$

Scanned with CamScanner

S | M

+51 $\boxed{0 \mid 110011}$

S | M

-23 $\boxed{1 \mid 101000}$ sign extension

```
  1
  0110011
1 1101000+
  0011011
```

carry out = 1 → ignore

final result 0011011 + 1 = 0011100

$\boxed{0 \mid 011100}$ $= (28)_{10}$

$25.5_d - 30_d$, n=7 signed

① convert to addition

25.5 + (-30)

② to binary

(using 1's comp)

25.5 = 11001.1

30 = 11110.0

+25.5 = $\boxed{0 \mid 11001.1}$

-30 = $\boxed{1 \mid 00001.1}$

25.5-30 = -14.5

over flow ?

-14.5 = $\boxed{1 \mid 100.1}$

1 bit for S

4 bits for M

∴ 5 bits

(No over flow)

Scanned with CamScanner

$\overset{}{0}\overset{}{1}\overset{}{1}\overset{1}{0}\overset{1}{0}1.1$

$100001.1\ +$

$111011.0 \implies \boxed{1|11011.0}_{1's} = -4.5$

cin = cout = 0 → no overflow (correct)

Ex: n = 4, Find 4+4 using 1's comp

$+4 = \boxed{0|100}$

$\overset{1}{0}100$ cin = 1 , cout = 0

$0100\ +$ cin ≠ cout → overflow

1000 (wrong result)

Ex: n = 5, Find 3-14 unsigned using 1's comp

3 = 0011 → +3 = 00011

14 = 1110 → -14 = 10001

$00\overset{1}{0}\overset{1}{1}1$

$10001\ +$

10100

cout = 0

∴ the answer is -01011 = $(-11)_{10}$

Scanned with CamScanner

Ex: Find $-1.5_d + 9.75$ (signed) n=7, 1's comp

$1.5 = (1.10)_b \rightarrow \overset{S}{\boxed{1}}\overset{M}{\boxed{0.01}} \rightarrow \overset{S}{\boxed{1}}\overset{M}{\boxed{1110.01}}$

unsigned

$9.75 = (1001.11) \rightarrow \boxed{0}\boxed{1001.11} \rightarrow \boxed{0}\boxed{1001.11}$

cin $\boxed{1}$ 1 1 1 1 1

11110.01

$\boxed{1}$ 01001.11+

cout

01000.00

cin = cout → no overflow

cout = 1 ⇒ final result = 01000.00 + 1

= 01000.01 = 8.25

Ex: Find −45 −90 signed, 2's comp

−45 + (−90)

$45 = 101101 \rightarrow \boxed{1}\boxed{010011}$ 2's

$90 = 1011010 \rightarrow \boxed{1}\boxed{0100110}$ 2's

cin $\boxed{\ }$ 1 1

11010011

cout

$\boxed{1}$ 10100110+

01111001

∴ cin ≠ cout ⇒ overflow

Scanned with CamScanner

Ex: Find the value:

cin→ $\boxed{1}$ 10 1 1

101101 ← signed 1's comp

001010

cout→ $\boxed{1}$ 111110 +   cin = cout → no overflow

110101

1

110101

1 +   ⇒ final answer: (-) 001001 = -9

110110

Ex: Find the value:

$(011010)_{1's}$

$(\quad 111)_{2's}$ +

option 1:

$(011010)_{1's} = (011010)_{2's}$

$(111)_{2's} = (111111)_{2's}$

cin → $\boxed{1}$ 1 1 1

011010

cout→ $\boxed{1}$ 111111 +

011001   ∴ cin = cout → answer is 011001

Scanned with CamScanner

Option 2:

$(111)_{2's} = (111111)_{2's} = (111110)_{1's}$

cin → $\boxed{1}111$

$(011010)_{1's}$

cout → $\boxed{1}(111110)_{1's}$ +

011000 cin = cout → no overflow

011000 + 1 = 011001

↳ binary code of $+25_d$

Binary code: string of zeros and ones to represent another quantities

$5_H \rightarrow (10101)_2$ $5.5_d \rightarrow (101.1)_2$

Binary codes:

① Character coding (ASCII)
② Gray code
③ BCD: Binary coded Decimal
④ Unicode

$\boxed{1}$ ASCII:

128 character → n = 7 digits (bits)
Byte = 8 bits so we use 8 digits to represent ASCII code characters (n = 8)
The extra digit is used for parity (error detection)

Scanned with CamScanner

✲ Even Parity : (even number of ones)

A = 1000001 ⇒ 10000010
a = 1100001 ⇒ 11000011

sender ⟶ receiver

11010100 11000100

number of ones n = 4 (even)

n = 3 (odd)
error detected

10110010 11010010

n = 4 (even) n = 4 (even)

No error detected

✲ odd parity : (odd number of ones)

A = 1000001 ⇒ 10000011
a = 1100001 ⇒ 11000010

sender ⟶ receiver

11000001 11010101

n = 3 (odd) n = 5 (odd)

no error detected

✲ disadvantages of parity detection:

1) No error recovery (location of error can't be detected)

2) Can detect odd number of errors only

Scanned with CamScanner

$\boxed{2}$ Gray code: (reflected binary)

000
001
011
010

110
111
101
100

we used 4 numbers in gray to generate another 4 numbers in gray

0000
0001
0011
0010
0110
0111
0101
0100

1100
1101
1111
1110
1010
1011
1001
1000

✡ conversion between binary and gray:

And, or, not, XOR ↖exclusive or

1101 and 0101 = 0101

1 XOR 1 ⇒ 0 , 1 XOR 0 ⇒ 1
0 XOR 0 ⇒ 0 , 0 XOR 1 ⇒ 1

1101 XOR 0101 = 1000

Scanned with CamScanner

☆ Convert from binary to gray :

① copy MSB

② $G[i] = XOR(B[i], B[i+1])$

$(1011101)_B$

↓

$(1110011)_G$

$(010111101)_B$

↓

$(011100011)_G$

☆ Convert from Gray to binary:

① copy MSB

② $B[i] = XOR(G[i], B[i+1])$

$(1110011)_G$

↓

$(1011101)_B$

Scanned with CamScanner

$\boxed{3}$ BCD: Binary coded decimal

(0-9) not as Hexa (act as Hexa)

$(567)_{10} \rightarrow (\underbrace{0101}_{5}\ \underbrace{0110}_{6}\ \underbrace{0111}_{7})_{BCD}$

$(C12)_{H} \rightarrow (\ X\ )_{BCD}$

$(7890)_{10} \rightarrow (\underbrace{0111}_{7}\ \underbrace{1000}_{8}\ \underbrace{1001}_{9}\ \underbrace{0000}_{0})_{BCD}$

☆ Byte representation of BCD $(7890)_{BCD}$

→ option 1: (Packed representation)

$\boxed{0111\ \ 1000}$	$\boxed{1001\ \ 0000}$
byte 1	byte 2

→ option 2: (unpacked representation)

$\boxed{\text{empty\_}0111}$	$\boxed{1000\ \text{empty}}$	$\boxed{1001\ \text{empty}}$	$\boxed{\text{empty}\ 0000}$
BCD1	BCD2	BCD3	BCD4

Gray → 2 → 10 → BCD (convet from gray to BCD)

Hexa → 10 → BCD (convert from Hexa to BCD)

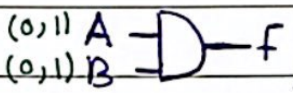
Scanned with CamScanner

4 Unicode: (same as ASCII but support other languages)

Ch 5: Boolean algebra and logic gates:

☆ logic gates → fundamental
→ non fundamental

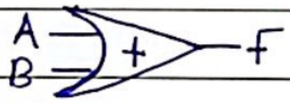
1 fundamental:

① And: $A \overset{And}{\cdot} B$, 

, truth table (behavior)

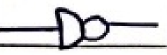
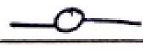
A	B	F	
0	0	0	false
0	1	0	
1	0	0	
1	1	1	true

A, B → inputs
F → output

② OR: $A + B$, 

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

Scanned with CamScanner

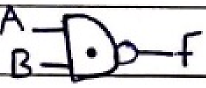
③ Not : $\overline{A}$, $\overline{B}$, A  $\overline{A}$, B  $\overline{B}$

A	$f = \overline{A}$
0	1
1	0

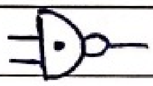
☆ Every input or output wire can carry 1 bit

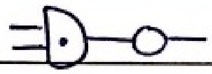
2 Non fundamental

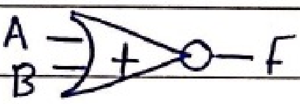
① NAND : (not And)

$\overline{A \cdot B}$, $(A \uparrow B)$, 

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

 → 1 gate (Nand)

 → 2 gates (And, not)

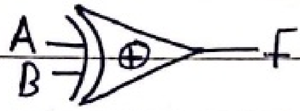
② NOR : $\overline{A+B}$, 

A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

$(A \downarrow B)$

Scanned with CamScanner

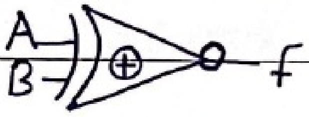
③ XOR : $A \oplus B$,



A	B	f
0	0	0
0	1	1
1	0	1
1	1	0

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

④ XNOR : $\overline{A \oplus B}$,



A	B	f
0	0	1
0	1	0
1	0	0
1	1	1

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

truth table : inputs | outputs

number of rows = $2^{\text{number of Inputs}}$

number of columns = number of inputs + number of outputs

Scanned with CamScanner

$\to A \cdot B$

✲ Logic gate : hardware that can do boolean function

✲ bolean expression: Connection of multiple boolean operation Ex: $(\bar{A} \cdot \bar{B}) + (C + \bar{D})$ - - - -

✲ term: part of expression that is evaluated as one unit $(\bar{A} \cdot \bar{B}) + (C + \bar{D}) \to$ 2 terms

✲ Literal: variable name

$(\bar{A} \cdot \bar{B}) + (C + \bar{D}) \to$ 4 literals

→ To evaluate a boolean expression
- $-$
- $+$
- $\oplus$
- $\uparrow$
- $\downarrow$

Remainder ✲ all inputs should be in binary $\{0, 1\}$

✲ result should be binary

✲ priority ( ) first

✲ Not

✲ And

✲ OR

$$F = \underset{XOR}{A \oplus B} = \overbrace{\bar{A} \cdot B}^{\text{term1}} + \overbrace{A \cdot \bar{B}}^{\text{term 2}}$$

XOR = 2 Not

2 And

1 OR

$$F = \underset{NOR}{A \downarrow B} = \overline{A + B} = \overline{(A + B)}$$

Scanned with CamScanner

⑤ Last ③ not x.C ④ (not x.c).E

Ex: Evaluate $f(A,B,C,D,E) = D + \overline{\underbrace{(A+B)}_{x}C} \cdot E$

① A OR B ② x.C

$A,B=0$, $C=1$, $D=0$, $E=1$

① $(A+B) = 0 \rightarrow f = D + \overline{0 \cdot C} \cdot E$

② $0 \cdot C = 0 \rightarrow f = D + \bar{0} \cdot E$

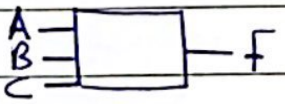
③ Not $= 1 \rightarrow f = D + (1 \cdot E)$

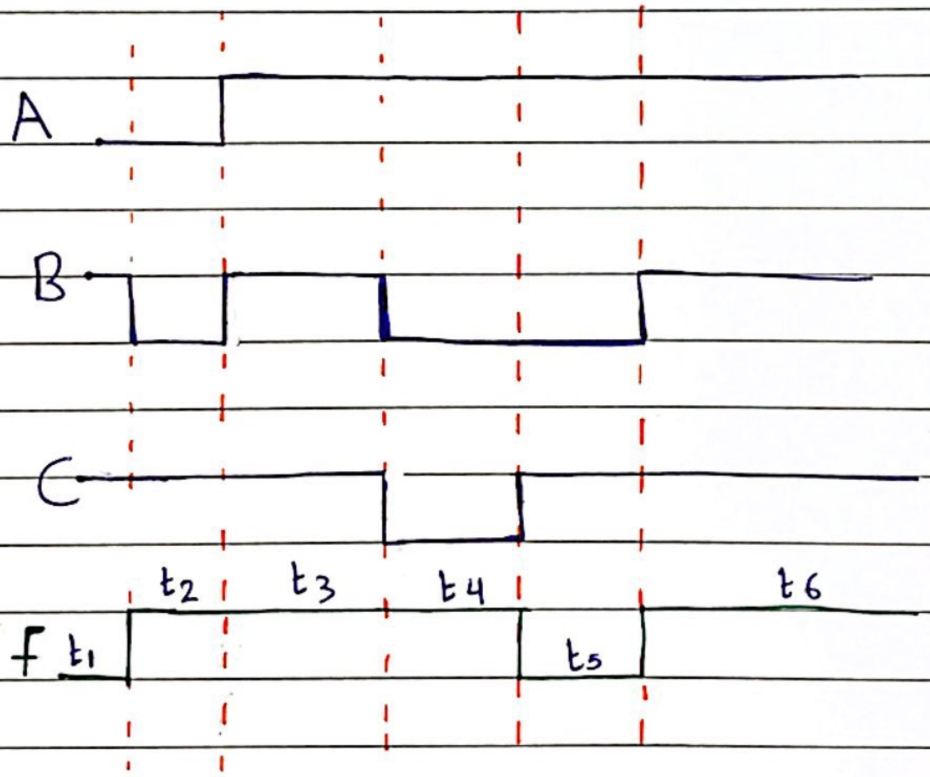
④ And $1 \cdot E = 1 \rightarrow f = D + 1$

⑤ OR $f = 1$

★ Evaluation based on timing diagram:

$f(A,B,C) = A \oplus B \oplus C$





N O T E B O O K

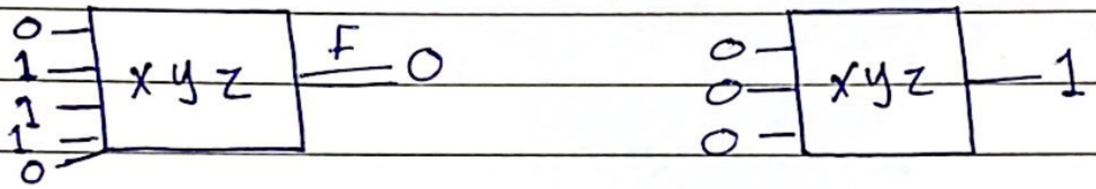
Scanned with CamScanner

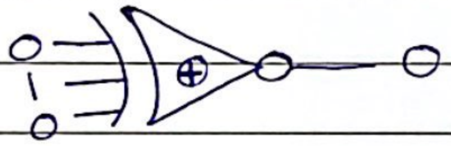
XOR is odd gate → produces 1 if there are odd number of 1 in input

1 XOR 1 XOR 1 XOR 1 = 0

1 XOR 1 XOR 0 XOR 1 = 1

Assume an even gate xyz





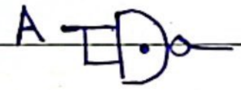
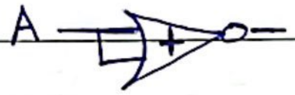
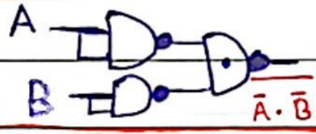
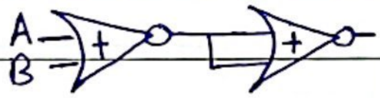
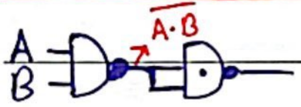
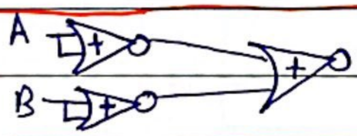
Scanned with CamScanner

Equivalency of logic gates: (universality)

- A universal gate is a gate that can act as Not/OR/And (can be fundamental)

- NAND and NOR are both universal

(NOR), (NAND) can work as Not, OR, AND

NAND	NOR	
		Not
		OR (A+B)
		AND (A·B)

$\overline{\bar{A}\cdot\bar{B}} = A+B$ → NAND As OR

$\overline{A\cdot A} = \bar{A}$ → NAND As Not

$$\left.\begin{aligned} y = x = \overline{A\cdot B} &= \overline{x\cdot y} \\ &= \overline{(\overline{A\cdot B})\cdot(\overline{A\cdot B})} \\ &= \overline{(\overline{A\cdot B})} = A\cdot B \\ &= A\cdot B \end{aligned}\right] \rightarrow \text{NAND As AND}$$

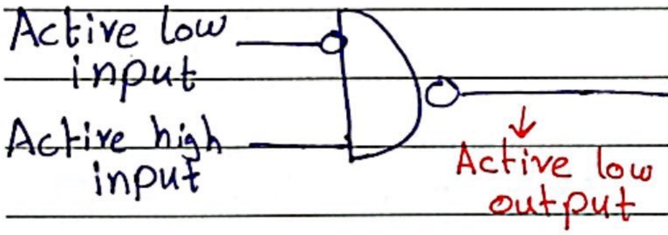
Scanned with CamScanner

$\overline{A+A} = \bar{A}$ → NoR As Not

$$\left.\begin{aligned} x = y &= \overline{A+B} = \overline{x+y} \\ &= \overline{\overline{(A+B)} + \overline{(A+B)}} \\ &= \overline{\overline{(A+B)}} \\ &= A+B \end{aligned}\right] \rightarrow \text{NoR As OR}$$

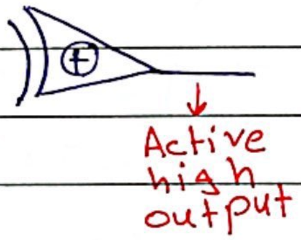
$$\left.\begin{aligned} x = \bar{A}&,\quad y = \bar{B} \\ \overline{x+y} &= (\overline{\bar{A}+\bar{B}}) \\ &= A \cdot B \end{aligned}\right] \rightarrow \text{NoR As AND}$$

✡ NoR, NAND are universal



1 = on = true = active

0 = off = false = not active

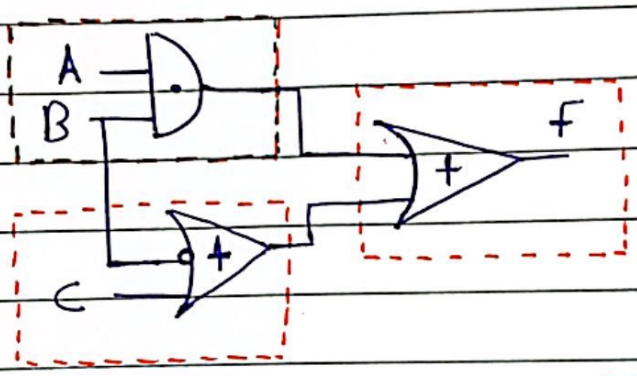


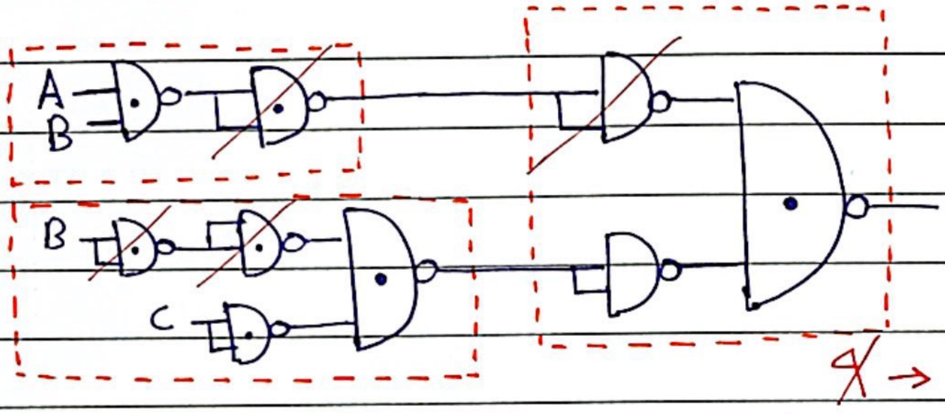
$A \cdot A = A \xrightarrow{\text{extension}} A \cdot A \cdot A \cdot A \cdots\cdots = A$

$A + A = A \xrightarrow{\text{extension}} A + A + A + A \cdots\cdots = A$

Scanned with CamScanner

Ex: Draw using NAND only





~~4~~ → 5 is minimum

★ Boolean Algebra values:

Duality: you can get a dual for any expression by replacing:

OR → AND

AND → OR

0 → 1

1 → 0

① closure:

$$B = \{0, 1\}$$

$X, y \in B$

$X + y \in B$

$X \cdot y \in B$

② Identity:

$0 + X = X$, $1 \cdot X = X$

③ commutative:

$$\begin{tikzcd} X + y = y + X \arrow[d, "dual"'] \\ X \cdot y = y \cdot X \Rightarrow Xy \end{tikzcd}$$

④ Distributive:

$X \cdot (y + z) = X \cdot y + x \cdot z$

$x + (y \cdot z) = (x + y) \cdot (x + z)$

⑤ Complement:

$X + \overline{X} = 1$

$X \cdot \overline{X} = 0$

⑥ Idempotency:

$\left.\begin{array}{l} X + X + X \text{ ------- } = X \\ X \cdot X \cdot X \text{ ------- } = X \end{array}\right]$ $x + y + x + x + x = x + y$

⑦ Null element:

$\overbrace{\text{-----} + X\overline{z} + y + X}^{\text{don't care}} + 1 = 1$

$\underbrace{\text{------} \cdot \overline{y} \cdot \overline{x}\, z \cdot X}_{\text{don't care}} \cdot 0 = 0$

Scanned with CamScanner

⑧ Involution: $\overline{\overline{X}} = X$, $\overline{\overline{\overline{X}}} = \overline{X}$, $\overline{\overline{\overline{\overline{X}}}} = X$

⑨ Associative: $X + (Y+Z) = (X+Y) + Z = X + Y + Z$

$(X\cdot Y)\cdot Z = X\cdot(Y\cdot Z) = X\cdot Y\cdot Z$

The same gate

⑩ Demorgan: $(\overline{X(\overline{Z}+\overline{W})}) = \overline{X} + Z\cdot W$

$(\overline{X\cdot Y}) = \overline{X} + \overline{Y}$

⑪ Absorption (common factor):

$X + XY = X(1+Y) = X\cdot 1 = X$

$X\cdot(X+Y) = X\cdot X + X\cdot Y = X + XY$

⑫ consensus: $XY + \overline{X}Z + YZ = XY + \overline{X}Z$

⑬ equivalency: $\underbrace{\overline{X}\cdot X}_{\text{2 literals and 1 gat}} = \underbrace{0} \rightarrow$ constant zero

Ex: Simplify: $\cancel{A\cdot A}_{A} + AC + AB + BC \rightarrow$ 3 ORs 4 ands

$A + AC + AB + BC$

① $A(\underbrace{1+\cancel{C+B}}_{1}) + BC$

② $A\cdot 1 + B\cdot C$

③ $A + B\cdot C \rightarrow$ 1 OR 1 and

Scanned with CamScanner

$$\boxed{\begin{aligned} x + \bar{x}y &= x + y \\ \bar{x} + xy &= \bar{x} + y \end{aligned}}$$

Ex: Simplify: $\underline{\bar{A}BC} + \underline{A\bar{B}\bar{C}} + \underline{\bar{A}\bar{B}\bar{C}} + A\bar{B}C + \underline{ABC}$

$BC(\underbrace{\cancel{\bar{A}+A}}_{1}) + \bar{B}\bar{C}(\underbrace{\cancel{A+\bar{A}}}_{1}) + A\bar{B}C$

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

$BC + \bar{B}(\bar{C} + AC)$

$BC + \bar{B}((\bar{C}+A)\cdot\underbrace{\cancel{(\bar{C}+C)}}_{1})$

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

Simplify: $\underline{xy} + \underline{\bar{x}z} + yz$

$xy + \bar{x}z + yz\cdot 1 \quad , \quad 1 = \bar{x} + x$

$xy + \bar{x}z + \underline{y\cdot z}(\bar{x}+x)$

$xy + \bar{x}z + xyz + \bar{x}yz = xy(\underbrace{\cancel{1+z}}_{1}) + \bar{x}z(\underbrace{\cancel{1+y}}_{1})$

$xy + \bar{x}z$

Minterm: a row of truth table in which the function produces $\underline{\underline{1}}$ and the boolean expression of minterm consists of all inputed sorted Anding together

$f(x,y,z) \rightarrow$ minterm $\Rightarrow x\cdot y\cdot z,\ \bar{x}\cdot y\cdot z \quad , \quad \underbrace{\bar{y}}_{\text{not minterm}} + x + z$ ✗

Scanned with CamScanner

☆ Maxterm: a row of truth table in which the function produces zero, the boolean expression consists of all inputs sorted grouped by OR

$F(A,B,C) \Rightarrow \underset{\text{maxterm}}{\bar{A}+\overset{\downarrow}{\bar{B}}+C}\ ,\ \underset{\text{maxterm}}{A+\overset{\downarrow}{B}+C}\ ,\ \underset{\text{not max term}}{A\overset{\downarrow}{+}\bar{C}+B}$

	A B C	F= ?
row 0	0 0 0	$\boxed{0} \rightarrow$ max term M_0 $F=A+B+C$
row 3	0 1 1	$\boxed{1}$ minterm $m_3=\bar{A}\cdot B\cdot C$
row 5	1 0 1	$\boxed{1}$ $m_5=A\cdot\bar{B}\cdot C$
row 6	1 1 0	$\boxed{0} \rightarrow$ max term $M_6=\bar{A}+\bar{B}+C$
row 7	1 1 1	$\boxed{0} \rightarrow$ max term $M_7=\bar{A}+\bar{B}+\bar{C}$

$F(x,y,z) = m_0 + m_6 \Rightarrow$ at row zero and row 6 $\Rightarrow F=1$

row 1, 2, 3, 4, 5, 7 $\Rightarrow F=0$

(OR) sum product
+

☆ SOP: All minterms grouped by OR

$\Rightarrow \underset{m_k}{(\ldots)} + \underset{m_j}{(\ldots)} + \underset{m_i}{(\ldots)}$ i, j, k → rows

↳ search when Function output is 1

☆ POS: all maxterms grouped by AND

AND (·) OR(+)

$\Rightarrow (\ldots+\ldots+\ldots)\cdot(\ldots+\ldots+\ldots)\cdot(\ldots+\ldots+\ldots)$

↳ search when function output is zero

Scanned with CamScanner

Ex: $F(X, Y, Z) = \underset{m_7}{\overset{1}{X} \cdot \overset{1}{Y} \cdot \overset{1}{Z}} + \underset{m_4}{\overset{1}{X}\,\overset{0}{\bar{Y}}\,\overset{0}{Z}}$ → SOP

Ex: $f(x,y,z) = (x + \bar{y}) \cdot \bar{z}$ → non standard not POS, not SOP

① zero @ z=1

② zero @ x=0 and y=1

→ * POS (Maxterm)

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

* List all max terms : M_1, M_2, M_3, M_5, M_7

$f = M_1 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_7$

at row 1, 2, 3, 5, 7 ⇒ f = 0

$(\underset{0}{x} + \underset{0}{y} + \underset{1}{\bar{z}}) \cdot (\underset{0}{x} + \underset{1}{\bar{y}} + \underset{0}{z}) \cdot (\underset{0}{x} + \underset{1}{\bar{y}} + \underset{1}{\bar{z}}) \cdot \text{------}$

Scanned with CamScanner

* List all minterms: m_0, m_4, m_6

at row zero and row 4 and row 6 → $F = 1$

$f = m_0 + m_4 + m_6$

$$(\underset{0}{\bar{x}} \cdot \underset{0}{\bar{y}} \cdot \underset{0}{\bar{z}}) + (\underset{1}{x} \cdot \underset{0}{\bar{y}} \cdot \underset{0}{\bar{z}}) + (\underset{1}{x} \cdot \underset{1}{y} \cdot \underset{0}{\bar{z}})$$

Ex: $f_1(x, y, z, w) = 1$, How many maxterms or minterms totally? $2^4 = 16$

Tytes of boolean expression:

1) SOP $\left(\begin{matrix}\text{complete}\\\text{canonical}\end{matrix}\right)$: $m_i + m_j + m_k$ $\quad \rightarrow f(A, B, C)$ $\quad (m_{\square} = \bar{A} \cdot B \cdot \bar{C})$

2) POS $\left(\begin{matrix}\text{complete}\\\text{canonical}\end{matrix}\right)$: $M_i \cdot M_j \cdot M_K$ $\quad (M = \bar{X} + y + \bar{Z} + w)$ $\quad \hookrightarrow f(x, y, z, w)$

3) SOP $\left(\begin{matrix}\text{not complete}\\\text{not canonical}\end{matrix}\right)$: $m + m + m$ ✗

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

4) POS $\left(\begin{matrix}\text{not complete}\\\text{not canonical}\end{matrix}\right)$ $M \cdot M \cdot M$ ✗

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

5) $f(x, y, z, w) = x \cdot y + z(\bar{x} + \bar{y})$ → not SOP not POS

$= x \cdot y + z\bar{x} + z\bar{y}$ → SOP $\left(\begin{matrix}\text{not}\\\text{complete}\end{matrix}\right)$

Scanned with CamScanner

Ex : $f(X,y,Z) = \underbrace{\bar{X}Z}_{\text{add } y\bar{y}} + \underbrace{\bar{y}}_{\text{add } xz}$

(expansion)

equivalent $= \bar{X}Z\cdot 1 + \bar{y}\cdot 1\cdot 1$

$= \bar{X}\cdot Z\cdot(y+\bar{y}) + \bar{y}(X+\bar{X})(Z+\bar{Z})$

$= \bar{X}Zy + \bar{X}Z\bar{y} + \bar{y}XZ + \bar{y}X\bar{Z} + \cancel{\bar{y}\bar{X}Z} + \bar{X}\bar{y}\bar{Z}$

مكرر

$= \underset{011}{\bar{X}yZ} + \underset{001}{\bar{X}\bar{y}Z} + \underset{\phi 01}{X\bar{y}Z} + \underset{100}{X\bar{y}\bar{Z}} + \underset{000}{\bar{X}\bar{y}\bar{Z}}$

$f(X,y,Z) = m_3 + m_1 + m_5 + m_4 + m_0$

rows $(0,1,3,4,5) \Rightarrow f=1$

rows $(2,6,7) \Rightarrow f=0$

$= \Sigma(0,1,3,4,5)$

$F(X,y,z) = \bar{X}Z + \bar{y}$

① make it POS (uncomplete)

$(\underbrace{\bar{x}+\bar{y}}_{z})\cdot(\underbrace{z+\bar{y}}_{x})$

② Add missing variables

$= (\bar{X}+\bar{y}+0)(z+\bar{y}+0)$

$= (\bar{X}+\bar{y}+z\cdot\bar{z})\cdot(z+\bar{y}+x\cdot\bar{x})$

③ Distribute and sort

POS (complete) $= \underset{1\,1\,0}{(\bar{X}+\bar{y}+Z)}\underset{1\,1\,1}{(\bar{X}+\bar{y}+\bar{Z})}\underset{0\,1\,0}{(X+\bar{y}+Z)}\underset{1\,1\,0}{(\bar{x}+\bar{y}+z)}$

$= M_6\cdot M_7\cdot M_2 \Rightarrow \Pi(2,6,7)$

Scanned with CamScanner

✲Equivalency rules:

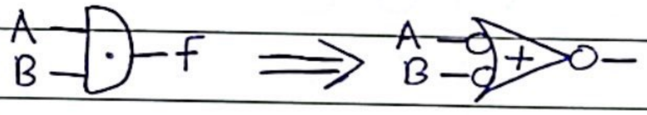
① expression:

$F(A,B,C,D) = \bar{A}\cdot(\bar{C}+D)$ ① (Active high)

$= \bar{A}\bar{C} + \bar{A}D$ ②

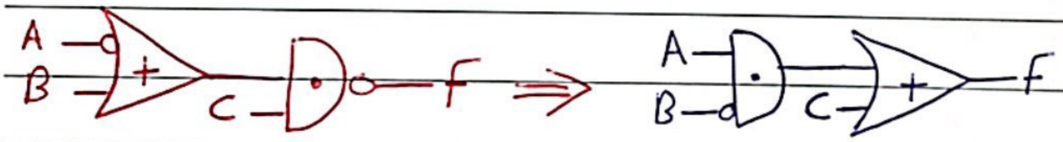
$= \overline{A + (C\cdot\bar{D})}$ ③ (Active low)

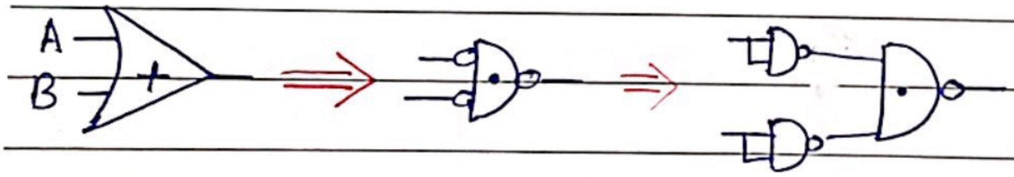
② Circuit:

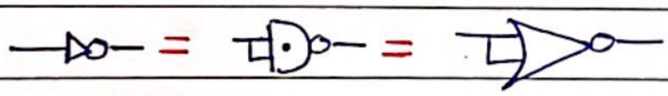


$f = A\cdot B \Rightarrow \overline{\bar{A} + \bar{B}}$

① invert all inputs
② change OR by AND
  // AND // OR
③ invert output finally







Scanned with CamScanner

KMAP: mechanism to reach simplist form for any boolean expression
- → not standard < SOP, POS
- → standard < SOP, POS

1) Draw KMAP:
- → tabular format for truth table
- → Matrix of same truth table size
- → Use gray for column # and row #

f(A,B)

A	B	F
0	0	x
0	1	x
1	0	x
1	1	x

input KMAP ⟹

4 cells

Cell ≈ row

f(X, y, Z)

xy z	00	01	11	10
0	0	2	6	4
1	1	3	7	5

OR

z xy	0	1
00	0	1
01	2	3
11	6	7
10	4	5

f(A,B,C,D) → 16 rows ≈ 16 cells

AB CD	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

Scanned with CamScanner

2) fill KMAP from the table

$f(A,B,C) = A\bar{B}C + B\bar{C}$ ✲ Sorting is important

$^{BC}_{A}$	00	01	11	10
0	ABC 000 row0 0	ABC 001 row1 0	row3 0	row2 1
1	row4 0	row5 1	row7 0	row6 1

$f = 1$ if:

option 1: if $A\cdot\bar{B}\cdot C = 1$ } 1 cell
A=1 & B=0 & C=1

option 2: if $B\cdot\bar{C} = 1$ } 2 cells A=0 A=1
B=1 & C=0

$m_5 + m_2 + m_6 = \Sigma(2,5,6)$

$M_0 \cdot M_1 \cdot M_3 \cdot M_4 \cdot M_7 = \Pi(0,1,3,4,7)$

3) Form groups based on SOP or POS:

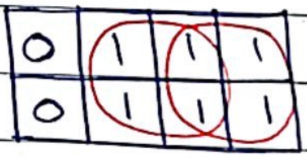
① group must have same value → 1 ⇒ SOP
→ 0 ⇒ POS

② group must be 1 cell, 2 cells, 4 cells, 8 cells or 16 cells

ADJACENT: Start by any cell and move to another cells that the binary ID is different by 1 digit, Stop when you reach starting cell

Scanned with CamScanner

③ maximize group size to minimize # of groups must cover all cells → SOP 1 → POS 0 even if you cover same cells more than once

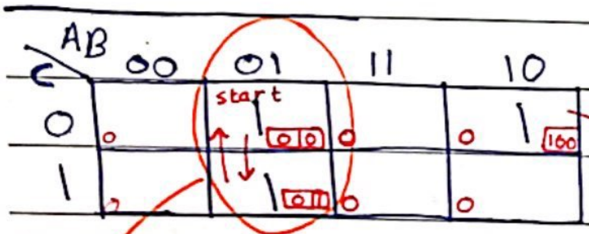


→ 3 groups each 2 cells
→ 2 groups each 4 cells

4) Find boolean expression for each group:

SOP: each group is minterm
Mix groups by OR

POS: each group is maxterm
Mix groups by AND



f(A, B, C)

$A\bar{B}\bar{C}$
100

$A\bar{B}\bar{C}$
100

SOP: ~~2 groups~~ → one group 2 cells
→ another group 1 cell

$AB\cancel{C} \rightarrow \bar{A}.B$
0 1 $\cancel{0,1}$

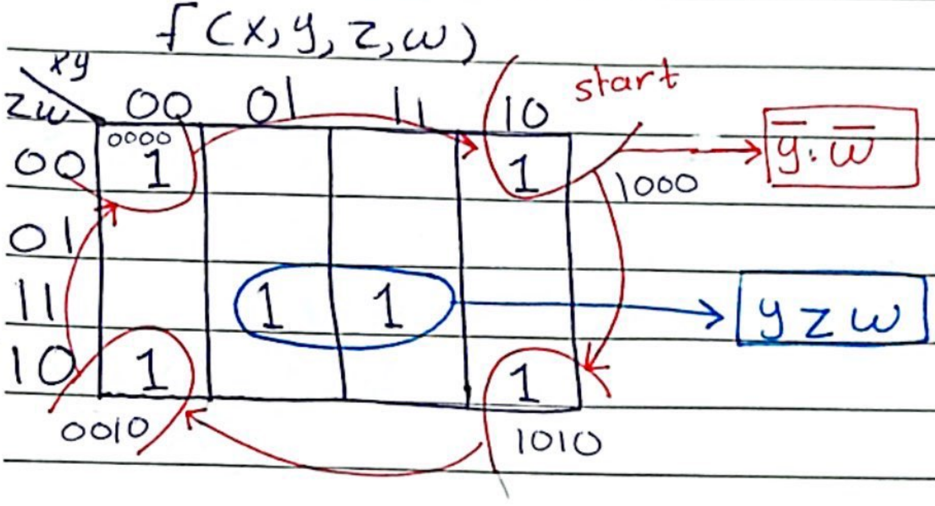
SOP: $\bar{A}B + A\bar{B}\bar{C}$
→ non complete SOP

$m_2 + m_3 + m_4$
$= \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}\bar{C}$
010 011 100
→ Complete SOP

NOTEBOOK

Scanned with CamScanner

$f(x,y,z,w)$



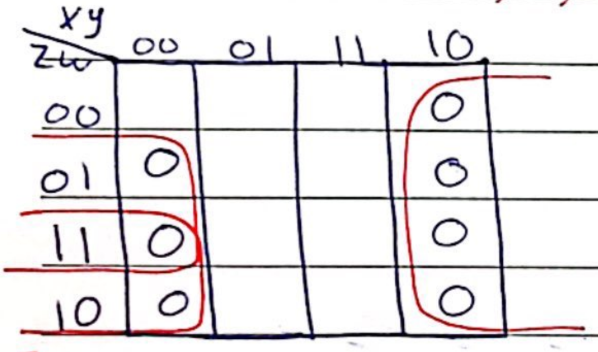
zw \ xy	00	01	11	10
00	1			1
01				
11		1	1	
10	1			1

$$\therefore f(x,y,z,w) = \bar{y}\bar{w} + yzw$$

$f(x,y,z,w) = \Pi(1,2,3,8,9,10,11)$ simplify f to minimum format:

rows: 1, 2, 3, 8, 9, 10, 11 → f = zero

rows: 0, 4, 5, 6, 7, 12, 13, 14 → f = 1

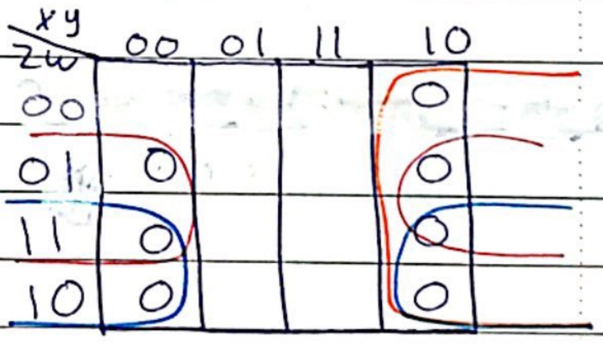


zw \ xy	00	01	11	10
00				0
01	0			0
11	0			0
10	0			0

3groups :

(4cells, 2cells, 2cells)

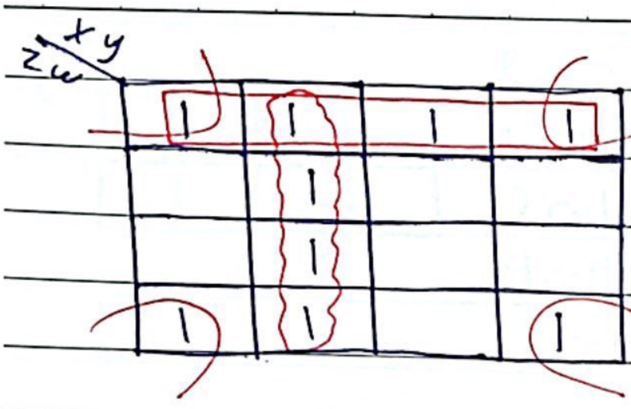
(4cells, 1cell, 2cells)



zw \ xy	00	01	11	10
00				0
01	0			0
11	0			0
10	0			0

3groups each 4 cells (SOP)

$$F = (y+\bar{w})\cdot(\bar{x}+y)\cdot(y+\bar{z})$$



1	1	1	1
	1		
	1		
1	1		1

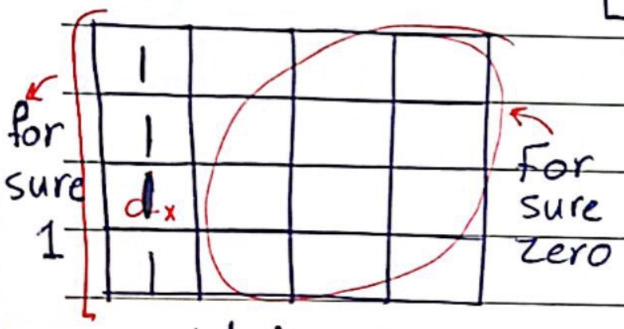
complete (standard SOP)

$= \Sigma(1,4,12,8,5,6,7,2,10)$

Simplify: $\bar{x}y + \bar{z}\bar{w} + \bar{y}\bar{w}$

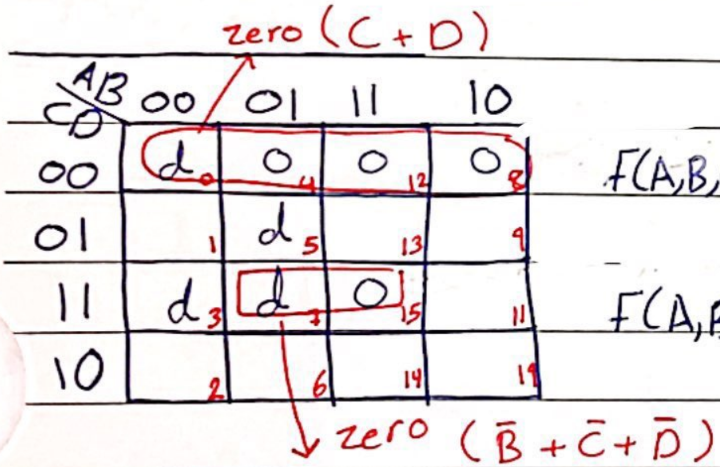
don't care:

Binary values
- → zero = off = false
- → 1 = on = true
- → X یا d don't care
  - ↳ this constant is not important you can use it as zero or 1



1			
1			
d_x			
1			

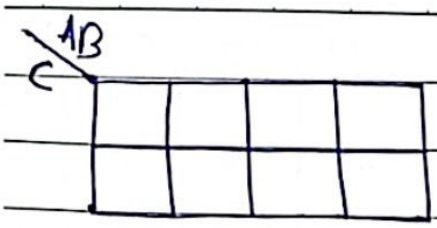
could be zero
could be 1



CD \ AB	00	01	11	10
00	d_0	0_4	0_{12}	0_8
01	$_1$	d_5	$_{13}$	$_9$
11	d_3	d_7	0_{15}	$_{11}$
10	$_2$	$_6$	$_{14}$	$_{10}$

$F(A,B,C,D) = \Pi(4,8,12,15) + \Sigma d(0,3,5,7)$

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



for 3 variables KMAP

1 cell => 3 variables

2 cells => 2 //

4 cells => 1 variable

8 cells => constant (0,1)

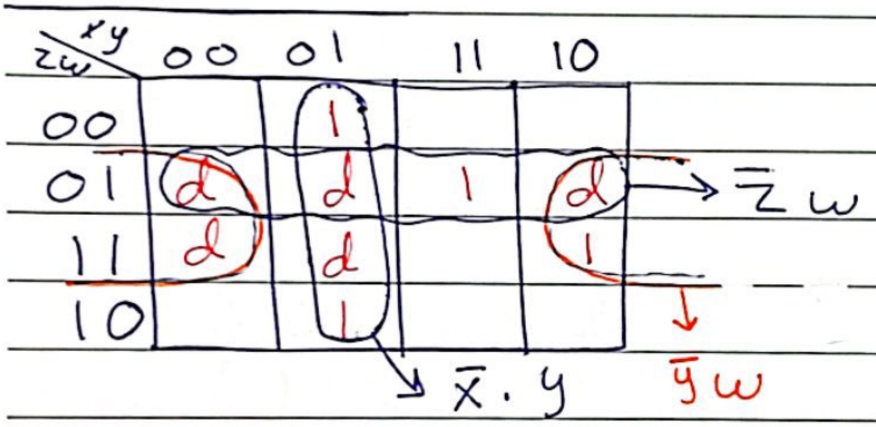
don't care

Binary values → ~~zero = off = False~~

→ 1 = ON = true

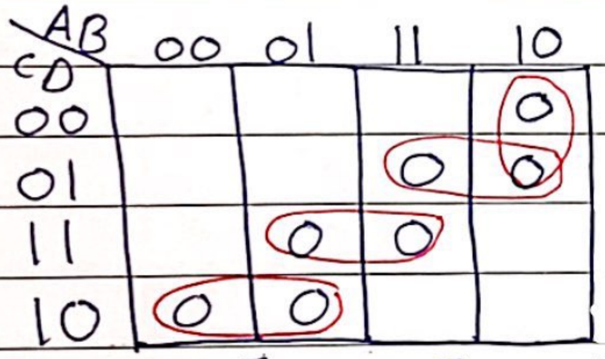
→ X او d don't care

this constant is not inportant you can use it as zero or 1



zw \ xy	00	01	11	10
00		1		
01	d	d	1	d
11	d	d		1
10		1		

~~$F(x,y,z,w) = \bar{z}w + \bar{y}w + \bar{x}y$~~



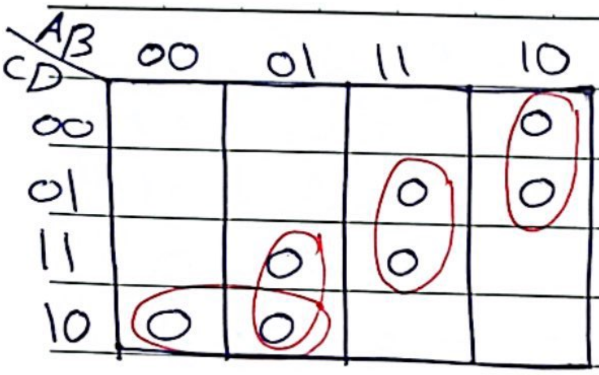
CD \ AB	00	01	11	10
00				0
01			0	0
11		0	0	
10	0	0		

4 groups (each 2 cells)

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

4 terms, 3 Literals

Scanned with CamScanner



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

4 groups (each 2 cells)

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

4 terms , 3 literals

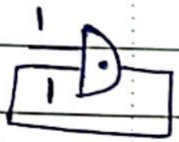
* Combinational circuits:

- built from gates
- No feed back
  - → discrete
  - → No clock needed
  - → Current output

f=0 discrete

* Sequential circuits:

- built from flip flops or latches
- Feed back exist
  - → continous
  - → needs clock (to pause circuit for very short time)
  - → times



continous clock → f= 1, 1, 1, 1, - - - -

times → old ~~output~~ state, current ~~output~~ state, Next ~~output~~ state

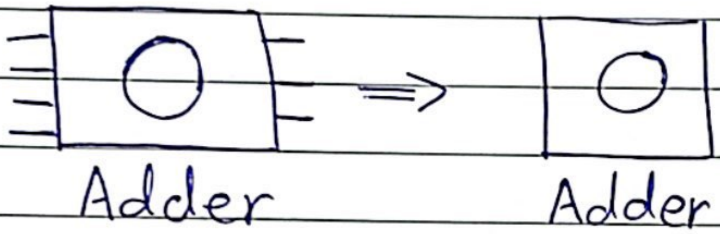
Scanned with CamScanner

Subject Date No.

Combinational circuits to be covered:

1) Adder
2) Decoder
3) Encoder
4) Magnitude comparator
5) Multiplexer

Circuit diagram



# of inputs
# of outputs
internal connection

✲ function
✲ boolean expression for every output
✲ How and what it is used for?

Chapter 8: Design and analysis of

① You are given the circuit diagram
✲ truth table
✲ KMAP
✲ boolean expression of each output

Scanned with CamScanner

② you are given a statement describing what the circuit should be (more complex).

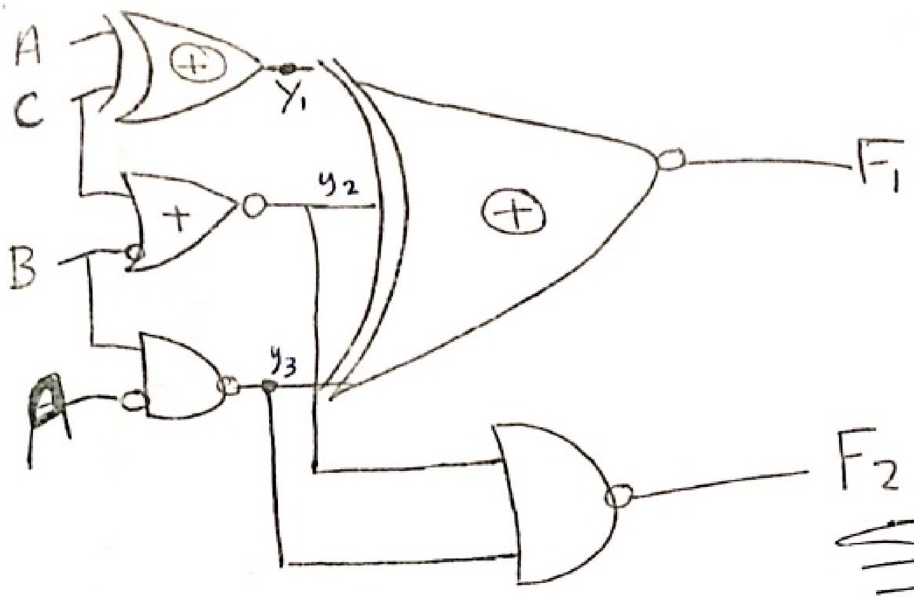
* analyze the statemen find → inputs
  → outputs
  → Relation
* truth table
* KMAP (for complex problems)
* boolean expression
* Circuit diagram

* for 4 inputs KMAP:

group of 1 cell ⇒ 4 variables
// // 2 cells ⇒ 3 //
// // 4 // ⇒ 2 //
// // 8 // ⇒ 1 variable
// // 16 // ⇒ constant (0,1)

Scanned with CamScanner

Analyze the following Combinalional circuit.



Steps

* ~~truth~~ table

* KMAP (if needed)

* boolean expression of every output

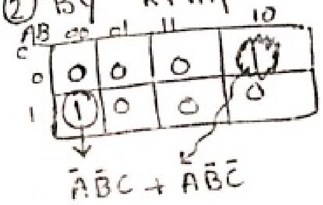
A	B	C	F1	F2
0	0	0	0	1*
0	0	1	1	1*
0	1	0	0	1
0	1	1	0	1*
1	0	0	1	1*
1	0	1	0	1*
1	1	0	0	0
1	1	1	0	1*

Finding Expression of F1

① By Minterms

$m_1 + m_4 = \bar{A}\bar{B}C + A\bar{B}\bar{C}$

② By KMAP



C \ AB	00	01	11	10
0	0	0	0	1
1	1	0	0	0

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

③ By Roles

Hard

Finding Expression of F_2

① one Maxterm ($M_6 \Rightarrow \bar{A}+\bar{B}+C$)

② $F_2 = \overline{\overline{(C+\bar{B})}\cdot\overline{(B\cdot\bar{A})}}$

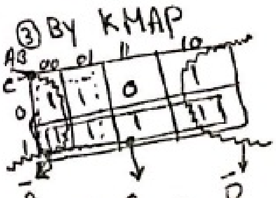
$= (C+\bar{B}) + (B\cdot\bar{A})$

by distribution

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

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

③ By KMAP



C \ AB	00	01	11	10
0	1	1	0	1
1	1	1	1	1

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

Scanned with CamScanner

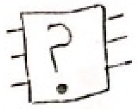
Design a Combinational circuit to Convert 3 binary bits into gray.

Solution

[1] Determine # of inputs, # of outputs + circuit behavior by analyzing the given statement.

⇒ As we learned in Lecture 4, you can convert binary code into gray by applying the fallowing equation

$G[i] = XOR(B[i+1], B[i])$ {for all bits Except MSB}

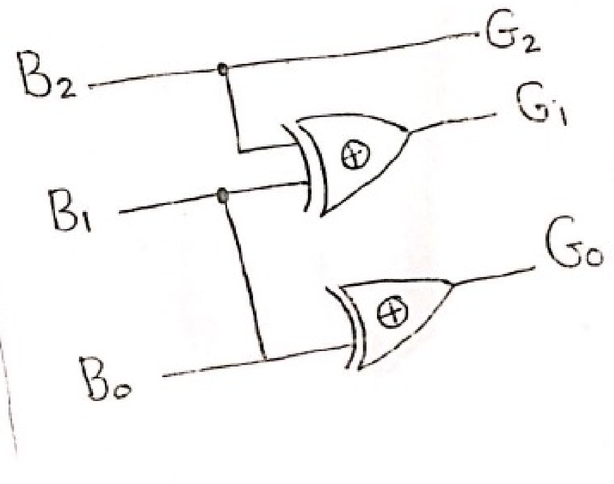
∴ # of inputs = 3 + # of outputs = 3 ⇒ 

Note In general, # of outputs required don't depend on # of inputs

[2] truth table.

B_2	B_1	B_0	G_2	G_1	G_0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

[4] Circuit diagram



[3] Find boolean expression for every output [every output has a boolean expression]

$G_2 = B_2$

B_0 \ B_2B_1	00	01	11	10
0	0	1	0	1
1	0	1	0	1

$G_1 = \overline{B}_2 B_1 + B_2\overline{B}_1$

B_0 \ B_2B_1	00	01	11	10
0	0	1	1	0
1	1	0	0	1

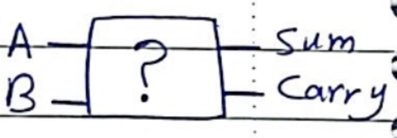
$G_0 = \overline{B}_1 B_0 + B_1\overline{B}_0$

Scanned with CamScanner

1 Half Adder

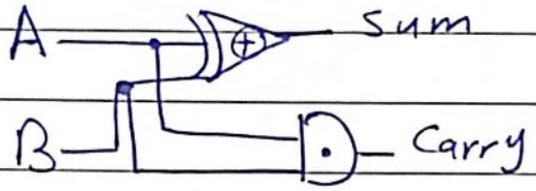
Design a circuit to add two binary bits

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



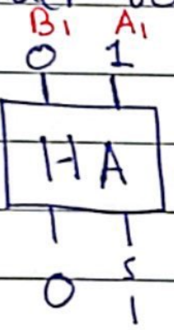
$Sum = A \oplus B$

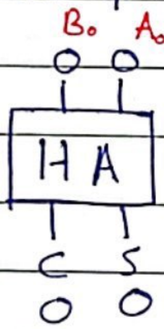
$Carry = A \cdot B$

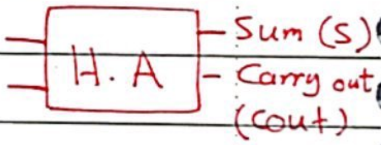


use Half adder to add 10 plus 0

$$\begin{array}{r} A\rightarrow 1\ 0 \\ B\rightarrow 0\ 0\ + \\ \hline 1\ 0 \end{array}$$



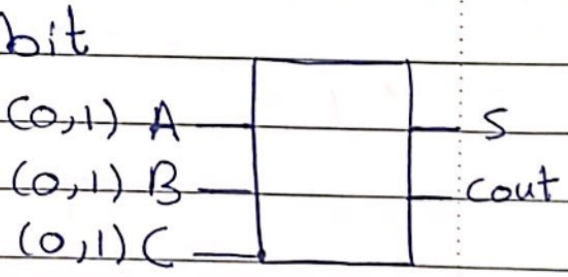


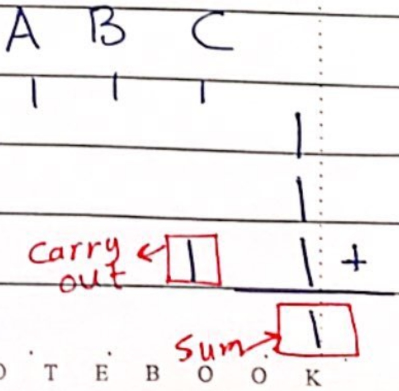


2 Full Adder

Adds three values each 1 bit

A	B	C	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1





Scanned with CamScanner

AB / C	00	01	11	10
0	0	1	0	1
1	1	0	1	0

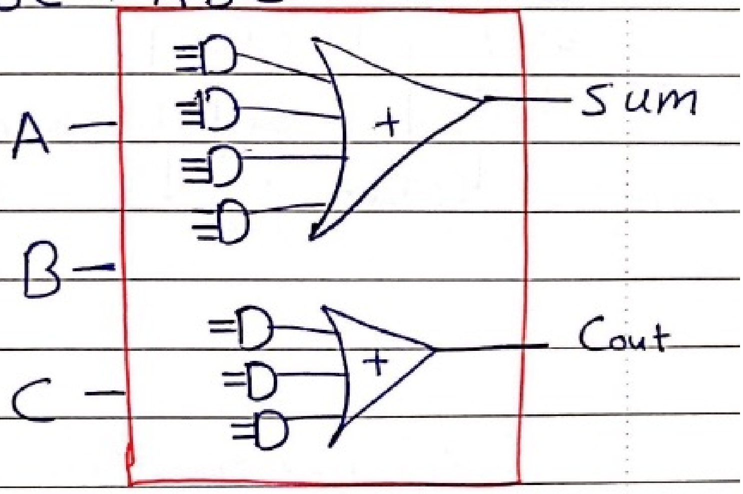
$S = \bar{A}\bar{B}C$

$$S = \bar{A}\bar{B}C + \bar{A}B\bar{C} + ABC + A\bar{B}\bar{C} = A \oplus B \oplus C$$

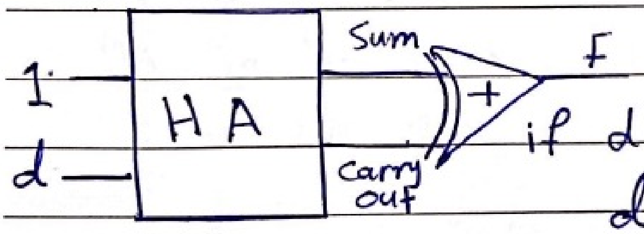
AB / C	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$BC + AB + AC$

$cout = BC + AB + AC$



full adder



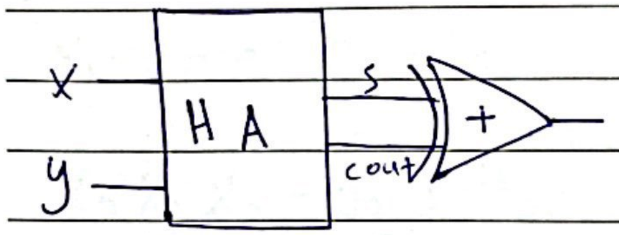
d: don't care

$$\text{if } d=0 \Rightarrow (1+0 = \overset{s}{1}\ \overset{cout}{0}) \Rightarrow f=1$$

$$d=1 \Rightarrow (1+1 = 0\ 1) \Rightarrow f=1$$

$$\boxed{f=1}$$

Scanned with CamScanner



Find f?

$F = M_0 = (x+y)$

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

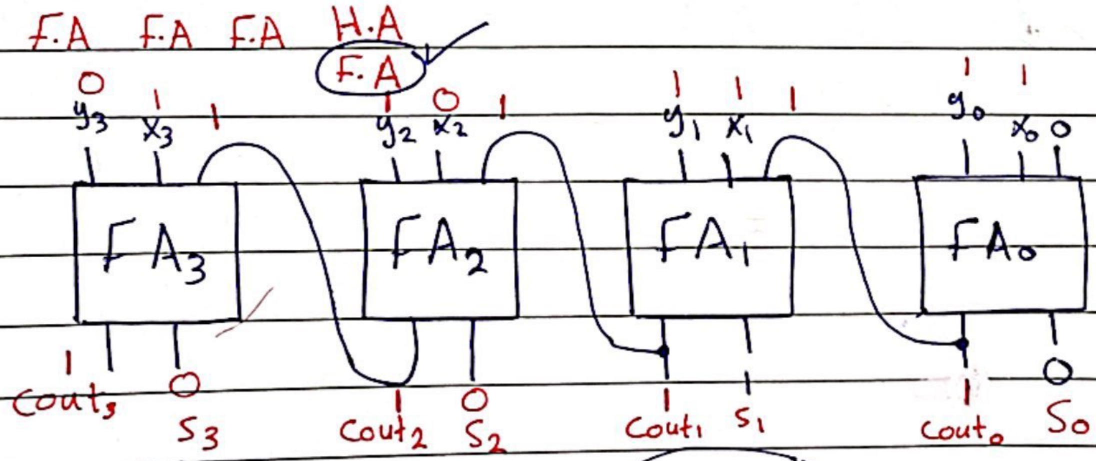
$F = m_1 + m_2 + m_3$
$= (\bar{x}y) + (x\bar{y}) + (xy)$

Ex: Use Full adder(s) to implement x+y where x, y are 4 bits, x = 1011 and y = 0111

$\square \quad \square \quad \square$
$x_3 \quad x_2 \quad x_1 \quad x_0$
$y_3 \quad y_2 \quad y_1 \quad y_0 \; +$
F.A F.A F.A H.A
F.A

option 1 : 1 H.A, 3FA

option 2 : 4 F.A



X = 1011
y = 0111

1 1 1 → cout
0 1 1
$\boxed{1}$ 0 1 1 1 +
0 0 1 0

Scanned with CamScanner

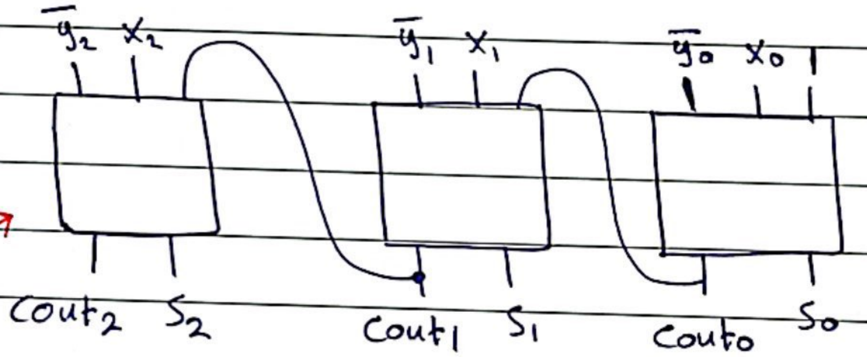
Ex: Use full adders to implement $x - y$ where x, y are 3 bits

$$\begin{array}{ccc} x_2 & x_1 & x_0 \\ \overline{y_2} & \overline{y_1} & \overline{y_0} + \\ & & 1 \\ \hline \boxed{F.A} & \boxed{F.A} & \boxed{F.A} \end{array}$$

$X_2 X_1 X_0 - y_2 y_1 y_0$

$x_2 x_1 x_0 - 2's(y_2 y_1 y_0)$

$X_2 X_1 X_0 + [y_2 + \overline{y_1} + \overline{y_0} + 1]$



Selector
→ 0 ⇒ +
→ 1 ⇒ −

✲ Full adder as subtractor

F.A Addition

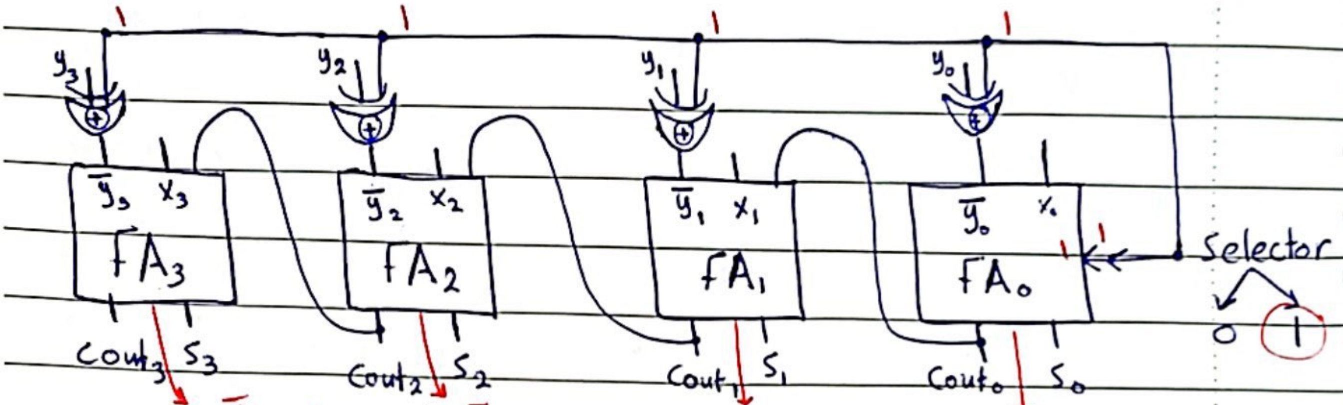
$$\begin{array}{ccc} x & x & x \\ y & y & y + \\ & & 0 \\ \hline \end{array}$$

F.A subtraction

$$\begin{array}{ccc} x & x & x \\ \overline{y} & \overline{y} & \overline{y} + \\ & & 1 \\ \hline \end{array}$$

Scanned with CamScanner

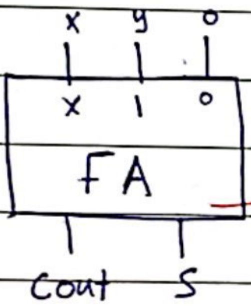
Design 4 bit Adder/subtraction



$x_3+\bar{y}_3+Cot_2$ $x_2+\bar{y}_2+cout_1$ $x_1+\bar{y}_1+Cot_0$ $1+x_0+\bar{y}_0 = x_0-y_0$

$y_0 \oplus 1 = \bar{y}_0$ $y_0 \oplus 0 = y_0$

$y = \bar{y}_0$ then subtraction $y = y_0$ then addition

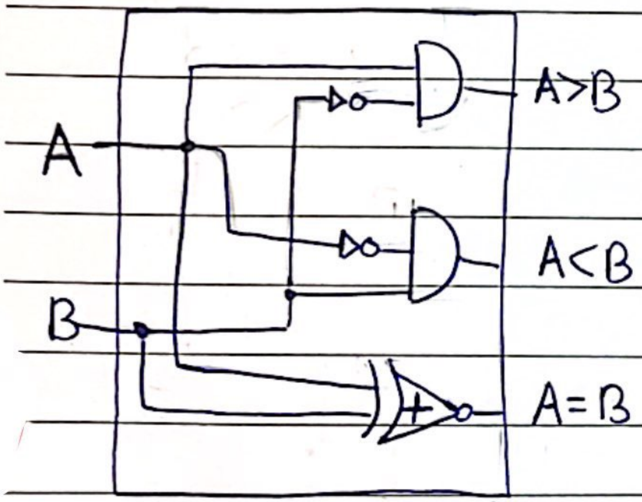


$x+y+z \Rightarrow x+1$

→ incrementer

$x+1+0 = x+1 \Rightarrow x{+}{+}$

1-Bit Magnitude comparator



A>B

A=1 و B=0

$A \cdot \bar{B}$

A<B

A=0 و B=1

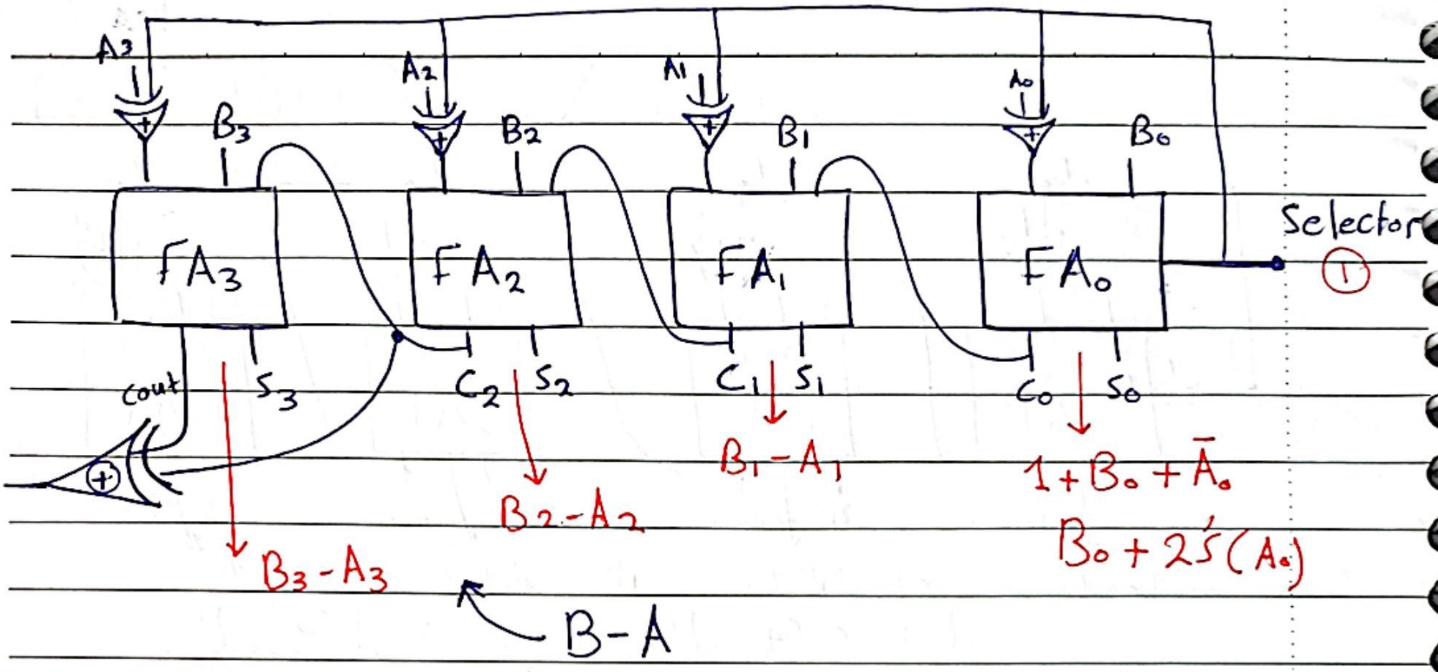
$\bar{A} \cdot B$

A=B

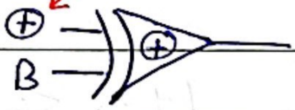
(A=1 و B=1) أو (A=0 و B=0)

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

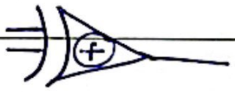
Scanned with CamScanner



✡ A + B → No need for ⊕ selector

✡ A - B → connect B's to 

✡ B - A → Connect

✡ B - A → Connet A's to 

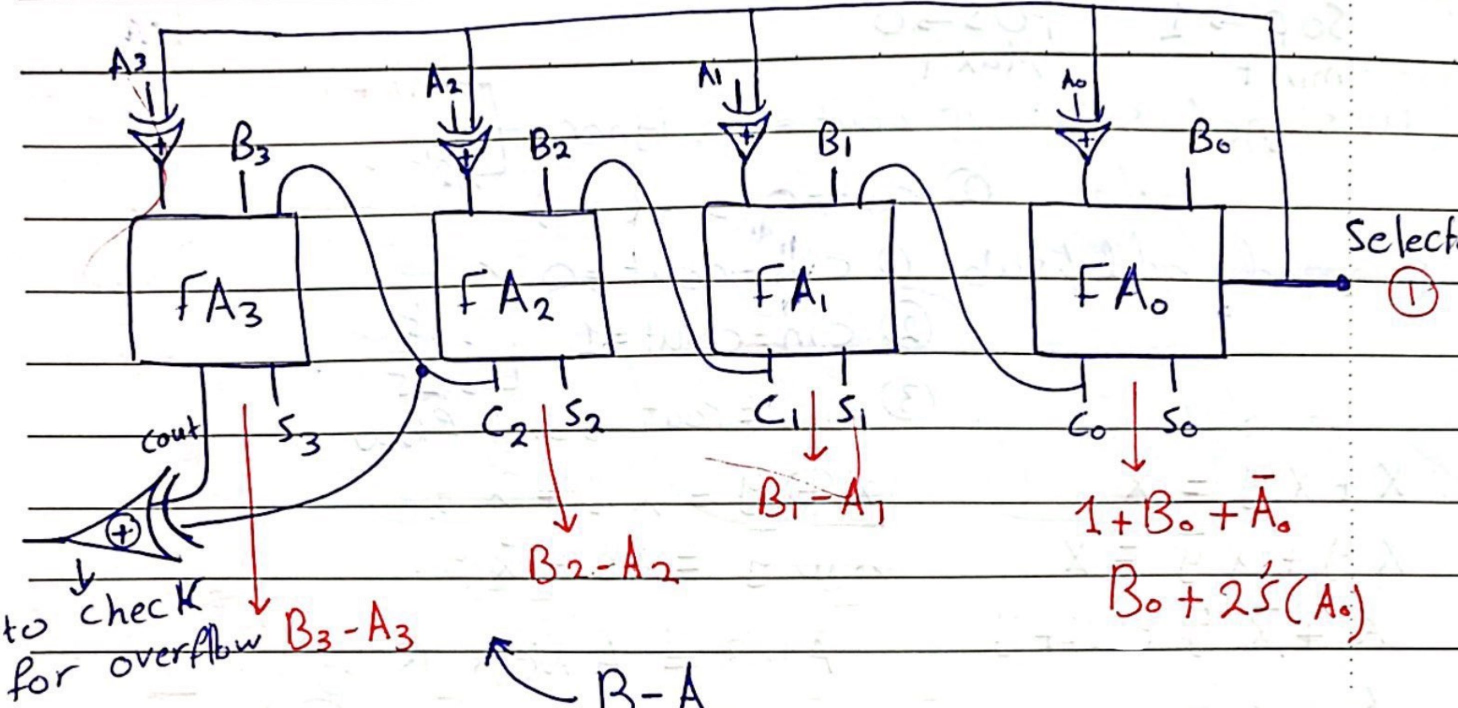
✡ -A - B → 8 full Adder

$-A-B = 2's(A) + 2's(B)$

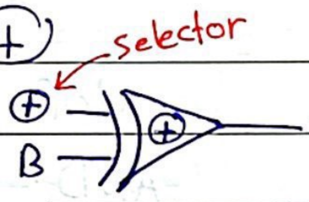
$\boxed{(\bar{A}+1) + (\bar{B}+1)} \Rightarrow F.A$

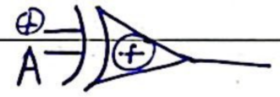
N O T E B O O K

Scanned with CamScanner



✡ $A+B \rightarrow$ No need for ⊕

✡ $A-B \rightarrow$ connect B's to 

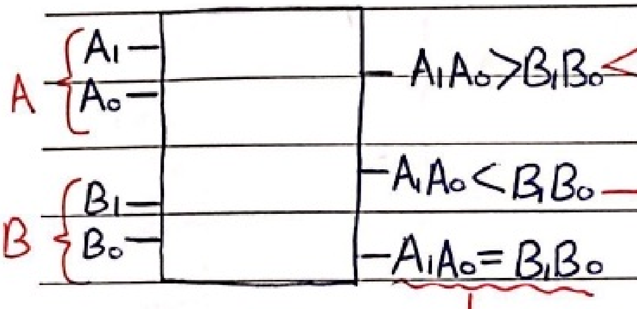
✡ $B-A \rightarrow$ Connet A's to 

✡ $-A-B \rightarrow$ 8 full Adder

$-A-B = 2's(A) + 2's(B)$

$\boxed{(\bar{A}+1)+(\bar{B}+1)} \Rightarrow F.A$

2 bit Magnitude Comparator



case1: $A_1>B_1$ $(A_1=1 , B_1=0)$

Case 2: $(A_1=B_1)$ & $(A_0>B_0)$

$$\boxed{(A_1\cdot\overline{B_1})+(\underbrace{(\overline{A_1\oplus B_1})}_{x_1}\cdot(A_0\cdot\overline{B_0}))}$$

Case 1: $A_1<B_1 \Rightarrow \overline{A}_1\cdot B_1$

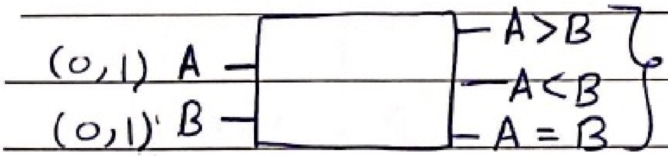
Case 2: $(A_1=B_1)$ & $(A_0<B_0)$

$$\boxed{(\underbrace{\overline{A_1\oplus B_1}}_{x_1})\cdot(\overline{A}_0\cdot B_0)) + \overline{A}_1 B_1}$$

$A_1=B_1$ & $A_0=B_0$

$$(\overline{A_1\oplus B_1})\cdot(\overline{A_0\oplus B_0})$$

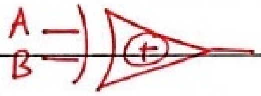
* Magnitude comparator:



only one will be active (has the value 1)

by expression meaning (without KMAP, without truth table)

$A>B \Rightarrow A=1$ & $B=0 \Rightarrow (A\cdot\overline{B})$



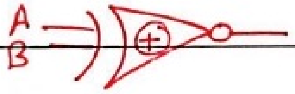
Not equal

≠ , !=

A	B	F	
0	0	0	No
0	1	1	yes
1	0	1	yes
1	1	0	No

m_1+m_2

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



equal

(==)

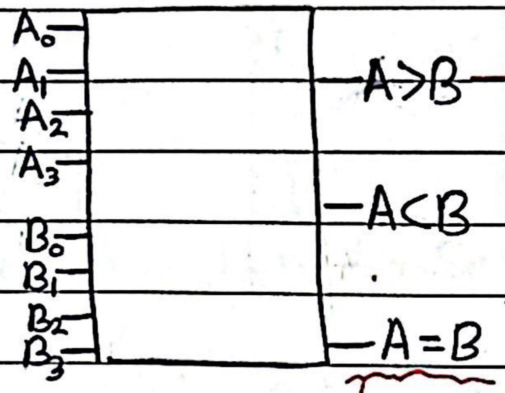
A	B	F	
0	0	1	yes
0	1	0	No
1	0	0	No
1	1	1	yes

m_0+m_3

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

Scanned with CamScanner

4 bit Magnitude Comparator



A>B →

→ Case 1: $A_3>B_3$ ($A_0 \sim A_2$, $B_0 \sim B_2$ don't care)

→ Case 2: $A_3=B_3$ & $A_2>B_2$

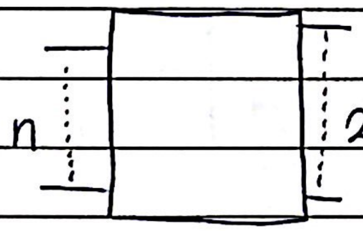
→ Case 3: $A_3=B_3$ & $A_2=B_2$ & $A_1>B_1$

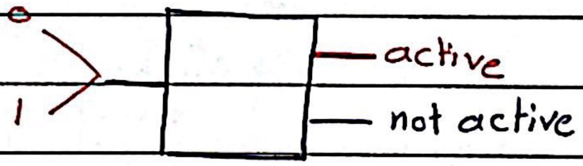
→ Case 4: $A_3=B_3$ & $A_2=B_2$ & $A_1=B_1$ & $A_0>B_0$

A=B → $A_3=B_3$ & $A_2=B_2$ & $A_1=B_1$ & $A_0=B_0=1$

$$(\overline{A_3 \oplus B_3}) \cdot (\overline{A_2 \oplus B_2}) \cdot (\overline{A_1 \oplus B_1}) \cdot (\overline{A_0 \oplus B_0})$$

☆ decoder(s)





1x2 dec

2x4 dec

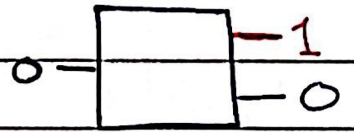
3x8 dec

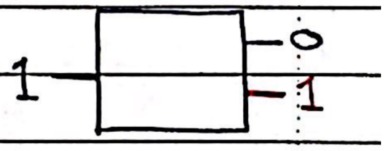
☆ Activate one output line only choosen based on input

Active high ← → Active Low

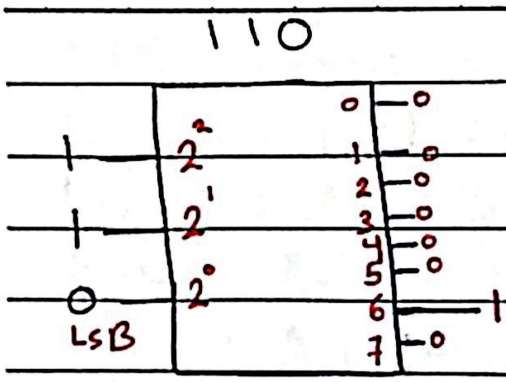
H L

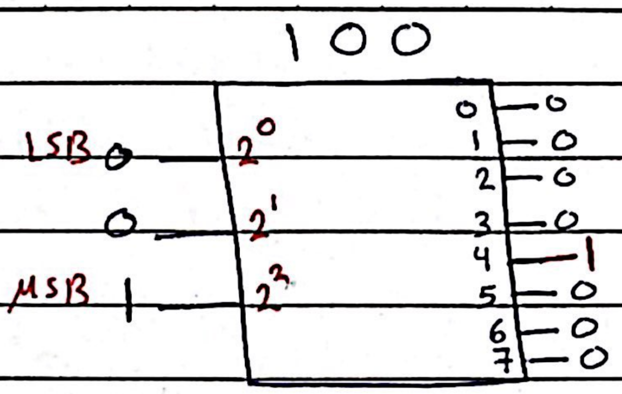
1 0





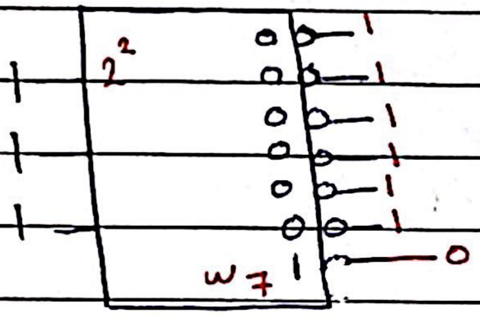
Scanned with CamScanner





3x8 active high decoder

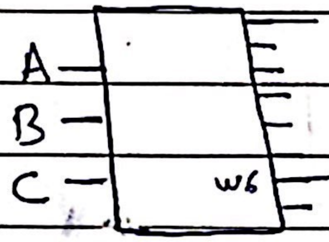
3x8 active high
↳ select and activate wire 4 from output



111 → W_7 will be active

3x8 active Low

$w_0 = 1$ if $A=0$ & $B=0$ & $C=0$ $(\bar{A}\cdot\bar{B}\cdot\bar{C})$

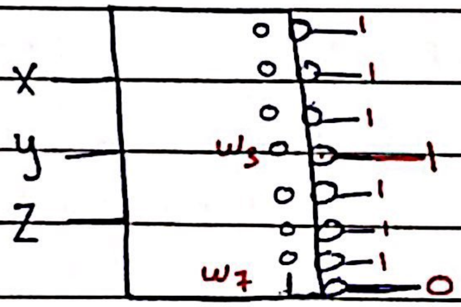


$w_6(A,B,C) = A\cdot B\cdot \bar{C}$

$w_3(X,y,Z) = \overline{(\bar{X}\cdot y\cdot Z)} \leftarrow m_3$

0 1 1

$= X + \bar{y} + \bar{Z} \leftarrow M_3$



$w_7(X,y,Z) = \bar{X} + \bar{y} + \bar{Z}$

1 1 1

Scanned with CamScanner

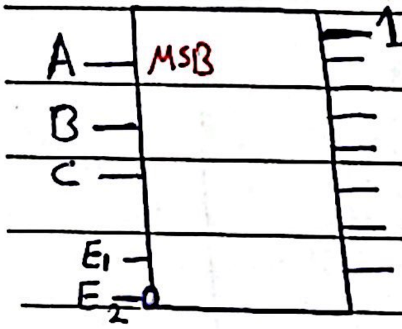
✡ Decoders:

type 1 : A.H (Active high)

type 2 : A.L (Active Low)

type 3 : A.H with enable(s)

type 4 : A.L with enable(s)



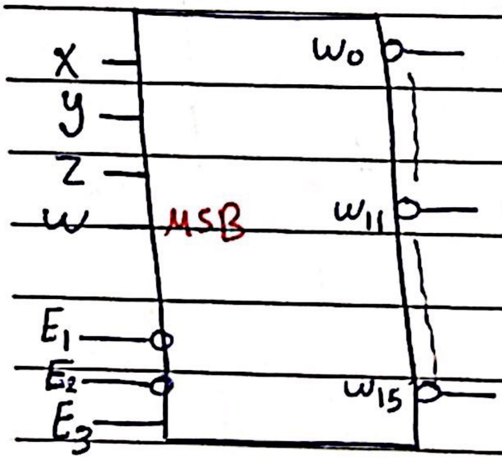
Dec is ON if $E_1 = 1$ & $E_2 = 0$

Dec is off if $E_1 = 0$ OR $E_2 = 1$

→ Any change on ABC will not produce output

3x8 active high ✡ Dec on & $A=0$ & $B=0$ & $C=0$

$$\boxed{(E_1=1 \;\&\; E_2=0)\cdot(A=0 \;\&\; B=0 \;\&\; C=0)}$$



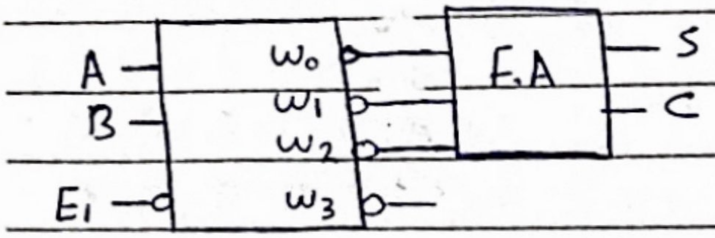
$= 0 \quad (\bar{E}_1\cdot\bar{E}_2\cdot E_3)\cdot(\overline{w.\bar{z}.y.x})$ POS

w	z	y	x
1	0	1	1

$(\bar{E}_1\cdot\bar{E}_2\cdot E_3)\cdot(\bar{w}+z+\bar{y}+\bar{x})$

4x16 active low with enables

Scanned with CamScanner



What will make $S = 0$?

S C

① $W_0 + W_1 + W_2 =$ 0 0 → not possible

$w_0 + w_1 + w_2 =$ 0 1

1 0 0

0 1 0

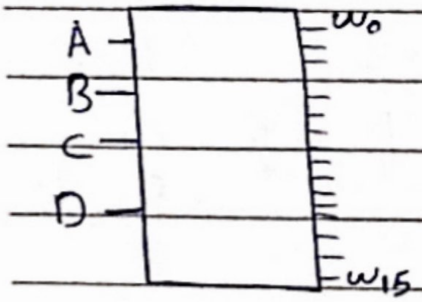
0 0 1

$E_1 = 0$

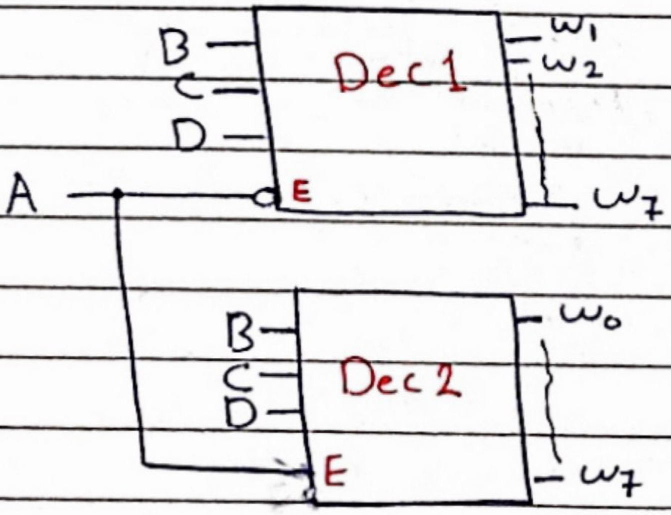
(Dec on) $f(\bar{A}\bar{B} + A\bar{B} + \bar{A}B)$

$S = (\bar{E}_1) \cdot (\bar{A}\bar{B} + A\bar{B} + \bar{A}B)$

Design 4x16 Dec by using 3x8 smaller dec (s)



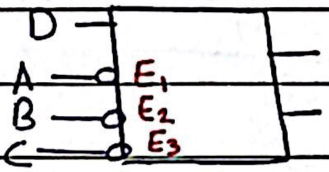
	A	B	C	D
	0	0	0	0
Dec1				
	0	1	1	1
	1	0	0	0
Dec2				
	1	1	1	1

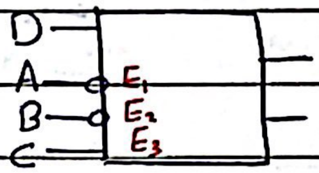


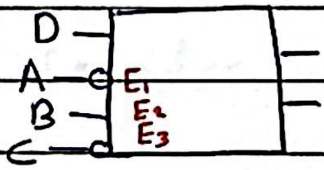
Scanned with CamScanner

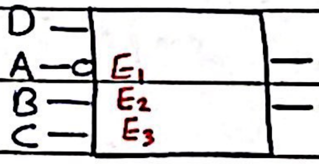
Design 4X16 Dec by using 1X2

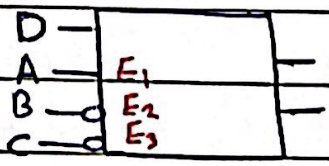
A	B	C	D	
0	0	0	0	Dec 1
0	0	0	1	
0	0	1	0	Dec 2
0	0	1	1	
0	1	0	0	Dec 3
0	1	0	1	
0	1	1	0	Dec 4
0	1	1	1	
1	0	0	0	Dec 5
1	0	0	1	
1	0	1	0	Dec 6
1	0	1	1	
1	1	0	0	Dec 7
1	1	0	1	
1	1	1	0	Dec 8
1	1	1	1	

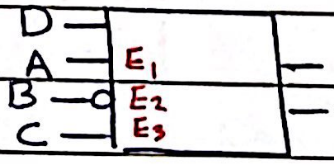


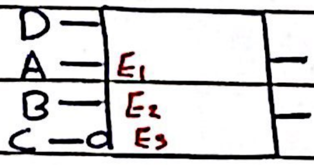


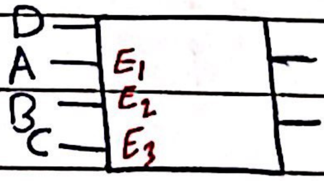








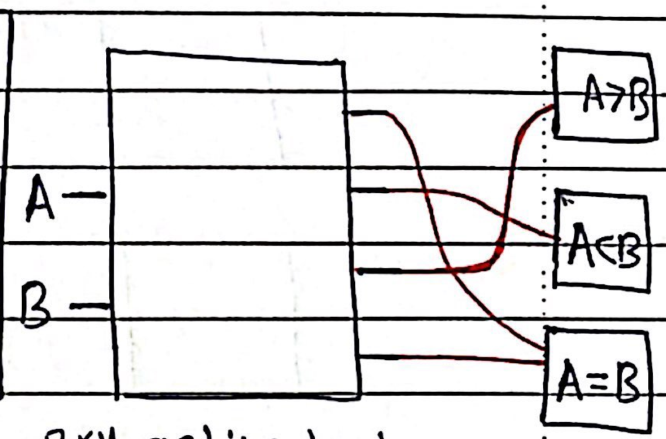




Scanned with CamScanner

Design 1 bit magnitude comparator using dec (s)

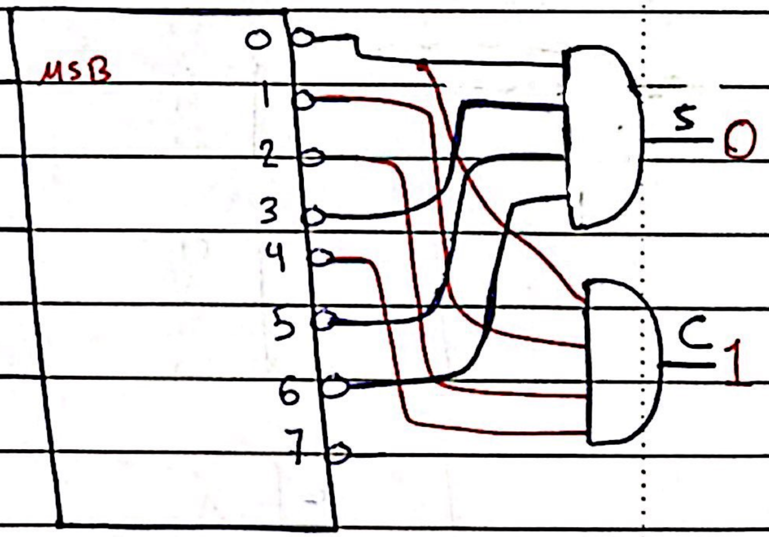
A	B	A >B	A <B	A =B
0	0	0	0	1 m_0
0	1	0	1 m_1	0
1	0	1 m_2	0	0
1	1	0	0	1 m_3



2x4 active high

Design full adder using A.L decoder implement POS

A	B	C	S	C
0	0	0	0✓	0✓
0	0	1	1	0✓
0	1	0	1	0✓
0	1	1	0✓	1
1	0	0	1	0✓
1	0	1	0✓	1
1	1	0	0✓	1
1	1	1	1	1

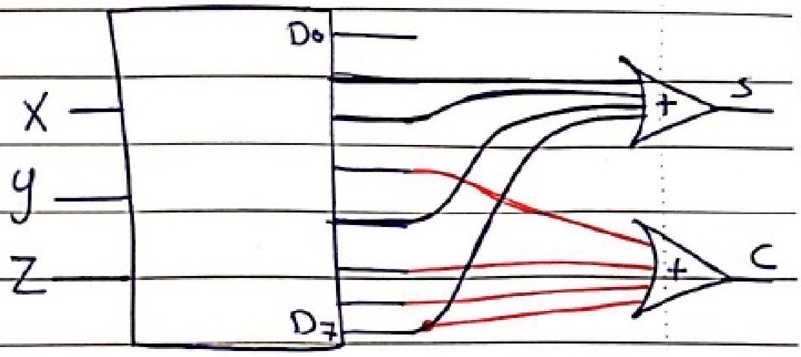


3x8 (AL)

Scanned with CamScanner

Design FA using AH decoder implement SOP

X Y Z	S C
0 0 0	0 0
0 0 1	1 0
0 1 0	1 0
0 1 1	0 1
1 0 0	1 0
1 0 1	0 1
1 1 0	0 1
1 1 1	1 1



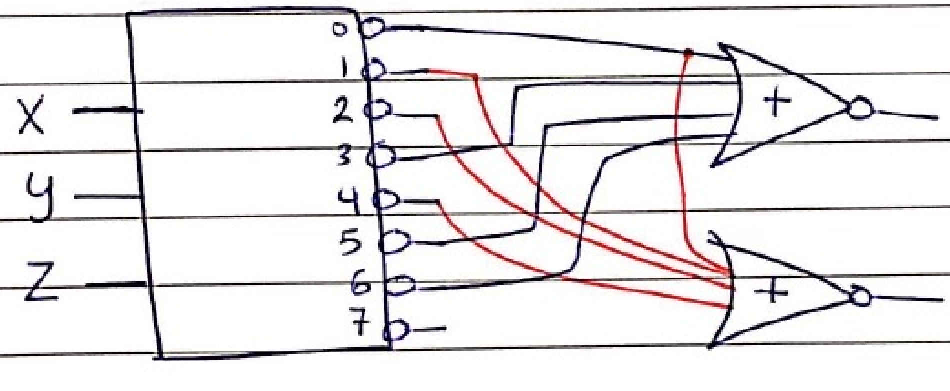
$S = \Sigma(1,2,4,7)$

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

Design FA using AL decoder implement SOP

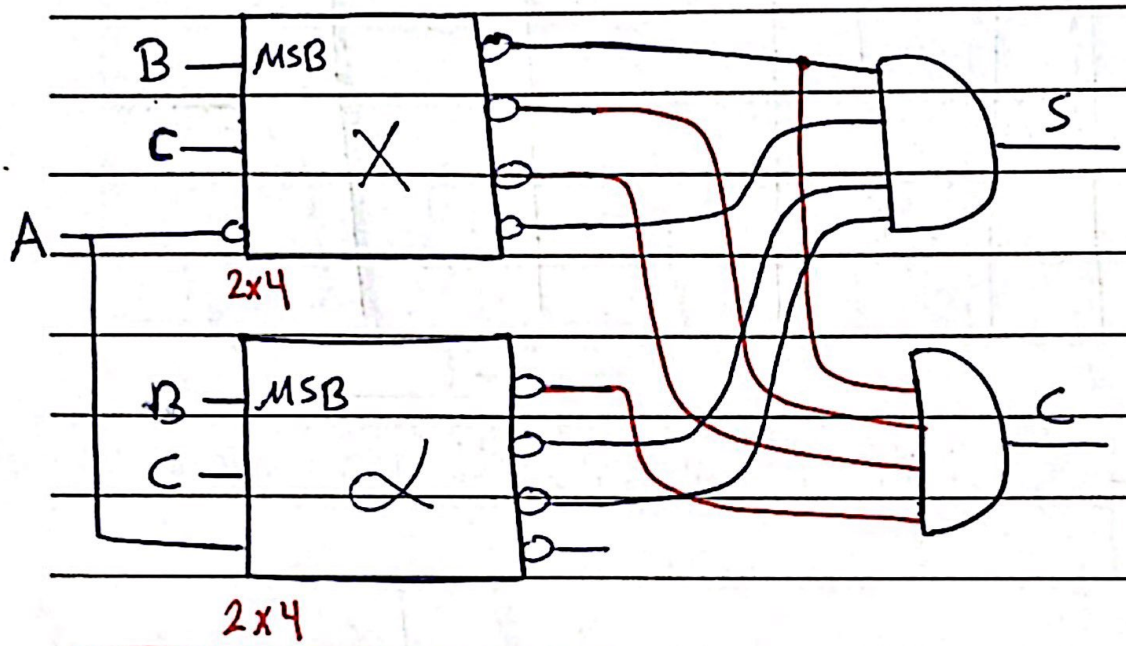
by using the same truth table: $S = \Sigma(1,2,4,7)$

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

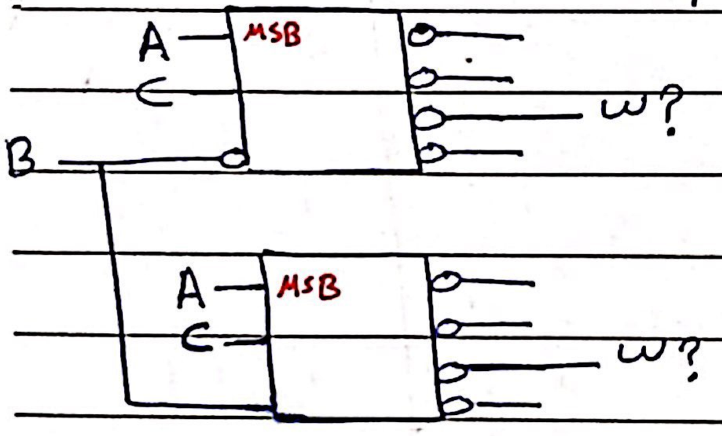


Scanned with CamScanner

Use 2X4 Dec (s) to implement full Adder (A.L) with POS



f (A, B, C)



A B C
1 0 0

A B C
1 1 0

Implementing a boolean expression using decode

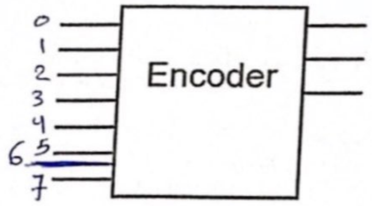
1- choose suitable size decoder

# of decoder inputs = # of function variables

2- You can use active high or active low decoder to implement the function as SOP or POS.

a- For SOP implementation
If you are using Active High decoder → Connect minterms to **OR** gate
If you are using Active Low decoder → Connect minterms to **NAND** gate

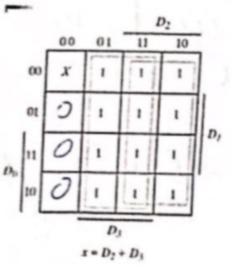
b- For POS implementation
If you are using Active High decoder → Connect maxterms to **NOR** gate
If you are using Active Low decoder → Connect maxterms to **AND** gate



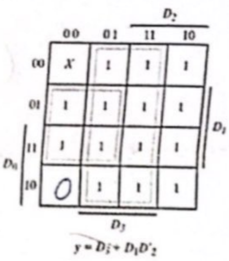
LSB							MSB			
D0	D1	D2	D3	D4	D5	D6	D7	X	Y	Z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

z=D1+D3+D5+D7
y=D2+D3+D6+D7
x=D4+D5+D6+D7

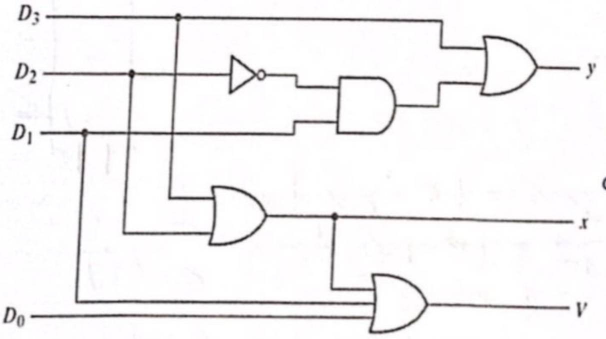
Inputs				Outputs		
D0	D1	D2	D3	x	y	v
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

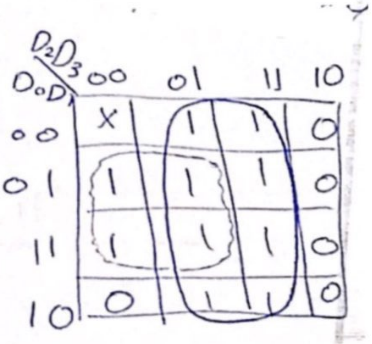


$x = D_2 + D_3$



$y = D_3 + D_1D'_2$



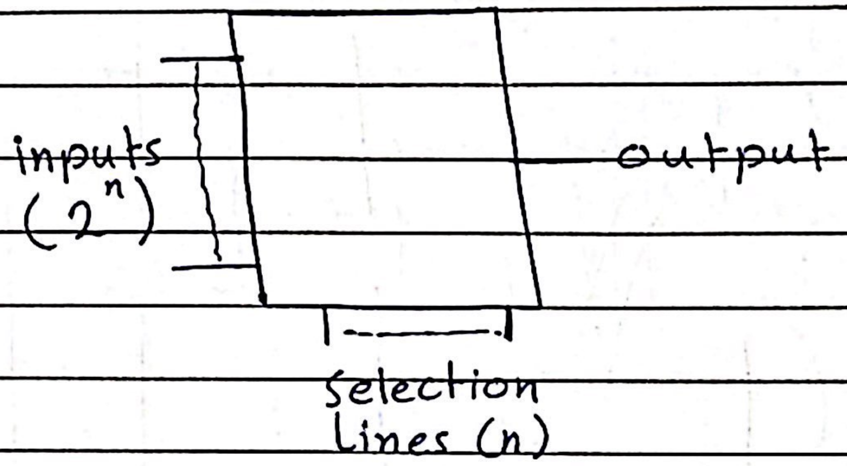


$y = D_3 + D_2' D_1$

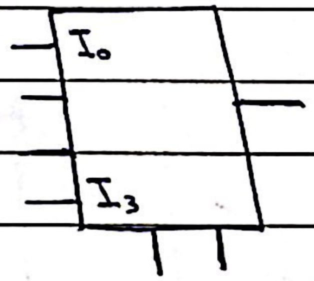
CS Scanned with CamScanner

Scanned with CamScanner

✡ Multiplexers: select one input and pass its value to output



2x1
4x1
8x1
16x1
32x1
⋮



0 0 → select $I_o \Rightarrow f = I_o$
0 1 → select $I_1 \Rightarrow f = I_1$
1 0 → select $I_2 \Rightarrow f = I_2$
1 1 → select $I_3 \Rightarrow f = I_3$

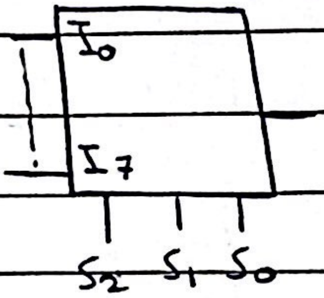
✡ If $S_1S_o = 00$ then $f = I_o$

$f = I_o$ when $S_1S_o = 00$	if $S_1S_o = 01$	if $S_1S_o = 10$
$\Rightarrow f = \bar{S}_1\bar{S}_o I_o$	$\bar{S}_1 S_o I_1$	$S_1\bar{S}_o I_2$

if $S_1S_o = 11$
$S_1 S_o I_3$

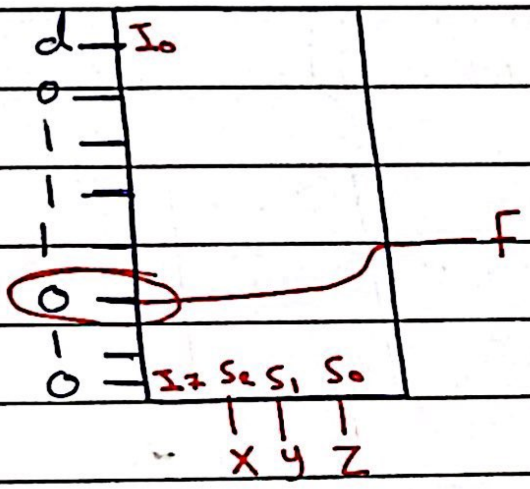
$$\boxed{f = I_o\bar{S}_1\bar{S}_o + I_1\bar{S}_1S_o + I_2S_1\bar{S}_o + I_3S_1S_o}$$

Scanned with CamScanner



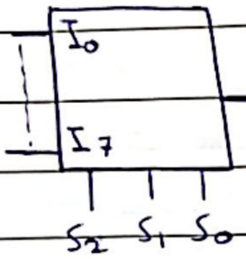
$$F = I_0 \bar{S}_2 \bar{S}_1 \bar{S}_0 + I_1 \bar{S}_2 \bar{S}_1 S_0 + I_2 \bar{S}_2 S_1 \bar{S}_0$$
$$+ I_3 \bar{S}_2 S_1 S_0 + I_4 S_2 \bar{S}_1 \bar{S}_0 + I_5 S_2 \bar{S}_1 S_0$$
$$+ I_6 S_2 S_1 \bar{S}_0 + I_7 S_2 S_1 S_0$$

Use the suitable Mux to implement
$F(X,Y,Z) = \Pi(1,5,7) + \Sigma d(0)$



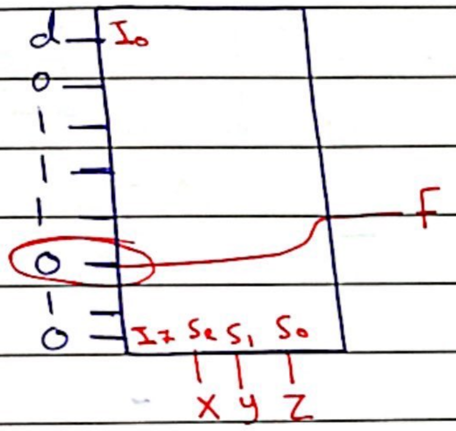
X	Y	Z	F
0	0	0	d
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Scanned with CamScanner



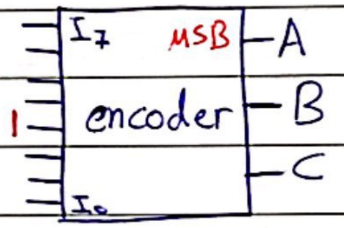
$$F = I_0\bar{S_2}\bar{S_1}\bar{S_0} + I_1\bar{S_2}\bar{S_1}S_0 + I_2\bar{S_2}S_1\bar{S_0} + I_3\bar{S_2}S_1S_0 + I_4S_2\bar{S_1}\bar{S_0} + I_5S_2\bar{S_1}S_0 + I_6S_2S_1\bar{S_0} + I_7S_2S_1S_0$$

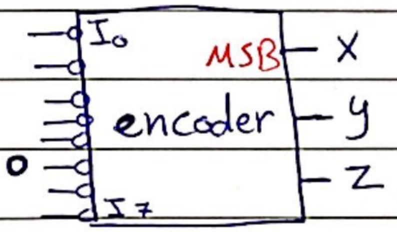
Use the suitable Mux to implement
$F(X,Y,Z) = \prod(1,5,7) + \sum d(0)$



X	Y	Z	F
0	0	0	d
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Determine what will be the output?

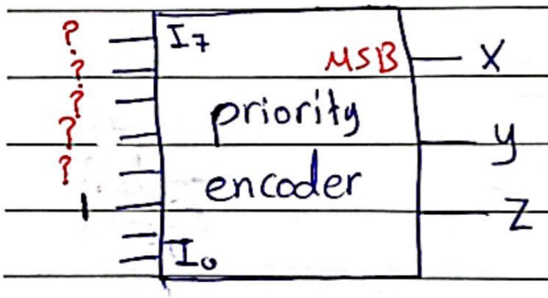




I_3 active	$I_5 = 0$ (Active)
$I_0 \sim I_2$ and $I_4 \sim I_7$ not active	$I_0 \sim I_4$ and $I_6 \sim I_7$ not active
ABC = 011	XYZ = 101

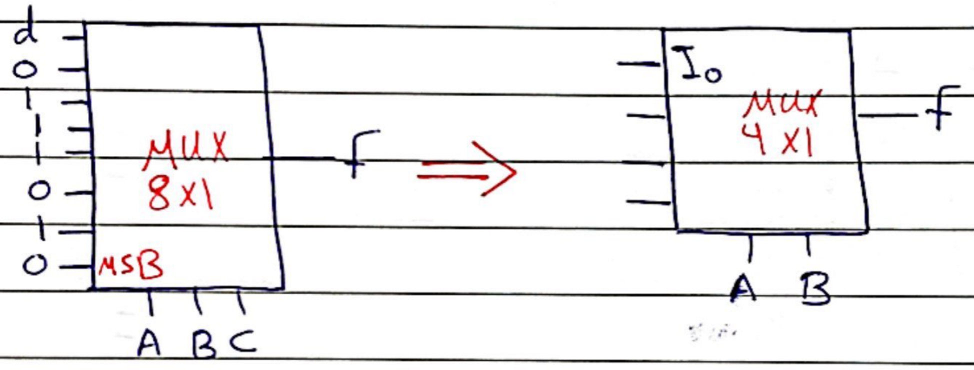
N O T E B O O K

Scanned with CamScanner



xyz couldn't be determined

$f(A,B,C) = \Pi(1,5,7) + \sum d(0)$



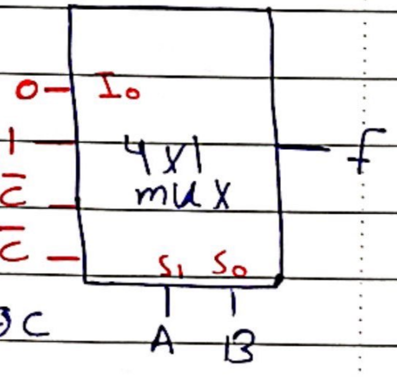
ABC	f
000	d
001	0
010	1
011	1
100	1
101	0
110	1
111	0

Group1 = 0, A, B
(I_0)

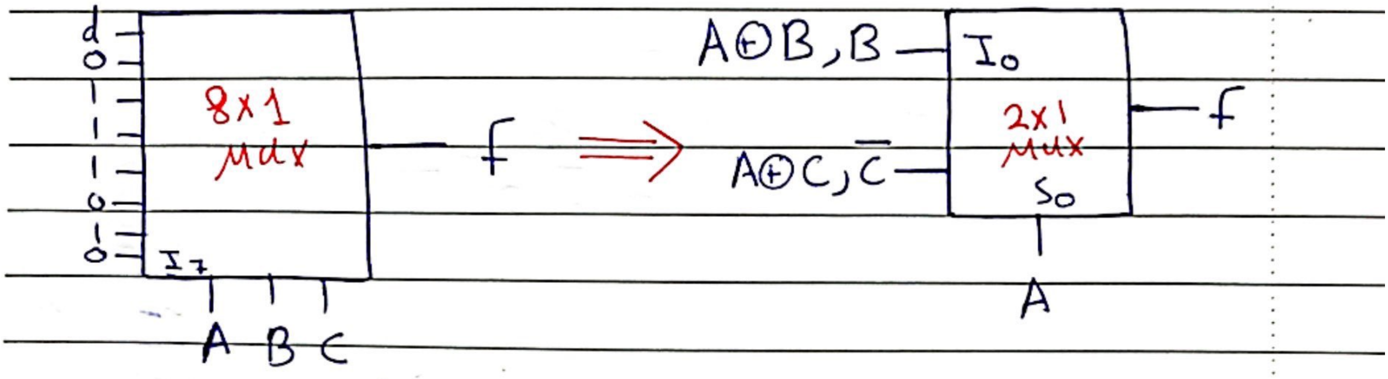
Group2 = 1, B
(I_1)

Group3 = $\bar{C}$, $A \oplus C$
(I_2)

Group4 = $\bar{C}$, $A \oplus C$, $B \oplus C$
(I_3)



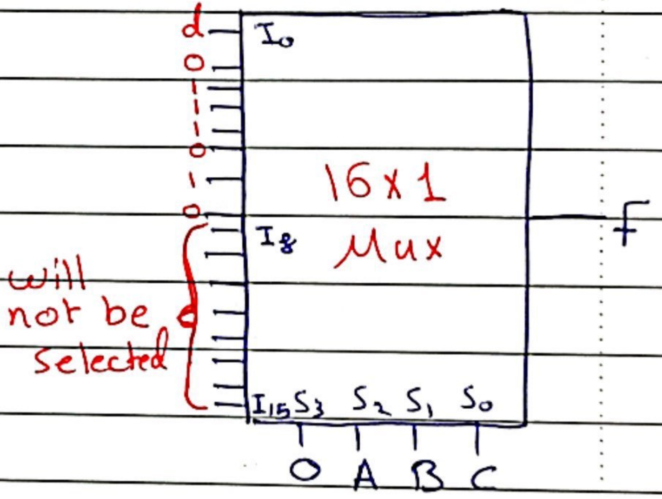
Scanned with CamScanner

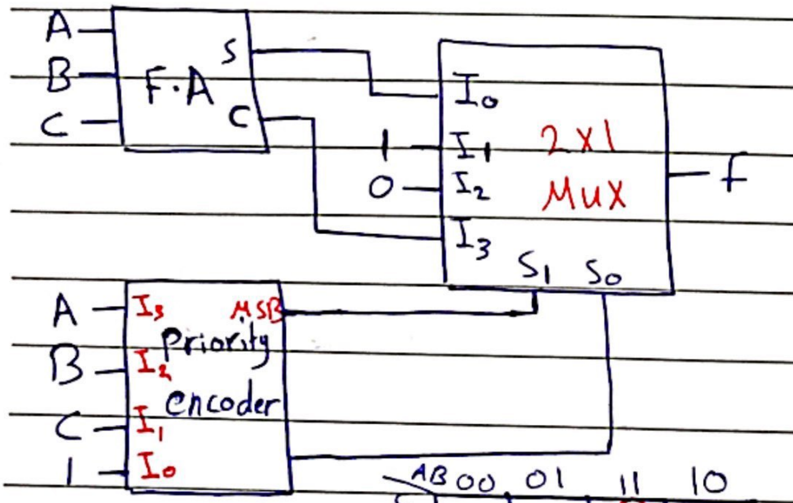


A B C	f
0 0 0	d
0 0 1	0
0 1 0	1
0 1 1	1
1 0 0	1
1 0 1	0
1 1 0	1
1 1 1	0

Group1 $= B, A \oplus B$

Group2 $= A \oplus C, \bar{C}$



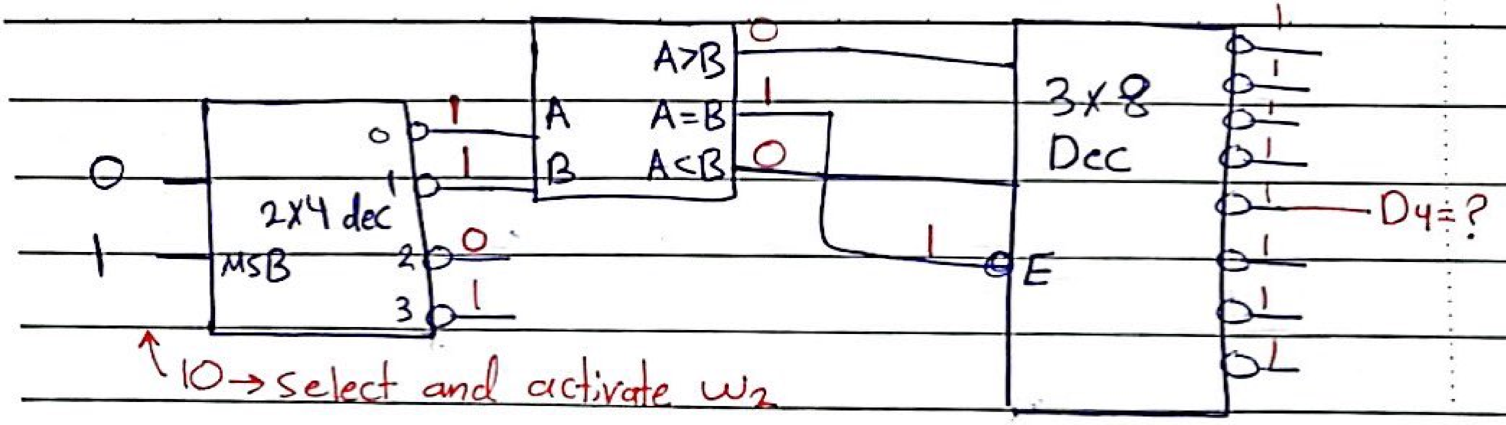


A B C	f
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	0
1 0 0	0
1 0 1	1
1 1 0	1
1 1 1	1

Carry

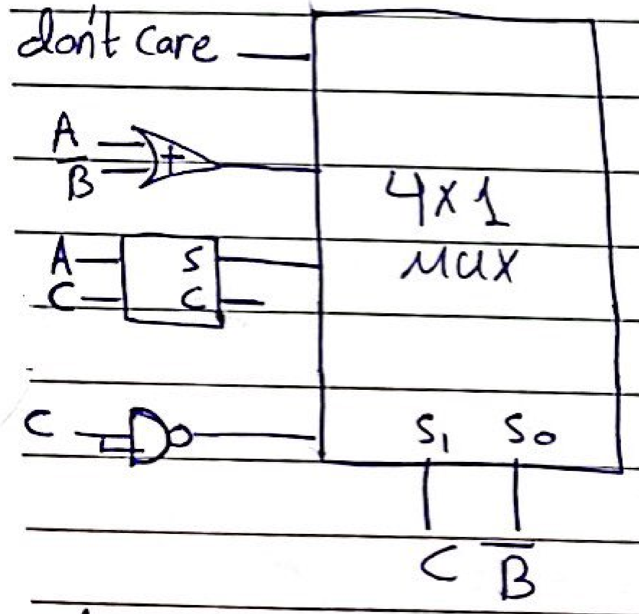
C \ AB	00	01	11	10
0	0	0	1	0
1	1	0	1	1

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



10 → select and activate w_2

Dec is off, D4 = 1



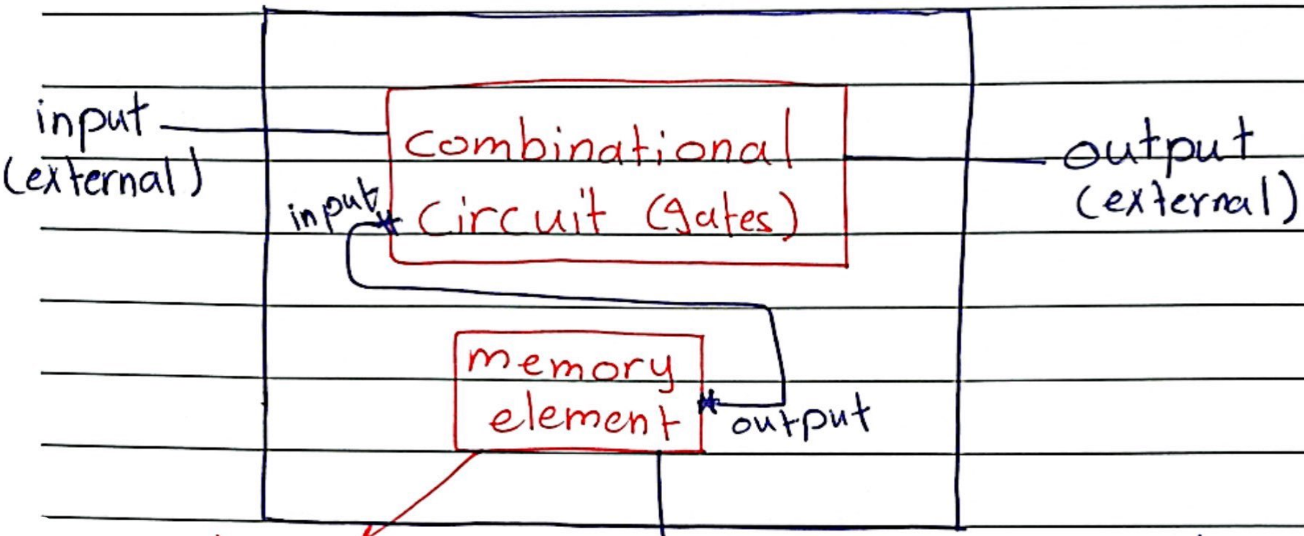
← find f(A,B,C)?

A B C	F
0 0 0	1
0 0 1	0
0 1 0	d
0 1 1	1
1 0 0	1
1 0 1	0
1 1 0	d
1 1 1	0

C \ AB	00	01	11	10
0	1	d	d	1
1	0	1	0	0

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

Sequential circuits:



state
every flip flop
and latch has
state (current output)

→ sequential circuit
built from flip flop or latch
store value
gate

✲ To find value of external output:

1) find value of all external ~~out~~ inputs
2) // // // // feed back(s)

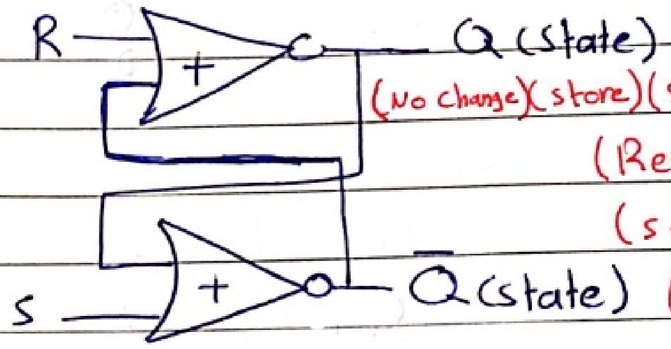
F(external inputs, feed back(s)) = - sequential
states)

✲ Sequential gates that can store values:

1) Latch
2) clocked latch
3) flip flop

$\boxed{SR}$ $\boxed{D}$ $\boxed{JK}$

$\boxed{T}$



SR-latch

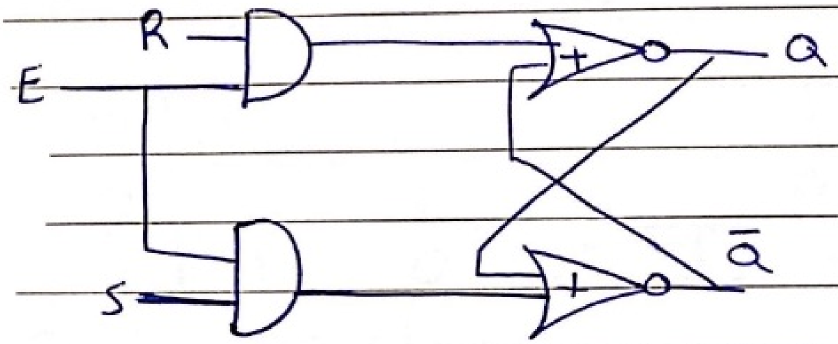
no change (store)

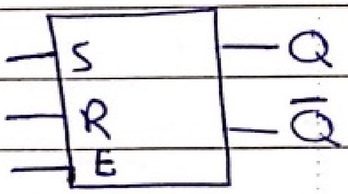
	S	R	$Q_{(t+1)}$	$\bar{Q}_{(t+1)}$
(No change)(store)(save) →	0	0	Q(t)	$\bar{Q}$(t)
(Reset) →	0	1	0	1
(set) →	1	0	1	0
(error) →	1	1	$\boxed{0}$	$\boxed{0}$

no change (store)

Not stable (error) (avoid)

Q(t): the value of Q now
Q(t+1): // // // Q next





Clocked SR-latch

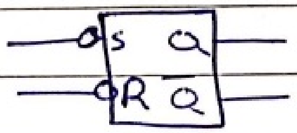
E	S	R	Q	$\bar{Q}$	
0	0	0	Q	$\bar{Q}$	E=0 No change
0	0	1	Q	$\bar{Q}$	
0	1	0	Q	$\bar{Q}$	
0	1	1	Q	$\bar{Q}$	
1	0	0	Q	$\bar{Q}$	→ No change
1	0	1	0	1	
1	1	0	1	0	
1	1	1	Error (Avoid)		

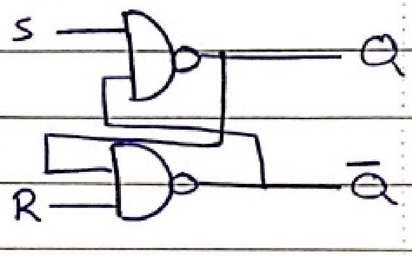
◆ If E=0 → No change
else (E=1)
clock SR-latch
acts as normal
SR-latch

Scanned with CamScanner

Active Low SR Latch

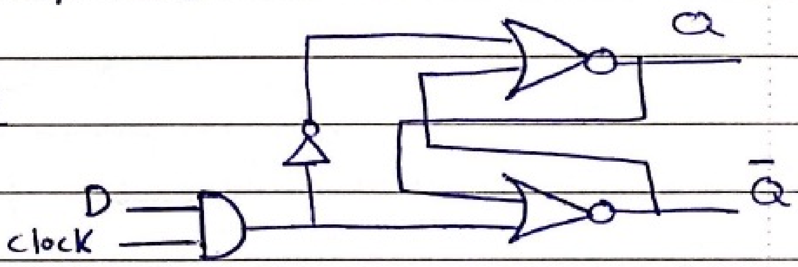
S	R	Q(t+1)	$\overline{Q(t+1)}$
0	0	1	1 → error
0	1	1	0 → set
1	0	0	1 → reset
1	1	Q(t)	$\overline{Q(t)}$ → no change

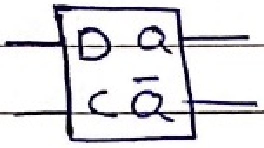




D- Latch (transparent)

C	D	Q(t+1)	$\overline{Q(t+1)}$
1	0	0	1
1	1	1	0
0	x	no change	

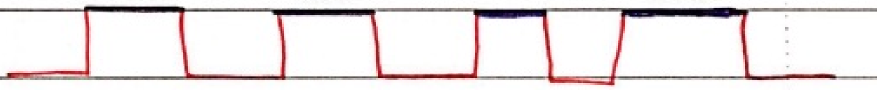




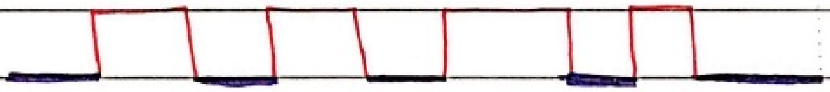
- Set : High / 1 / On
- Reset : Low / 0 / off

Latch / Asynchronous

Positive level:



negative level:



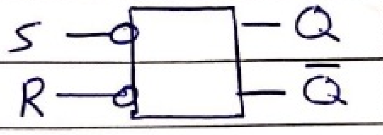
Scanned with CamScanner

Subject Sequential circuits Date No.

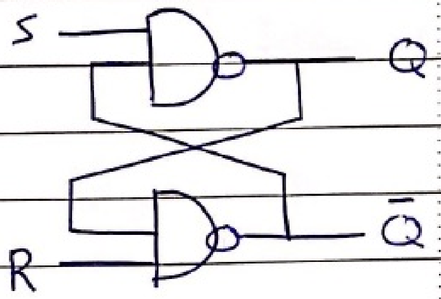
latches : pulse triggered
flip flops : edge triggered

Asynchronous (latches) : SR, D
Synchronous (flip flops) : D, JK, T

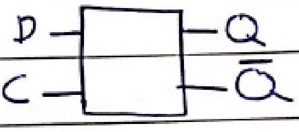
Active low SR latch:

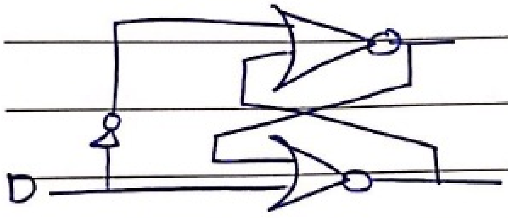


S R	Q $\bar{Q}$	
0 0	1 1	error
0 1	1 0	set
1 0	0 1	reset
1 1	Q $\bar{Q}$	No change



D latch:



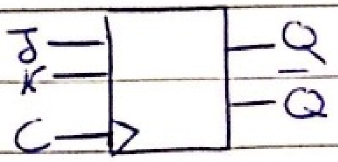


C D	Q $\bar{Q}$
0 0	0 1
0 1	1 0
1 x	no change

$Q = D$

Scanned with CamScanner

JK flip flop



J K	Q $\bar{Q}$	
0 0	Q $\bar{Q}$	→ no change
0 1	0 1	→ reset
1 0	1 0	→ set
1 1	$\bar{Q}$ Q	→ toggle / flip

J K Q	Q(t+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 \ JK	00	01	11	10
0	0	0	1	1
1	1	0	0	1

$$Q(t+1) = J\bar{Q} + \bar{K}Q$$

T flip flop:

T	Q	
0	Q	→ no change
1	$\bar{Q}$	→ toggle

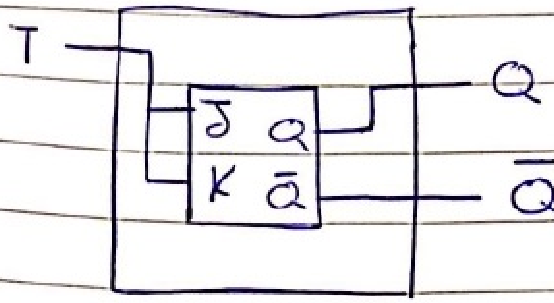
T Q	Q(t+1)
0 0	0
0 1	1
1 0	1
1 1	0

$$Q(t+1) = T \oplus Q(t)$$

Scanned with CamScanner

Subject Date No.

Ex: Design T from JK flip flop:



☆ Direct input:

1) Direct set (Preset): makes the output one
2) Direct reset (clear): makes the output zero

☆ Sequential circuits Analysis:

max # of rows = 2^{n+m}

n = # of inputs
m = # of flip flops

# of states = 2^m

Scanned with CamScanner

Sequential circuits & storage elements

Logic circuits

Combinational

Sequential

- circuits Require clock
- output is Function of current input & previous output.
- Contains memory elements to save output for short period.

divided onto

latches

- level sensitive clock
- If clock exists, the latch changes it output when clock (clK) = 1
- when clock = 0 output is not changed.

level sensitive

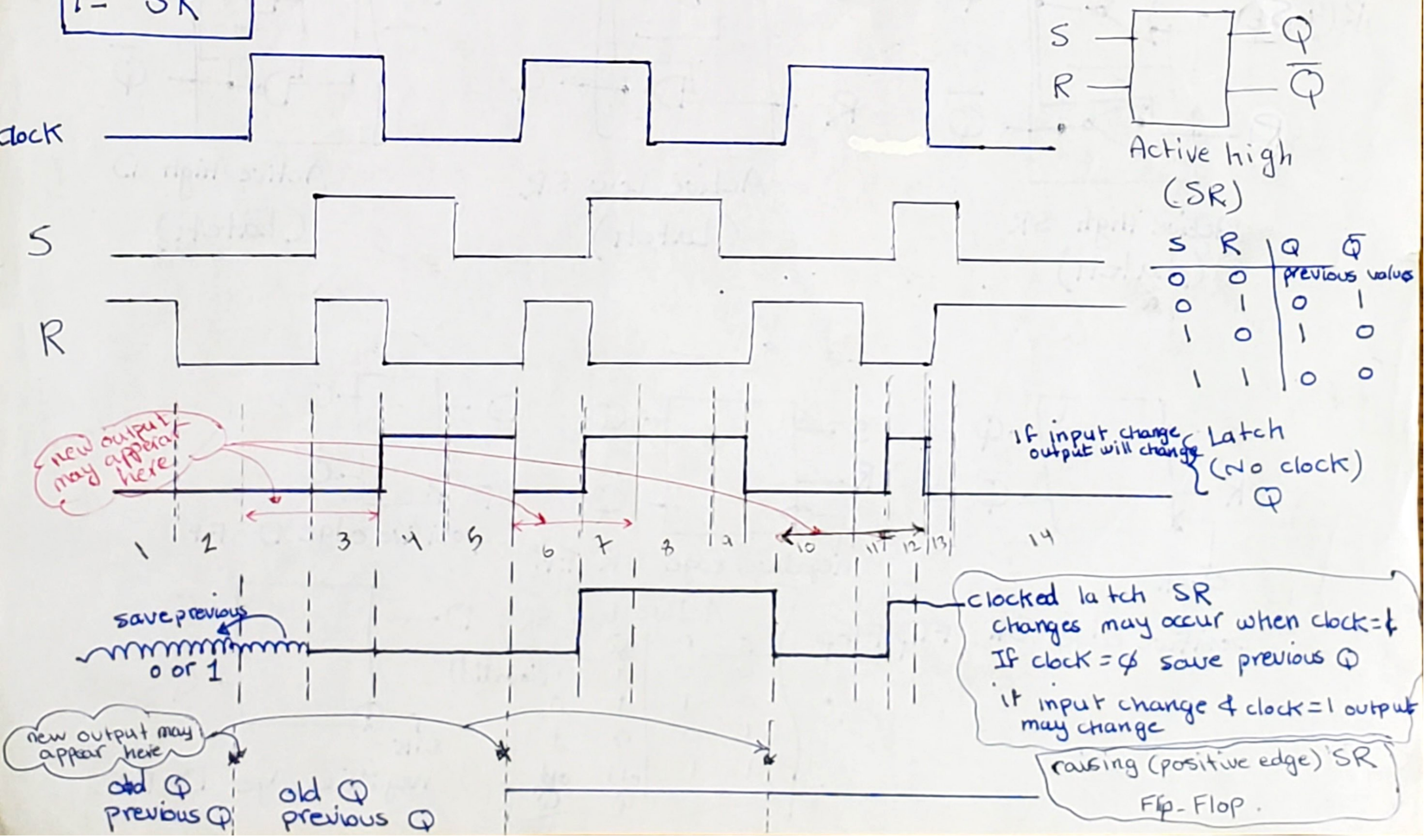
Flip-Flops

- edge sensitive
- Output changes either on falling (negative ↓) edge or on raising positive (↑) edge
- ~~when~~ Between ↓ or ↓ ↑ output is not changed.

edge sensitive

- Sequential gates (SR, D, JK, T)

1- SR



Active high (SR)

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

If input change output will change { Latch (No clock) Q

clocked latch SR
changes may occur when clock = 1
If clock = 0 save previous Q
if input change & clock = 1 output may change

raising (positive edge) SR Flip-Flop.

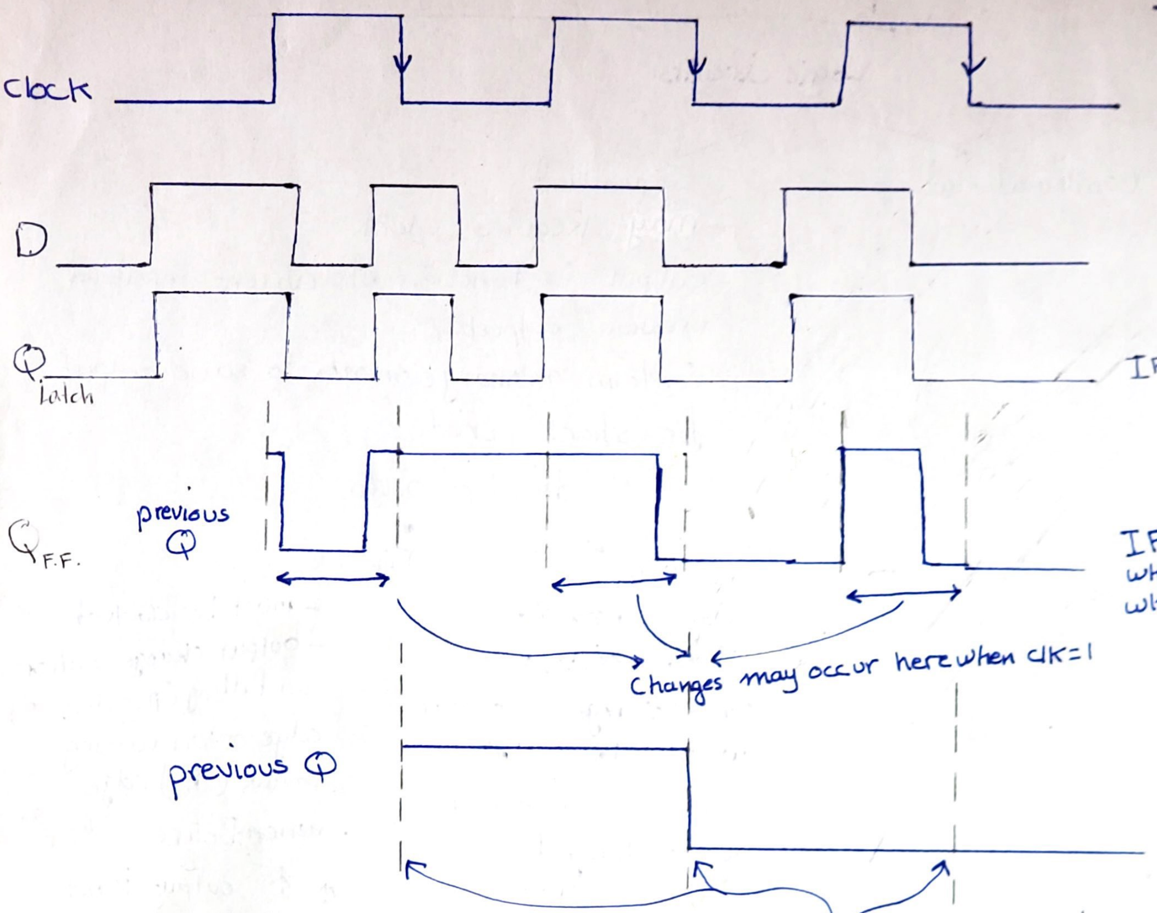
page2

D circuit

D	Q(t)
0	0
1	1

output Q(t)=D

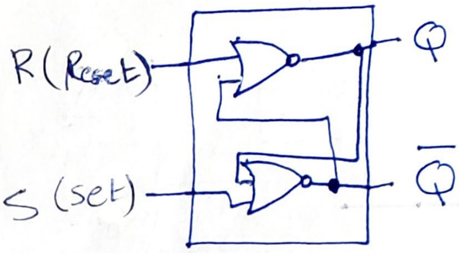
always Q(t)=D



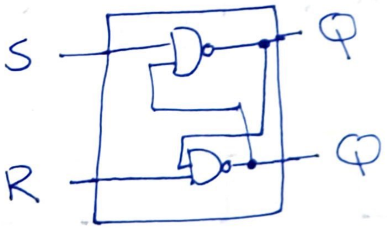
IF D is Latch with No Clock.

IF D is clocked latch
when clK = 1 output = D
when clK = 0 save previous output

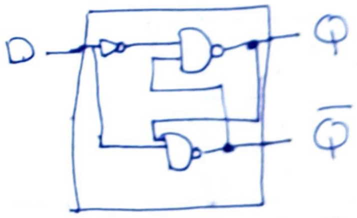
Changes on output may occure here (the change is saved till next Falling ↧)



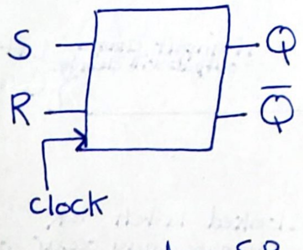
Active High SR (Latch)



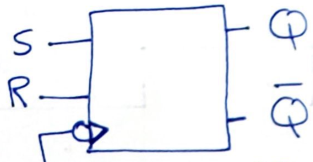
Active Low SR (Latch)



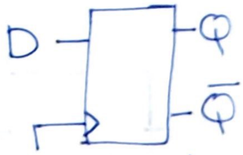
Active high D (Latch)



positive edge SR Flip Flop



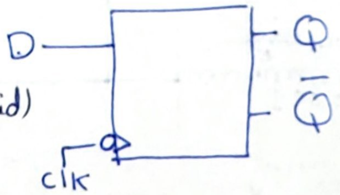
negative edge SR F.F.



positive edge D F.F

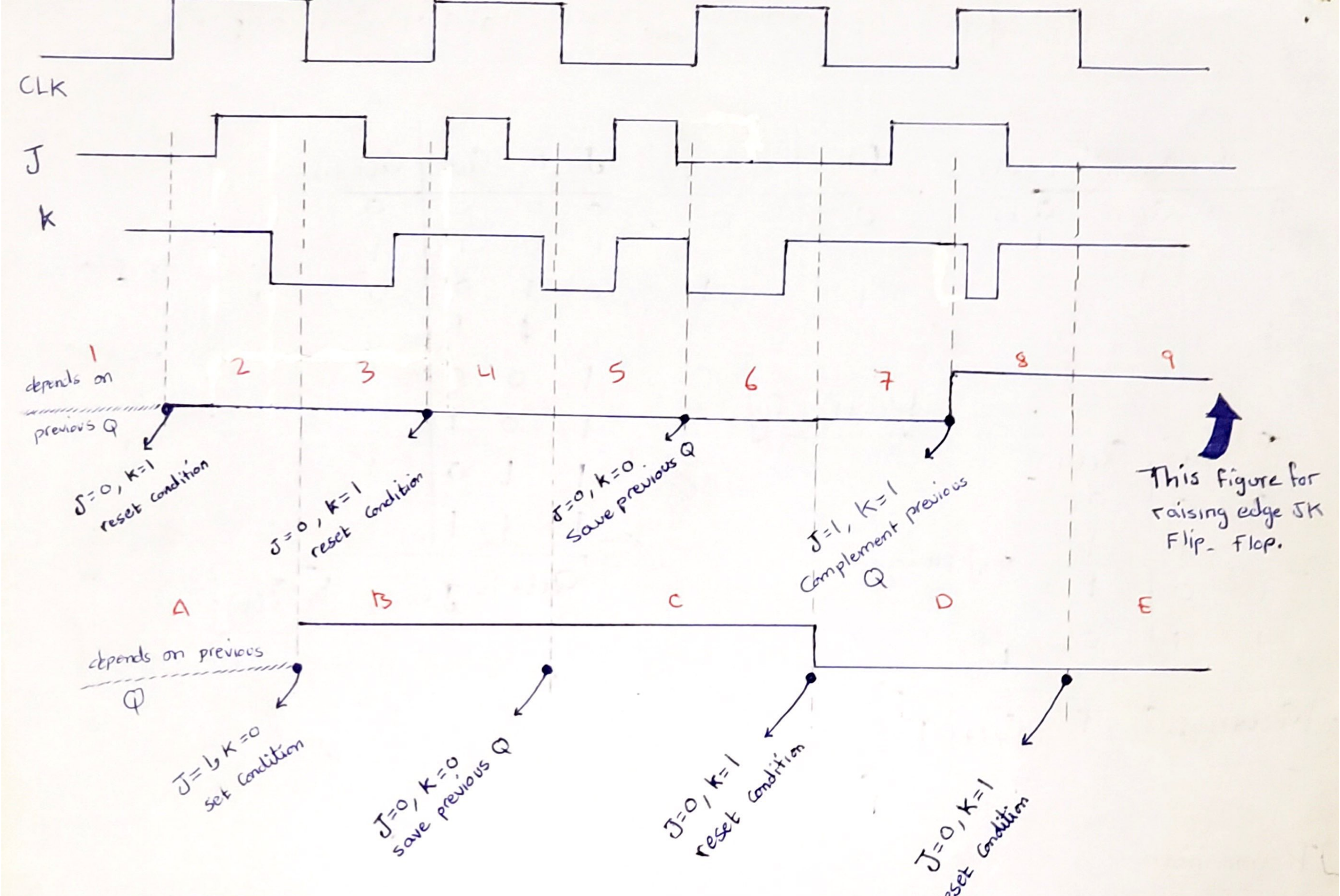
Active low SR

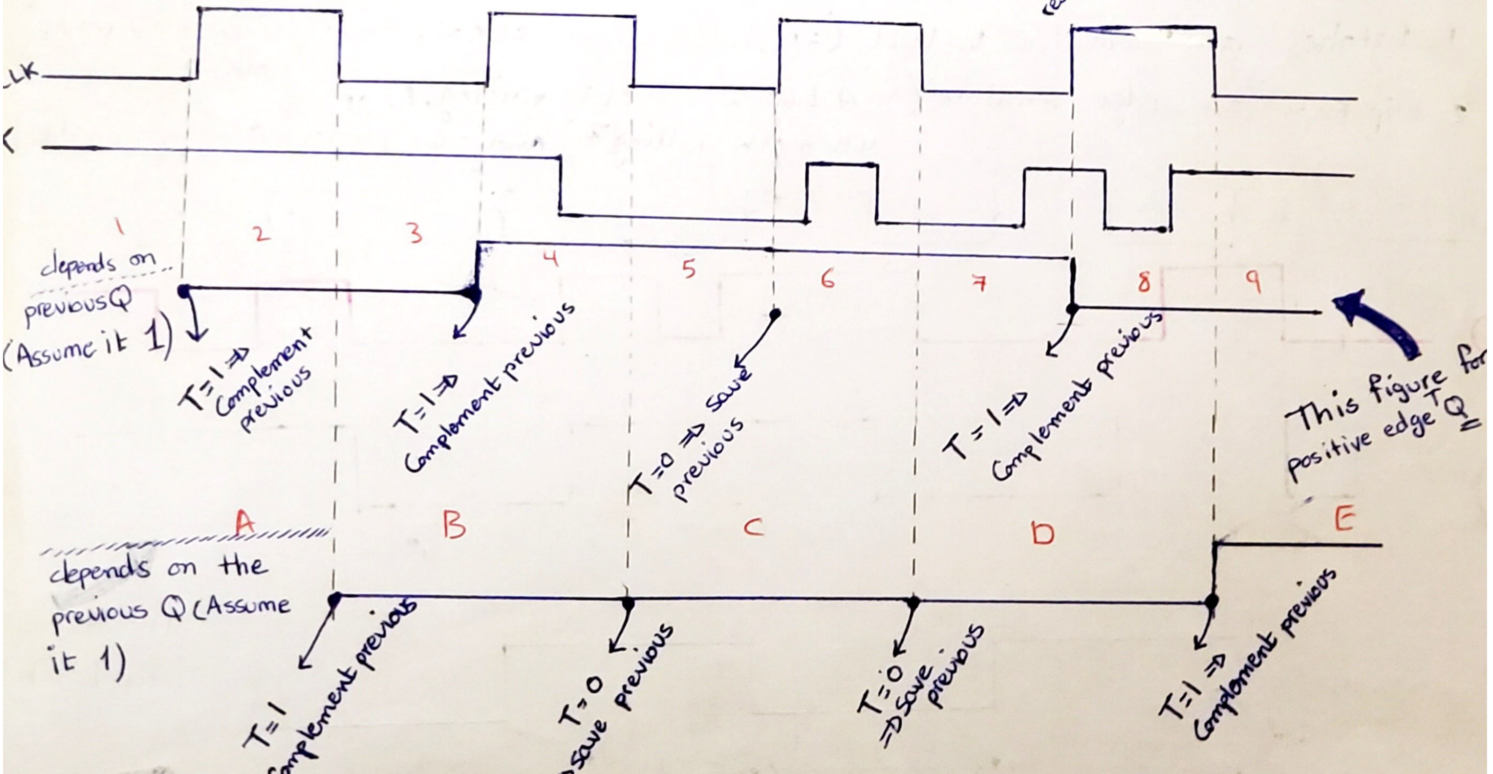
S	R	Q	Q̄
0	0	1	1 (invalid)
0	1	1	0
1	0	0	1
1	1	old Q	old Q̄



negative edge D F.F.

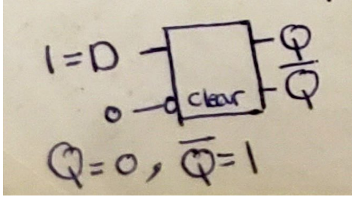
Scanned with CamScanner



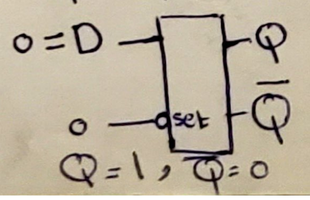

LK
K
1
2
3
4
5
6
7
8
9
depends on previousQ (Assume it 1)
T=1 ⇒ Complement previous
T=1 ⇒ Complement previous
T=0 ⇒ save previous
T=1 ⇒ Complement previous
This figure for positive edge T Q
A
B
C
D
E
depends on the previous Q (Assume it 1)
T=1 Complement previous
T=0 ⇒ save previous
T=0 ⇒ save previous
T=1 ⇒ Complement previous

* each Flip Flop or latch may have 2 extra inputs clear and set (preset)

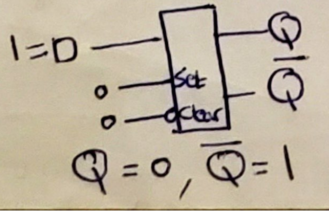
able clear input ⇒ Q for Flip Flop = 0, IF set is enabled ⇒ Q of Flip Flop = 1



$Q=0, \overline{Q}=1$



$Q=1, \overline{Q}=0$



$Q=0, \overline{Q}=1$

Cannot Activate both ⇒ unknown Q

Excitation Table :

Q(t)	Q(t+1)	T
0	0	0 → save
0	1	1 → complement
1	0	1 → complement
1	1	0 → save

$T = Q(t) \oplus Q(t+1)$

Q(t)	Q(t+1)	J	K	
0	0	0	X	0 0 → save 0 1 → reset
0	1	1	X	1 1 → complement 1 0 → set
1	0	X	1	1 1 → complement 0 1 → reset
1	1	X	0	0 0 → save 1 0 → set

Q(t)	Q(t+1)	D
0	0	0
0	1	1
1	0	0
1	1	1

Scanned with CamScanner

Excitation tables of flip flops

$\boxed{1}$ SR

Q_t	Q_{t+1}	S	R	
0	0	0	X	(Save or Reset)
0	1	1	0	(Set)
1	0	0	1	(Reset)
1	1	X	0	(Save or Reset)

$\boxed{2}$ D

Q_t	Q_{t+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

characteristic tables will be used in analysis, where excitation tables will be used in design problems.

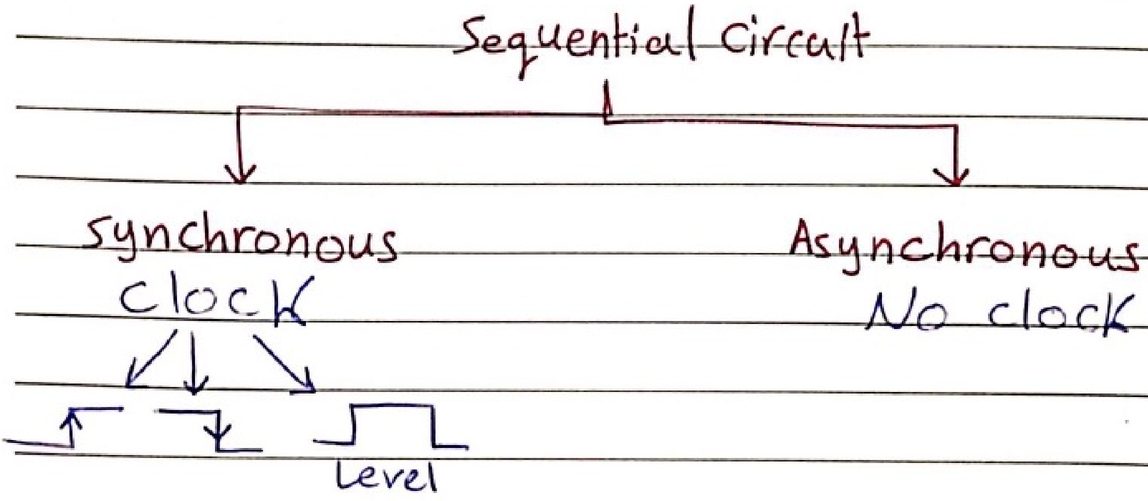
$\boxed{3}$ JK

Q_t	Q_{t+1}	J	K	
0	0	0	X	(Save, Reset)
0	1	1	X	(Complement, Set)
1	0	X	1	(Complement, Reset)
1	1	X	0	(Save, Set)

$\boxed{4}$ T

Q_t	Q_{t+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Scanned with CamScanner



✿ Flip Flop / Latch

① No clock latch

② clocked latch

③ Flip Flop

✿ Finite state Machine (FSM) Types:

1) Mealy FSM

$Q(t+1) \Leftarrow Q(t)$, input (x, y, z)

output $\Leftarrow Q(t)$, input

output depends on current states and external inputs

2) Moore FSM:

$Q(t+1) \Leftarrow Q(t)$, input (x, y, z)

output $\Leftarrow Q(t)$

output depends on current states only

Scanned with CamScanner

Analyze the Fallowing sequential circuit

$\boxed{Q_3}$

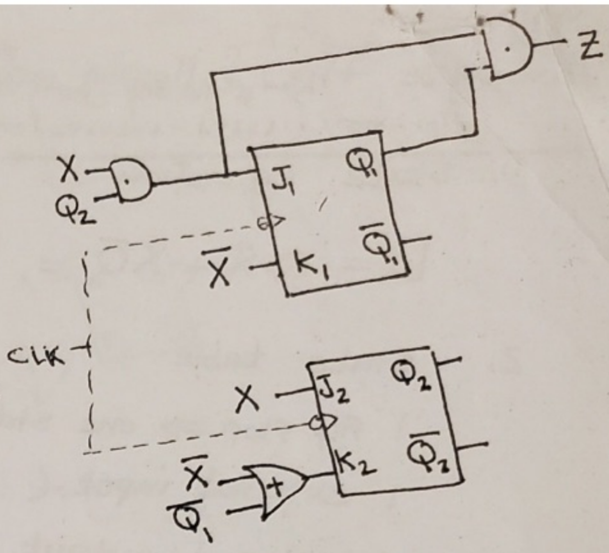
1. state equations

$J_1 = X \cdot Q_2 \quad , \quad K_1 = \overline{X}$

$J_2 = X \quad , \quad K_2 = \overline{X} + \overline{Q_1}$

$Z = Q_1 \cdot Q_2 \cdot X$

J	K	Q
0	0	save
0	1	reset
1	0	set
1	1	Comp



2. State table

	present state		external inputs	Next State		external outputs
	Q_2	Q_1	X	Q_2	Q_1	Z
State A	0	0	0	0 reset	0 reset	0
	0	0	1	1 comp	0 save	0
State B	0	1	0	0 reset	0 reset	0
	0	1	1	1 set	1 save	0
State C	1	0	0	0 reset	0 reset	0
	1	0	1	0 Comp	1 set	0
State D	1	1	0	0 reset	0 reset	0
	1	1	1	1 set	1 set	1

State table size = $2^{(\text{\# of Flip Flops} + \text{external inputs})} = 2^{(2+1)} = 8$ rows

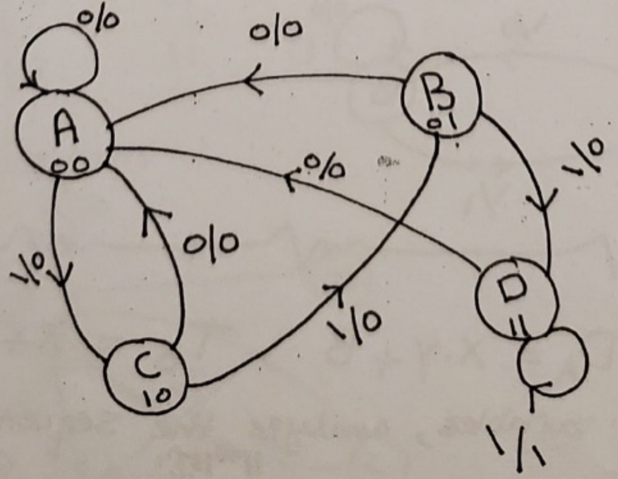
<u>Never</u> associate the external output with the Next state

3. State diagram

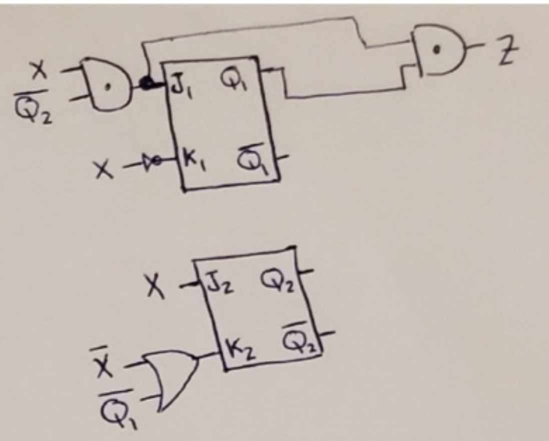
$\# \text{ of States} = 2^{(\text{\# of Flip Flops})} = 2^2 = 4$

$\# \text{ of outgoing arows} = 2^{(\text{\# of external inputs})} = 2^1 = 2$

Mealy FSM



Analyze the following Sequential circuit



*Note : the circuit is not disconnected.

For this circuit :

- The number of external inputs = 1 (X)
- The " " " outputs = 1 (Z)
- Number of states = 1 , For every F.F their is a state
- Number of circles = $2^{\#\text{OF F.F}} = 2^2 = 4$
- Number of outgoing arrows For each circles = $2^{\#\text{OF external inputs}} = 2^1 = 2$

Analysis steps

1- State equations (every input of every F.F has a state equation)

$\Rightarrow J_1 = X\overline{Q_2}$, $K_1 = \overline{X}$, $J_2 = X$, $K_2 = \overline{X} + \overline{Q_1}$, $Z = X \cdot \overline{Q_2} \cdot Q_1$

2- State table

present state		external inputs	Next State		external output
Q_2	Q_1	X	Q_2	Q_1	Z
0	0	0	0	0	0
0	0	1	1	1	0
0	1	0	0	0	0
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	1	1	0

Assume

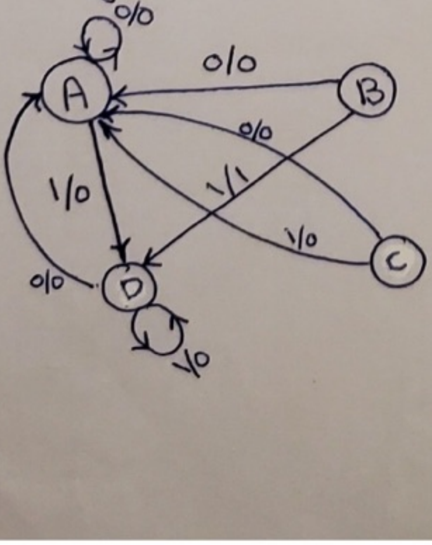
$Q_2Q_1 = 00 \Rightarrow$ State A

$Q_2Q_1 = 01 \Rightarrow$ State B

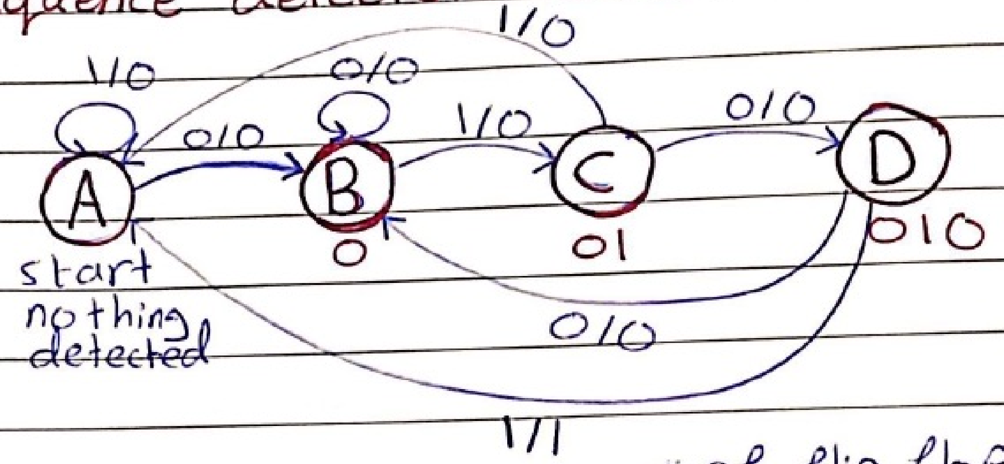
$Q_2Q_1 = 10 \Rightarrow$ State C

$Q_2Q_1 = 11 \Rightarrow$ State D

3- State diagram



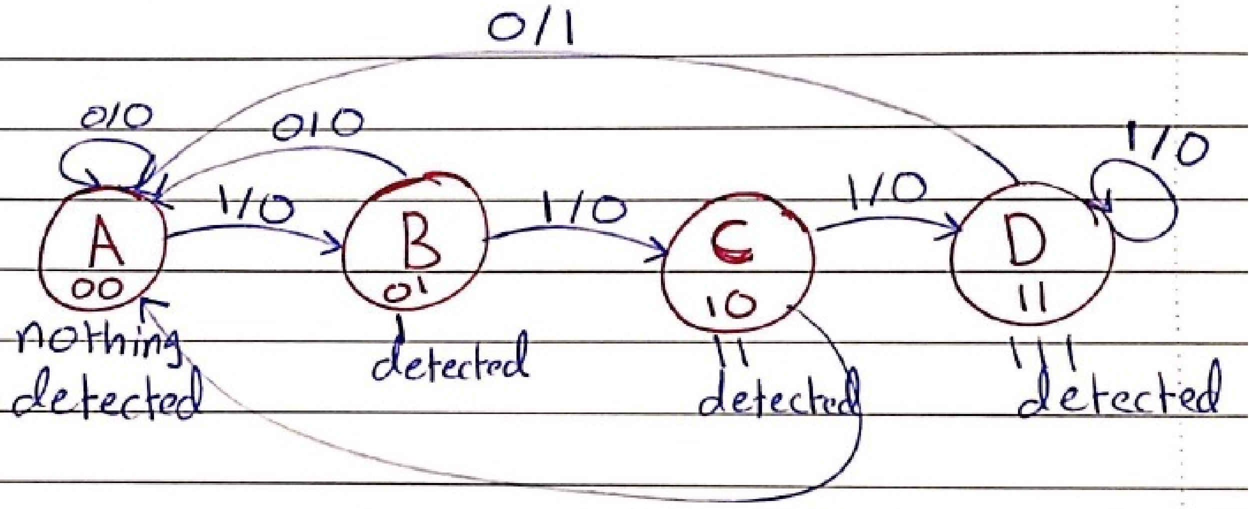
Sequence detector 0101



*Max # of circles = $2^{\#\text{ of flip flop}}$

*Max # of outgoing arrow of each circle = $2^{\#\text{ of external inputs}}$

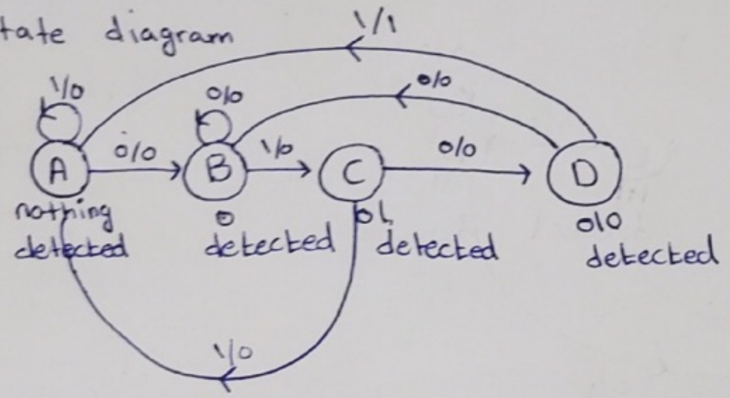
Sequence detector 1110



Scanned with CamScanner

design a sequence detector to detect 0101 using JK Flip_Flop

1- State diagram



2 - State table

	present states			Nextstate		
	Q_1	Q_0	external input X	Q_1	Q_0	external output Z
A	0	0	0	0	1 B	0
	0	0	1	0	0 A	0
B	0	1	0	0	1 B	0
	0	1	1	1	0 A	0
C	1	0	0	1	1 D	0
	1	0	1	0	0 A	0
D	1	1	0	0	1 B	0
	1	1	1	0	0 A	1

* since we have 4 states ⇒ we need 2 Flip- Flops ⇒ ~~there~~ are 2 states

* there exist 1 external input

* " " " " output

Assume

state A = 00

State B = 01

state C = 10

state D = 11

3 - state equations

J_1	K_1	J_0	K_0
0	X	1	X
0	X	0	X
0	X	X	0
1	X	X	1
X	0	1	X
X	1	0	X
X	1	X	0
X	1	X	1

$X \backslash Q_1Q_0$	00	01	11	10
0	0	0	X	X
1	0	1	X	X

$J_1 = Q_0 X$

$X \backslash Q_1Q_0$	00	01	11	10
0	X	X	1	0
1	X	X	1	1

$K_1 = X + Q_0$

$X \backslash Q_1Q_0$	00	01	11	10
0	1	X	X	1
1	0	X	X	0

$J_0 = \overline{X}$

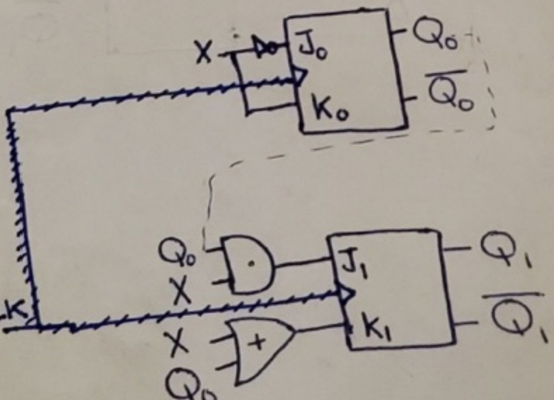
$X \backslash Q_1Q_0$	00	01	11	10
0	X	0	0	X
1	X	1	1	X

$K_0 = X$

excitation table

Q_t	Q_{t+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

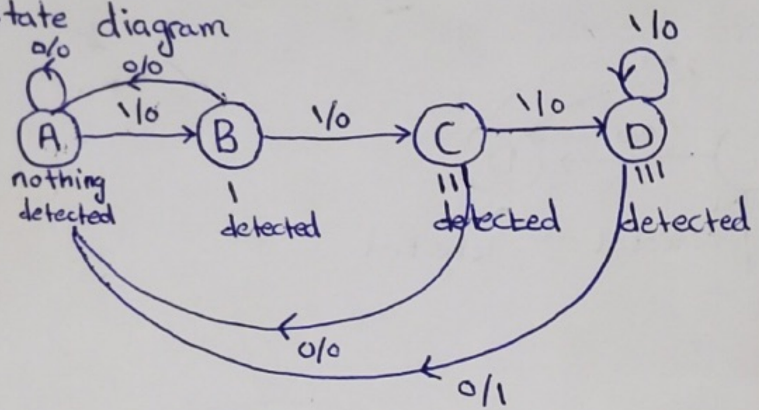
3- draw



Scanned with CamScanner

design a sequence detector to detect 1110 using D Flip Flop.

1. state diagram



2- State table

p.state Q_1	Q_0	external input X	N. state Q_1	Q_0	external output Z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	0

D_1	D_0
0	0
0	1
0	0
1	0
0	0
1	1
0	0
1	1

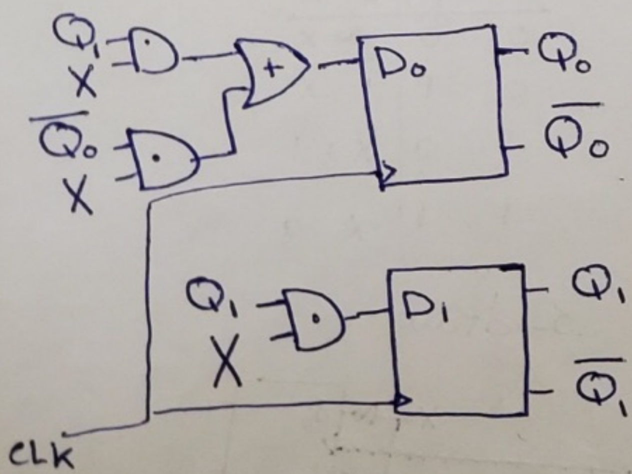
$X \backslash Q_1Q_0$	00	01	11	10
0	0	0	0	0
1	0	1	1	1

$D_1 = Q_1X$ + XQ_0

$X \backslash Q_1Q_0$	00	01	11	10
0	0	0	0	0
1	1	0	1	1

$D_0 = Q_1X + \overline{Q_0}$

3- state diagram



Scanned with CamScanner

Counters

* Counters are classified into types

1- based on Counting sequence

- Up counters (0,1,2,3,0,1,2,3 --) (2,4,6,2,4,6 --)
- down Counters (3,2,1,0, 3,2,1,0 --) (8,6,4,2,0,8,6,4,2,0 --)
- Up/down Counters

↳ these counters may Count either up or down based on selection Line (c) ⇒ C=0 Count up, C=1 ⇒ Count dow

2- base on counting regularities

- Regular:

$$\underbrace{\text{start by } (0)\text{, end on } (2^{\# \text{OF F.F}}-1)\text{, increment by } 1}_{\text{Regular, up}} =$$

$$\underbrace{\text{start by } (2^{\# \text{OF F.F}}-1)\text{, end on } (0)\text{, decrement by } 1}_{\text{Regular, down}}$$

- Irregular: start any where, Jump to any where

Ex 0,1,2,3, 0,1,2,3 ⇒ start @ zero, end @ $2^{\# \text{OF FF}}-1 = 2^2-1=$
alway incremented by 1
⇒ Regular up.

0,1,2,----,7,0,1,2, ---- 7,0 ⇒ also Regular up.

3,2,1,0,3,2,1,0 ⇒ Regular down
→ start @ $2^{\# \text{FF}}-1 = 2^3-1$
→ stop @ Zero
→ decrement by 1

7,5,3,1,7,5,3,1 ⇒ Irregular, why?

15,10,5,0 ⇒ also Irregular, why?

These are Irregular down Counters.

0,1,2,6,10,15 ⇒ Irregular up, why?

Note * Irregular Counters are harder to design
* There are ways to design Regular up, or Regular down using simplified model.

Scanned with CamScanner

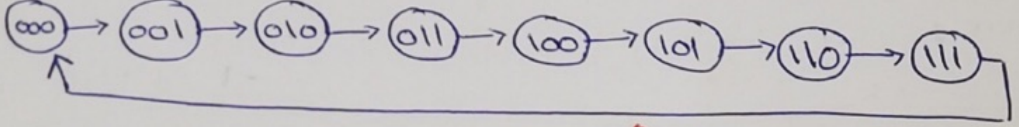
Q. design a regular up Counter that Counts from 0 to 7 (the question maybe, use 3 F.F to design a regular up Counter). Use JK Flip Flops.

S. Important step you must find the number of required F.F first

How? Convert the max count into binary ⇒ Find the number of bits ⇒ # of bits = # of F.F's, why??!

For the above Counter, max Count = 7, $7_d = (111)_b = 3$. F.F's.

1- State diagram



* No external input if it's up, down.
IF it's up/down external input is needed.

* No external outp always.

Q_t	Q_{t+1}	J	K
0	0		
0	1		
1	0		
1	1		

2- State table

present state			N. State								
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	[illegible]	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	0	0	X	1	X	1	X	1

3- State equations

$J_0 = K_0 = 1$, why it should be one always?

$Q_0 \backslash Q_2Q_1$	00	01	11	10
0	0	X	X	0
1	1	X	X	1

$J_1 = Q_0$

$Q_0 \backslash Q_2Q_1$	00	01	11	10
0	X	0	0	X
1	X	1	1	X

$K_1 = Q_0$

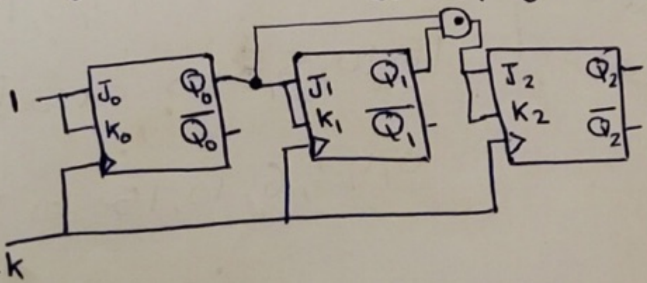
$Q_0 \backslash Q_2Q_1$	00	01	11	10
0	0	0	X	X
1	0	1	X	X

$J_2 = Q_1Q_0$

$Q_0 \backslash Q_2Q_1$	00	01	11
0	X	X	0
1	X	X	[illegible]

$K_2 = Q_1Q$

no external input/output Columns ⇒ Nothing on arrows



Notes

* This is synchronous Counter since all F.F's operate on th same clock signal

* The counting sequence (numbers) is taken on $Q_2Q_1Q_0$

Scanned with CamScanner

design the same previous Counter using T. Flip Flop

* The same steps

1- State diagram (the same)

2- State table (the same)

T_2	T_1	T_0
0	0	1
0	1	1
0	0	1
1	1	1
0	0	1
0	1	1
0	0	1
1	1	1

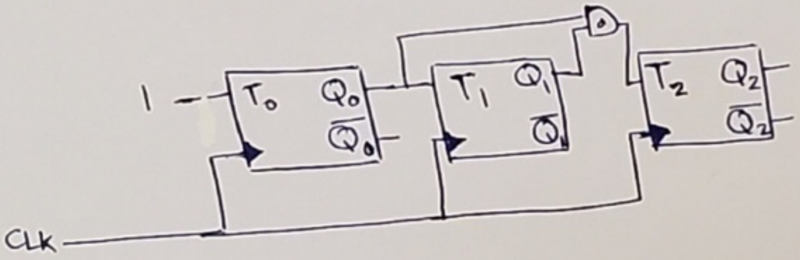
* $T_0 = 1$, as mentioned befor, the Least significant F.F must have one as input

* $T_1 = Q_0$

* $T_2 =$

$Q_1 \cdot Q_0$

Q_0 \ Q_2Q_1	00	01	11	10
0	0	0	0	0
1	0	1	1	0

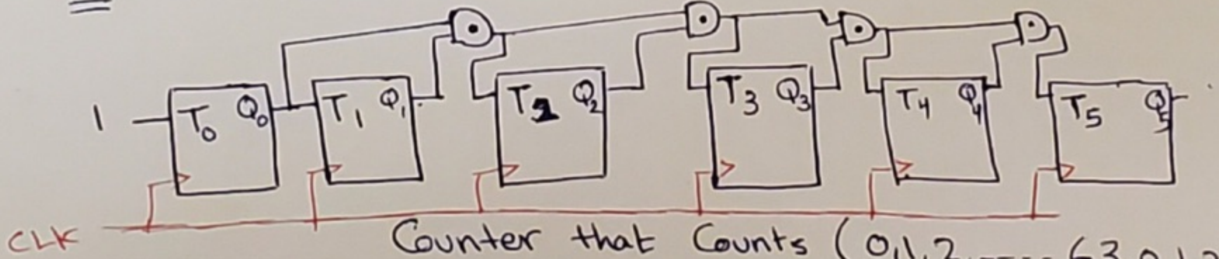


* Simplified design for Regular up counters

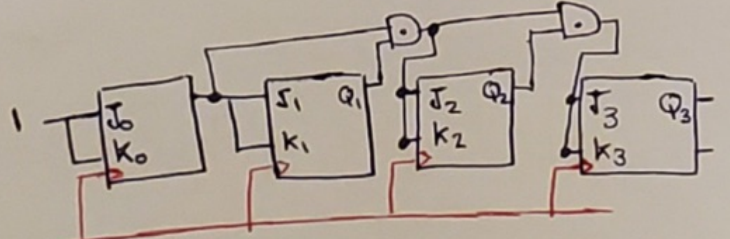
1- Connect LS (Least significant) F.F inputs to Zero.

2- Connect the input of the remaining F.F to the ANDed previous output.

Ex



Counter that Counts (0,1,2,---- 63,0,1,2,---)



Counter that Counts (0,1,2,---,15,0,1,2,---15).

Design a Counter to count according to the following sequence

7, 6, 5, 4, 3, 2, 1, 0, 7, 6 - - - -

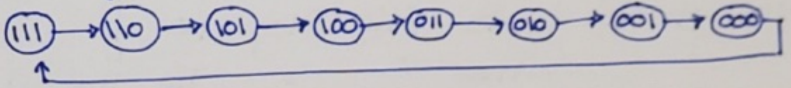
1- State diagram

Number of states = 8 ⇒ # OF Flip Flops = 3

From the counting sequence ⇒ this is Regular, down counter.

step width = 1 — all range values are covered

State diagram



No external output / input

2- State table

present state			Next state		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	0	1	1	1
0	0	1	0	0	0
0	1	0	0	0	1
0	1	1	0	1	0
1	0	0	0	1	1
1	0	1	1	0	0
1	1	0	1	0	1
1	1	1	1	1	0

every Flip Flop has a State

Be careful to MSB States

3- state equations

- Every input for every flip-flop has an equation
- the state equations depends on the Flip Flop types
- Lets design using a- JK b- T

✓ using JK

present			Next								
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	1	1	1	1	X	1	X	1	X
0	0	1	0	0	0	0	X	0	X	X	1
0	1	0	0	0	1	0	X	X	1	1	X
0	1	1	0	1	0	0	X	X	0	X	1
1	0	0	0	1	1	X	1	1	X	1	X
1	0	1	1	0	0	X	0	0	X	X	1
1	1	0	1	0	1	X	0	X	1	1	X
1	1	1	1	1	0	X	0	X	0	X	1

Q_2Q_1 / Q_0

1	0	X	X
0	0	X	X

$J_2 = \overline{Q_1} \cdot \overline{Q_0}$

Q_2Q_1 / Q_0

X	X	0	1
X	X	0	0

$K_2 = \overline{Q_1} \cdot \overline{Q_0}$

Q_2Q_1 / Q_0

1	X	X	1
0	X	X	0

$J_1 = \overline{Q_0}$

X	1	1	X
X	0	0	X

$K_1 = \overline{Q_0}$

$J_0 = 1$

$K_0 = 1$

Q_2Q_1 / Q_0

1	0	0	[illegible]
0	0	0	0

$T_2 = \overline{Q_1} \cdot \overline{Q_0}$

Q_2Q_1 / Q_0

1	1	1	1
0	0	0	0

$T_1 = \overline{Q_0}$

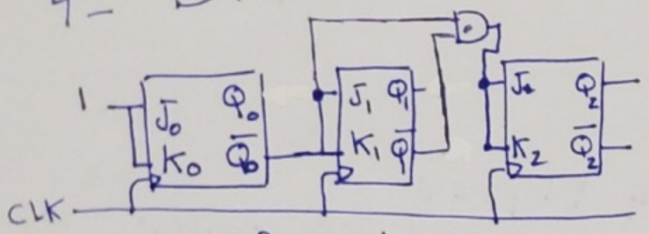
$T_0 = 1$

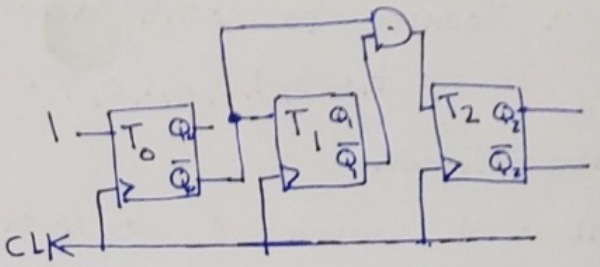
using T

present			Next					
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	T_2	T_1	T_0
0	0	0	1	1	1	1	1	1
0	0	1	0	0	0	0	0	1
0	1	0	0	0	1	0	1	1
0	1	1	0	1	0	0	0	1
1	0	0	0	1	1	1	1	1
1	0	1	1	0	0	0	0	1
1	1	0	1	0	1	0	1	1
1	1	1	1	1	0	0	0	1

Scanned with CamScanner

4_ Draw





simplified design

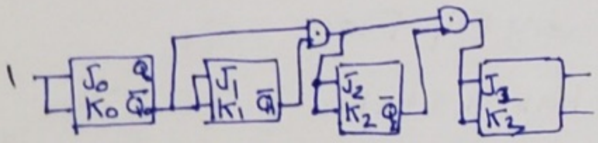
you Can use the following technique if the counter is <u>Regular down</u> & build from either <u>JK</u> or <u>T</u> Flip Flops

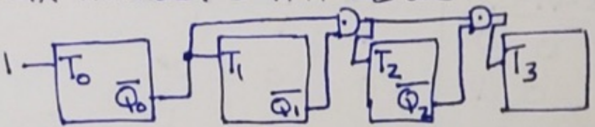
1- Connect the inputs of the LSB Flip Flop to <u>1</u>

2- " " " " other " " to the <u>ANDing</u> of previous Flip Flops ($\overline{Q}$) signal

<u>Ex</u> design a Counter to Count <u>15</u>, 14, 13, ---, 0, 15, 14 ---, 0

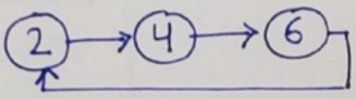
max number = 1111 ⇒ we need 4 Flip Flops





use T Flip Flops to design a counter to count 2, 4, 6, 2, 4, 6 ----

1- State diagram



⇒ number of Flip Flops = 3 since the max count = 6 = 110

2- State table

Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	0	X	X	X
0	0	1	X	X	X
0	[illegible]	0	1	0	0
0	1	1	X	X	X
1	0	0	1	1	0
1	0	1	X	X	X
1	1	0	0	1	0
1	1	1	X	X	X

3- State equations

T_2	T_1	T_0
X	X	X
X	X	X
1	1	0
X	X	X
0	1	0
X	X	X
1	0	0
X	X	X

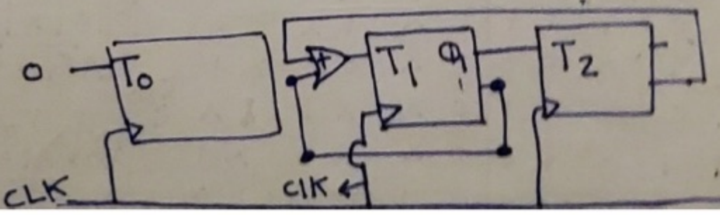
Q_0 \ Q_2Q_1				
	X	1	1	0
	X	X	X	X

$T_2 = Q_1$

Q_0 \ Q_2Q_1	00	01	11	10
	X	1	0	1
	X	X	X	X

$T_1 = \overline{Q_2} + \overline{Q_1}$

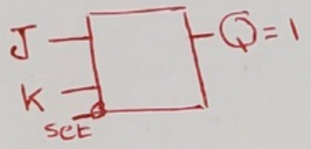
$T_0 = 0$



Remind

Any Flip-Flop has clear and set inputs
clear will make Q=0, if Clear is Active
set will make Q=1, if set is Active

Ex



if J=K=Q=1, set=0
⇒ Next Q=1 (set is Active Low)

* we will use clear and set to design an irregular Counter

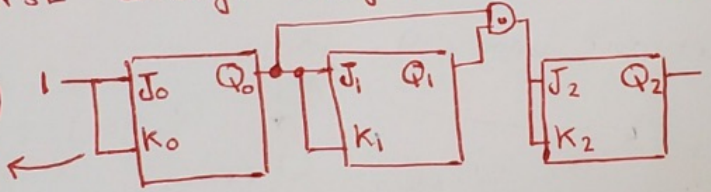
Q1 design a Counter that counts 2,3,4,5, 2,3,4,5---

Solution such counter is irregular because it start at 2 and ends at 5.

2,3,4,5,[]

First design aregular counter (regular/up)

This is the simplified model



why 3 Flip Flops?

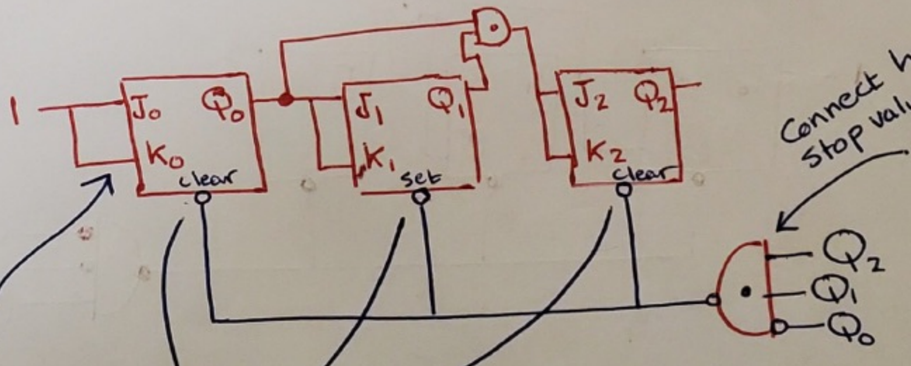
.010

modify it to limit the counts, How?

Replace 6 by 2 (after 5 we don't wish to see 6 we wish to see 2)

Important

This way works Correctly when the step width = 1
when Counter Counts 0,2,4,6 this way doesn't work



Connect here the stop value (6)

Counter clock is missing

Connect here the start over value (2)

6 = 110
$Q_2 Q_1 Q_0$
⇒ Q_2 not Complement
⇒ Q_1 " "
⇒ Q_0 Complement

which means
clear Q_0
clear Q_2
set Q_1

$2_d = (010)_{binary}$
MSB (Q_2)
LSB (Q_0)

Scanned with CamScanner

design a counter to count 9, 10, 11, 12, 9, 10, 11, 12 ---

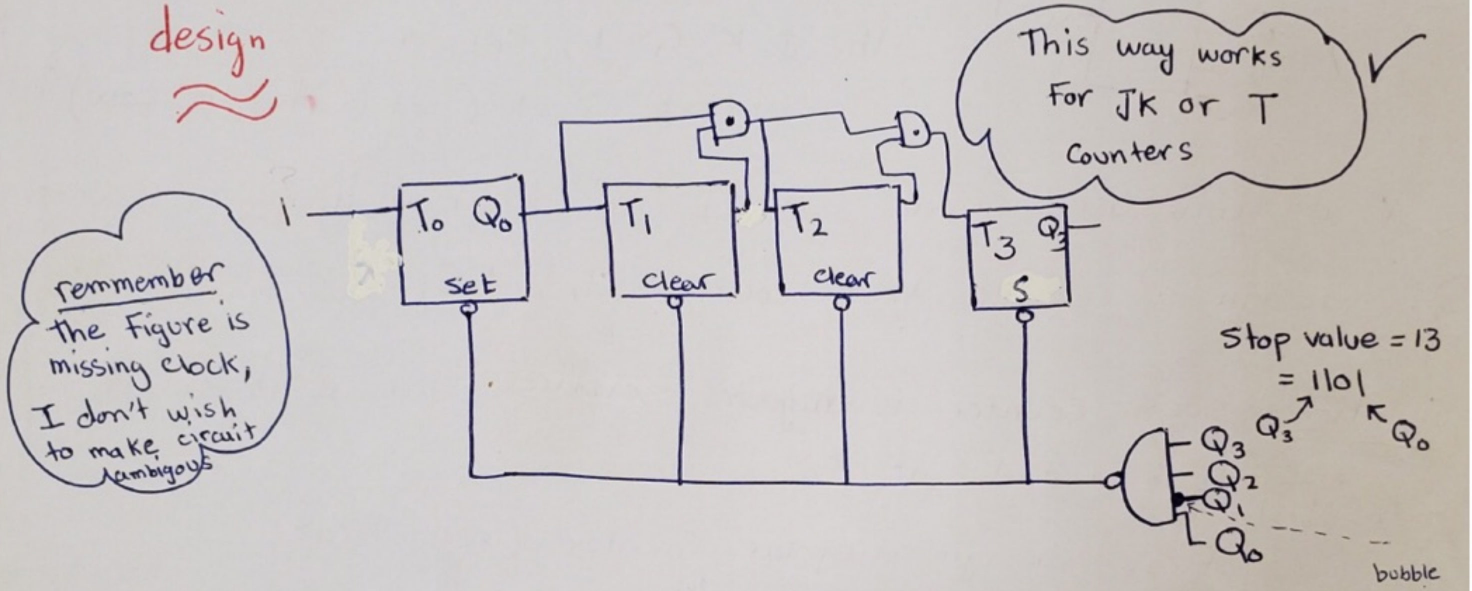
* our previous way works because step width = 1 ✓

* Stop value = 13 (the value we don't wish to see)

* Start over value = 9

* # of required Flip - Flops = 4 , why?

design



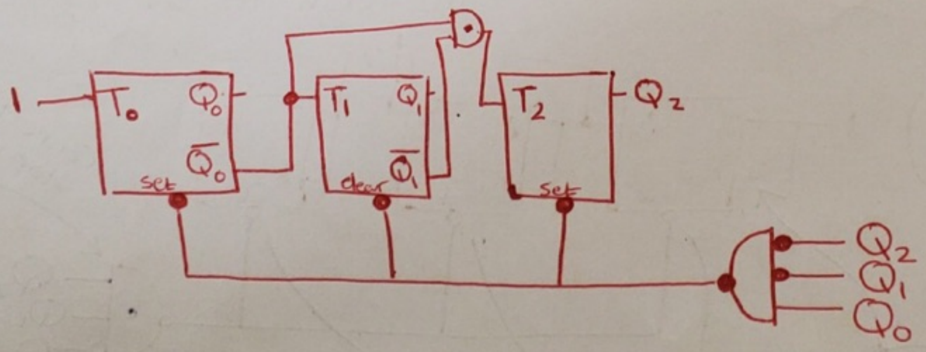
This way works for JK or T counters ✓

remmember the figure is missing clock, I don't wish to make circuit ambigous

Start over value = 9 = 1001

Q_3 (set). Q_2 (clear) Q_1 (reset, clear) Q_0 (set)

Ex what is the counting sequence for the counter below?



* For above circuit if current state is 4, what is the state after 3 clock pulses?

5)

Design a counter to count up and down based on external signal (X). IF X=0 ⇒ Count up, IF x=1 ⇒ Count down. The min Count is 0 and the max count is 3. Count step = 1

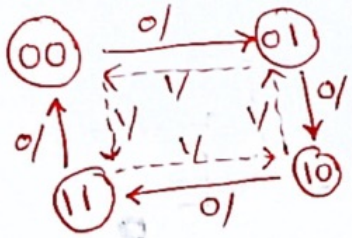
From the Statement

→ # of States = 4 (0, 1, 2, 3)

→ X is an external input to count either up or down

→ number of Flip Flops = 2

1 State diagram



2 State table

present State Q_1	Q_0	external input X		Next State Q_1	Q_0	external output
0	0	0	Up	0	1	NO external output exist
0	0	1		1	1	
0	1	0	Up	1	0	
0	1	1		0	0	
1	0	0	Up	1	1	
1	0	1		0	1	
1	1	0	Up	0	0	
1	1	1		1	0	

3 - State equations

J_1	K_1	J_0	K_0
0	X	1	X
1	X	1	X
1	X	X	1
0	X	X	1
X	0	1	X
X	1	1	X
X	1	X	1
X	0	X	1

$X \backslash Q_1Q_0$	00	01	11	10
0	0	1	X	X
1	1	0	X	X

$J_1 = Q_0\bar{X} + \bar{Q}_0 X = Q_0 \oplus X$

$X \backslash Q_1Q_0$	00	01	11	10
0	X	X	1	0
1	X	X	0	1

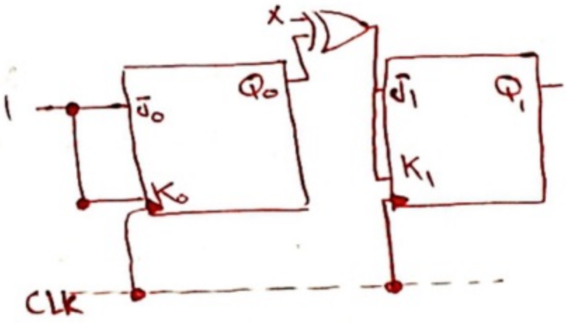
$K_1 = Q_0\bar{X} + \bar{Q}_0 X = Q_0 \oplus X$

$X \backslash Q_1Q_0$				
	1	X	X	1
	1	X	X	1

$J_0 = 1.$

$X \backslash Q_1Q_0$				
	X	1	1	X
	X	1	1	X

$K_0 = 1$



Scanned with CamScanner

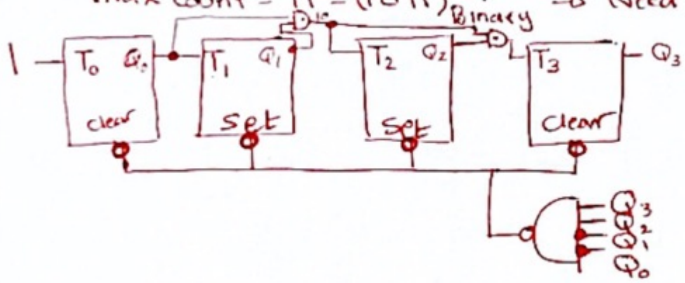
design a Counter to Count $\overset{MSB}{}$ 6, 7, 8, 9, 10, 11, 6, 7, 8, 9, 10, 11, 6, ---

Stop Condition = 12 = $(\overset{MSB}{1}100)_B$

Start over value = 6 = $(\overset{MSB}{0}110)_B$

max Count = 11 = (1011) Binary ⇒ Need 4 Flip Flops

0110



0110

1100

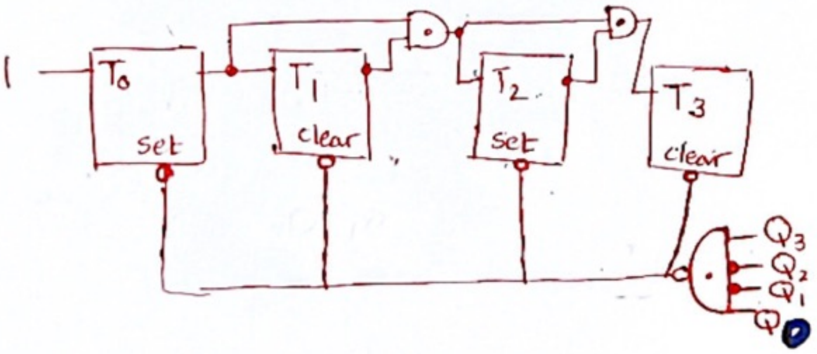
design a Counter to count 5, 6, 7, 8, 5, 6, 7, 8, ---

stop Coundition = 9 = $(1001)_B$

Start over value = 5 = $(0101)_B$

max Count = 8 ⇒ (1000) Binary ⇒ 4 Flip Flops.

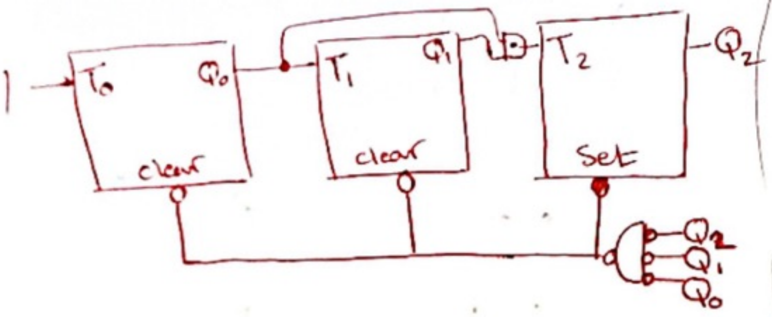
0110



010

1001
1001

design a Counter to Count 4, 5, 6, 7, 4, 5, 6, 7



Stop Condition = 0 = $(000)_B$

Start over value = 4 = $(100)_B$

max Count = 7 = $(111)_B$ ⇒ 3 FlipFlops

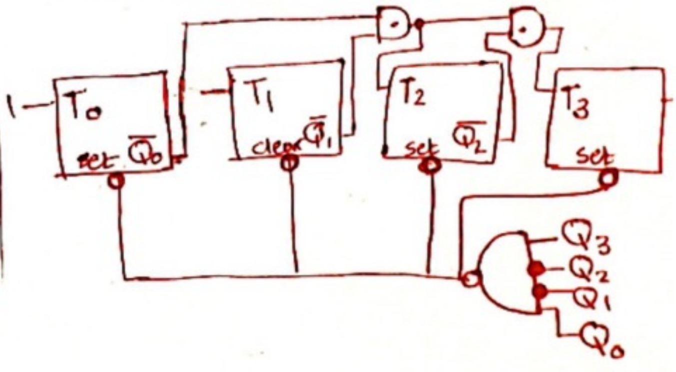
Design a Counter to Count

13, 12, 11, 10, 13, 12, 11, 10 ----

Stop Condition = 9 = $(1001)_B$

Start over value = 13 = $(1101)_B$

max Count = 13 = $(1101)_B$ ⇒ 4 Flip Flops



Registers

1- Sequential circuits consist of one or more flip flops that can 1)Load new 2)store 3)shift 4)output binary values.

2- Number of FF determines the register size.

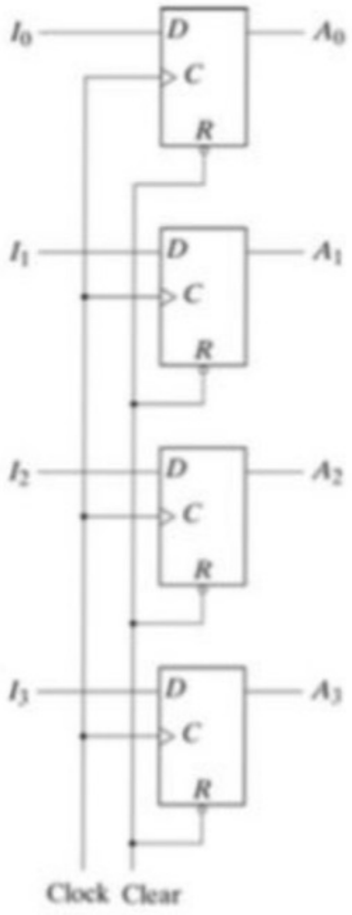
3- **Four** types of registers:

   a. Parallel input/parallel output

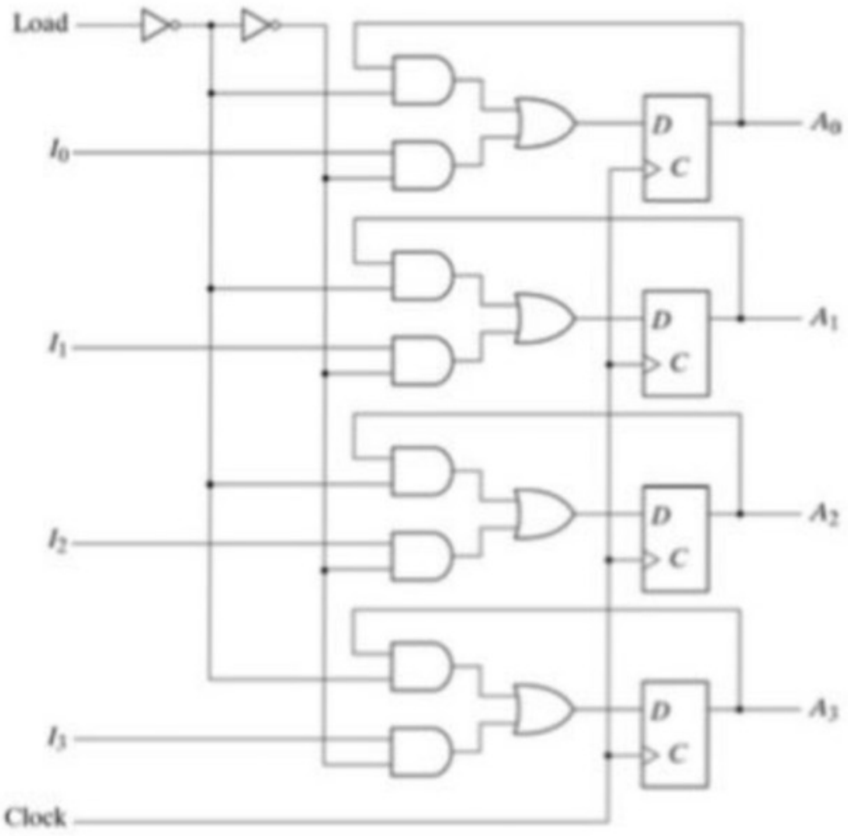
   b. Serial input/serial output

   c. Serial input/parallel output

   d. Parallel input/serial output.



1



2

Scanned with CamScanner

Registers Types II

- Different combination of registers inputs and outputs leads to 4 types of registers:
  - Parallel input/parallel output
  - Serial input/serial output
  - Serial input/parallel output
  - Parallel input/serial output.
- The main operations performed by registers are:
  - shifting left and right.
    - applied with either serial output or with serial input.
  - No operation (just storage of data for later use).
    - Applied with parallel input/parallel output registers only.

Parallel in/Parallel out Registers I

- The next figure shows a 4-bit parallel in/parallel out register.
- 4-bit: since it contains 4 D-flip flops.
- All the 4 bits are applied at the same time to all flip flops input.
- After 1 clock pulse (exactly after the +ve edge) all bits are seen at the output of the flip flops (ready to read).
- So, 1 clock pulse is needed for both the input and output.
- The values of all inputs I0 – I3 remains unchanged as long as you want keep the stored data.
- Also, you can reset all flip flops to 0 by activating the clear direct input.

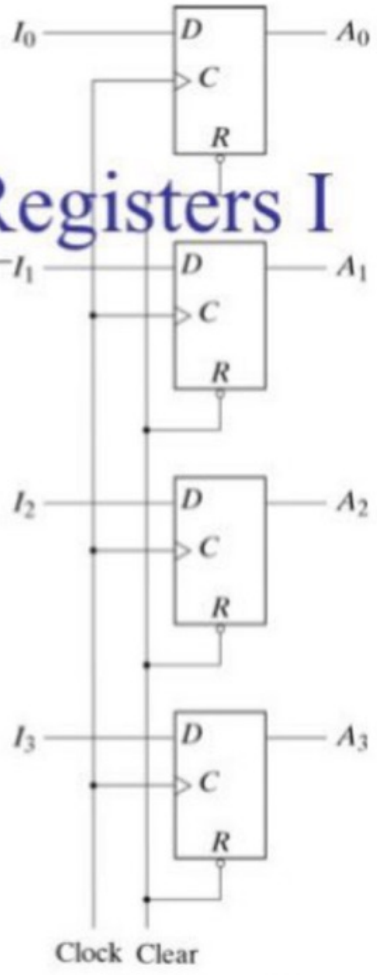


Fig. 6-1 4-Bit Register

Scanned with CamScanner

Parallel in/Parallel out Registers II

- Registers with Paralle Load
  - Has an additional gate and a control line calle "load".
  - These gates provide additional functions: either to load a new value to be stored in th register, or to keep the same value stored in th register.

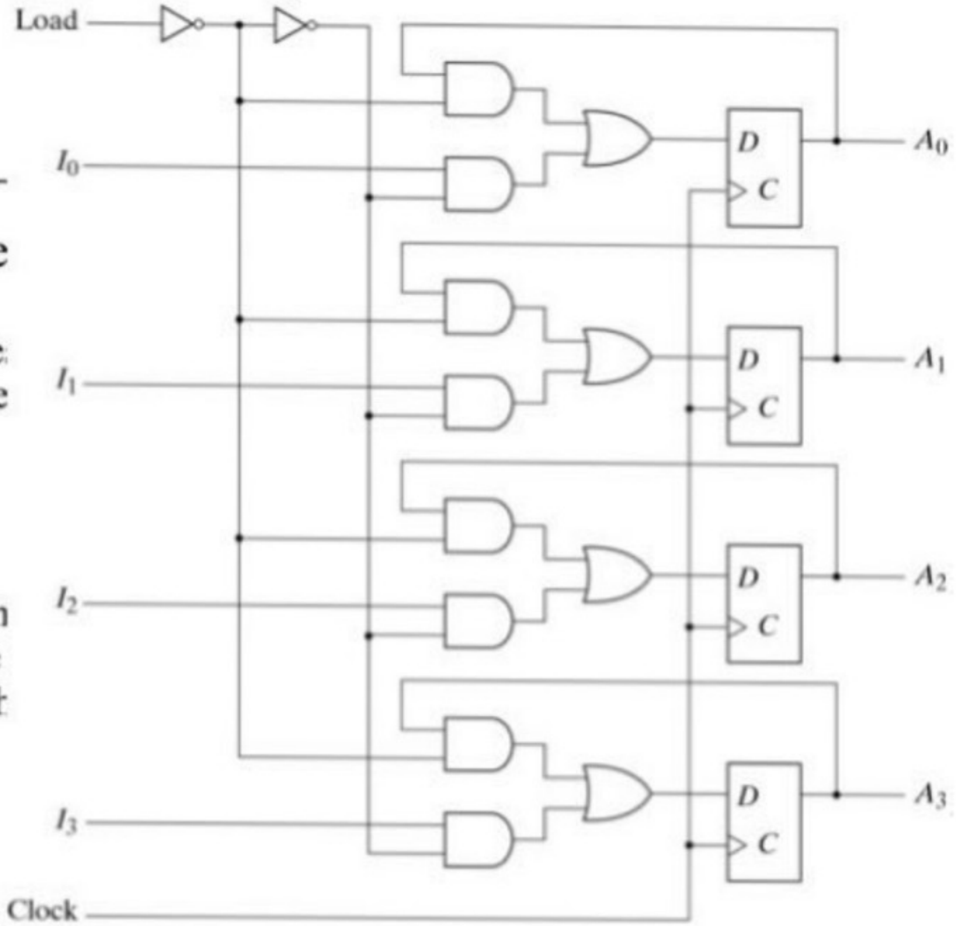


Fig. 6-2 4-Bit Register with Parallel Load

Parallel in/Parallel out Registers II

- Remember that to avoid changing the output value of a D FF you must do the following:
  - Either do not change D.
  - Or avoid seeing the clock at the FF.
- The register with parallel load avoid the need for the above two options using the gates and the "load" control line.
- When load = 1 □ you load a new binary value to the register.
- When load = 0 □ it means that you do not want to change the stored value in the register. This done by making the FF input (i.e. D) equals the current flip flop output Q.
- Have a look at the circuit to trace the above operation.

Serial in/Serial out Registers I

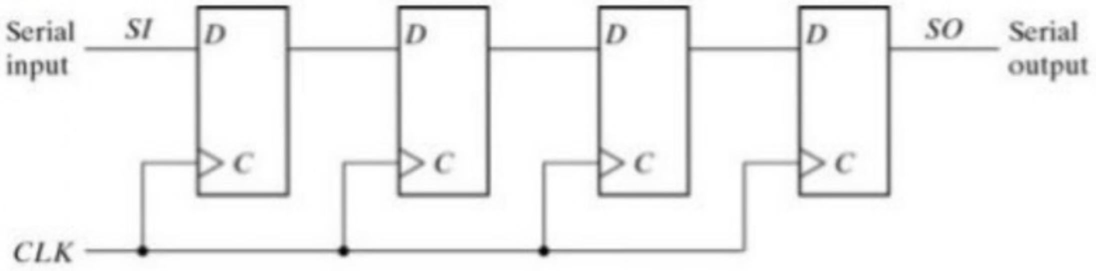


Fig. 6-3 4-Bit Shift Register

The above figure shows a 4-bit serial in/serial out register.

4-bit: since it contains 4 D-flip flops.

At the input you can apply only 1 bit at a time. So, you need 4 clock pulses to transfer all the input bits.

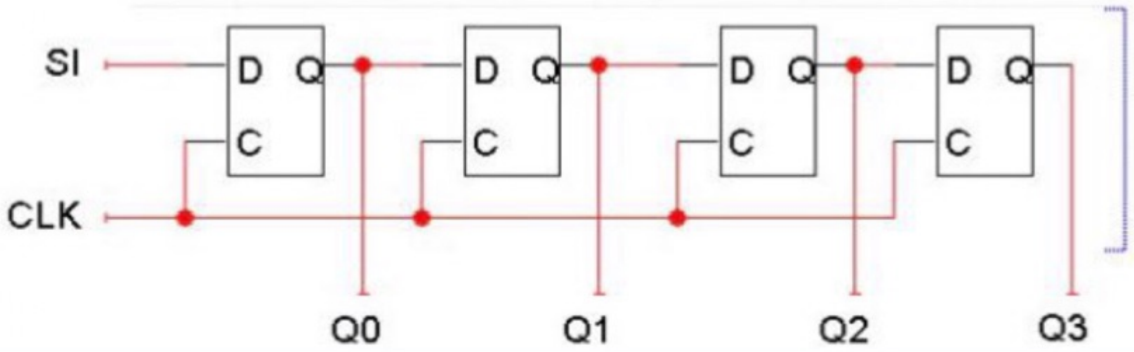
Serial in/Serial out Registers II

- Also, at the output you can read only 1-bit at a time. The first bit of the stored data can be seen at the output at the last clock cycle needed to store the data at the input (at clock cycle 4).
- Additional 3 clock cycles are needed to read the remaining 3 bits of the stored data.
- So, at total we need 7 clock pulses to store and read the data in the above register.
- The above register performs shift right operation on the input data.

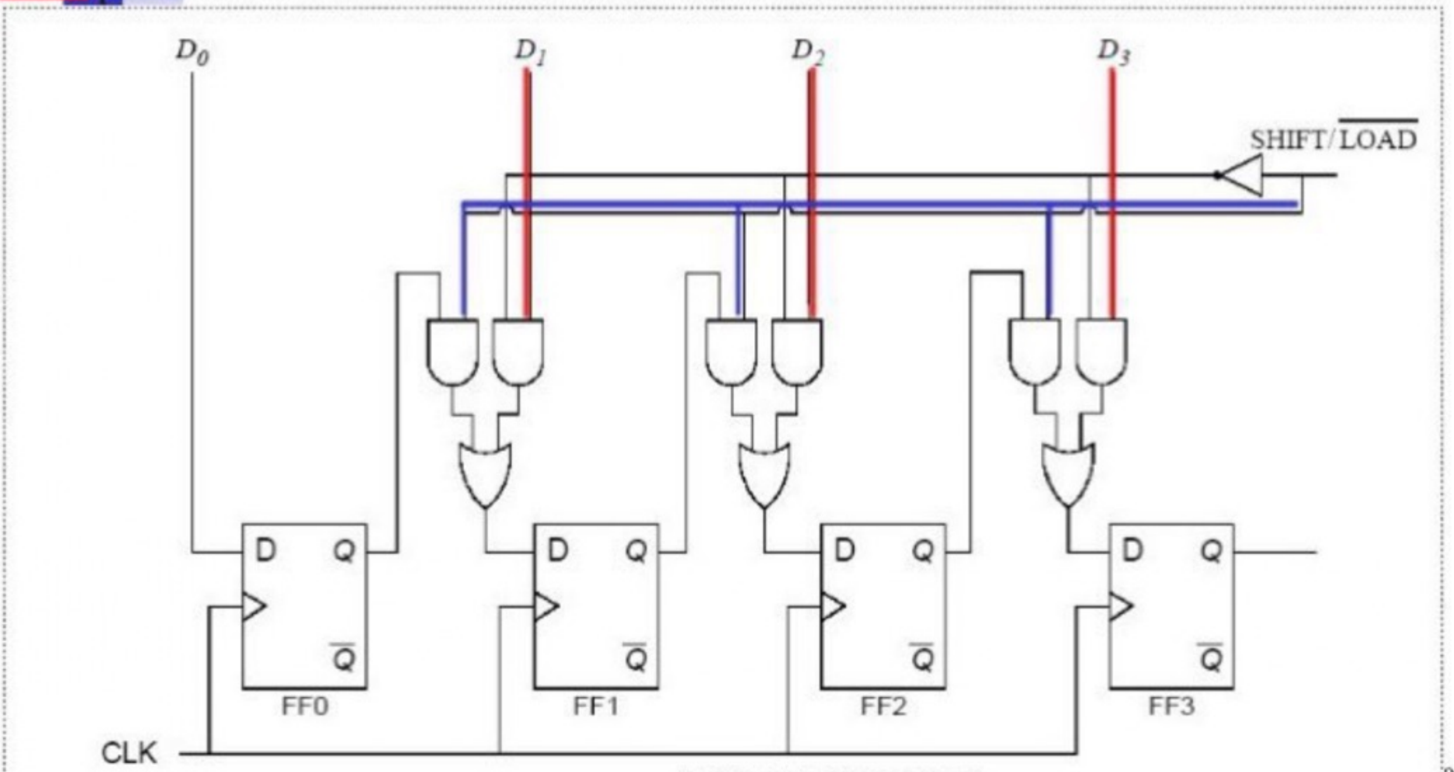
Scanned with CamScanner

Serial in/Parallel out Registers

- The same as in the figure in the previous slide with the exception that you are allowed to read all the outputs of the flip flops at the same time.
- Here you need n clock pulses to have an n-bit data stored in the register (and also ready for read).



Parallel in/Serial out Registers I



Parallel in/Serial out Registers II

- This register performs shift right operation.
- As you see in the previous slide there are external gates with a control line (similar to registers with parallel load).
- When this control line is 0 □ the Load operation is performed since it is active low.
- When the control line is 1 □ the shift operation is performed since it is active high.
- For this register you enter all inputs in parallel. However, you only see one bit at a time at its serial output.

Parallel in/Serial out Registers III

- When the Load is active (i.e. value on the control line = 0) all parallel inputs are loaded to the registers (i.e. applied to D inputs and seen on the FFs outputs).
- At this instance you only see D3 value on the output.
- Then you want to see all inputs values on the output, this is accomplished by shifting these values toward the output.
- To start shifting you must put 1 on the control line and you need an additional 3 clock pulses to see the remaining three bits on the output (one bit at each clock pulse).

Scanned with CamScanner

Universal Shift Register

- Is a bidirectional shift register (capable of shifting right and left) in addition to parallel load capabilities.
  - So, it contains all the registers types on the same circuit and it can be configured using control lines to select one of these types.
  - And it perfroms the two operations supplied by registers
    - Shifting left and right.
    - And no operation (storage of data).

Universal Shift Register

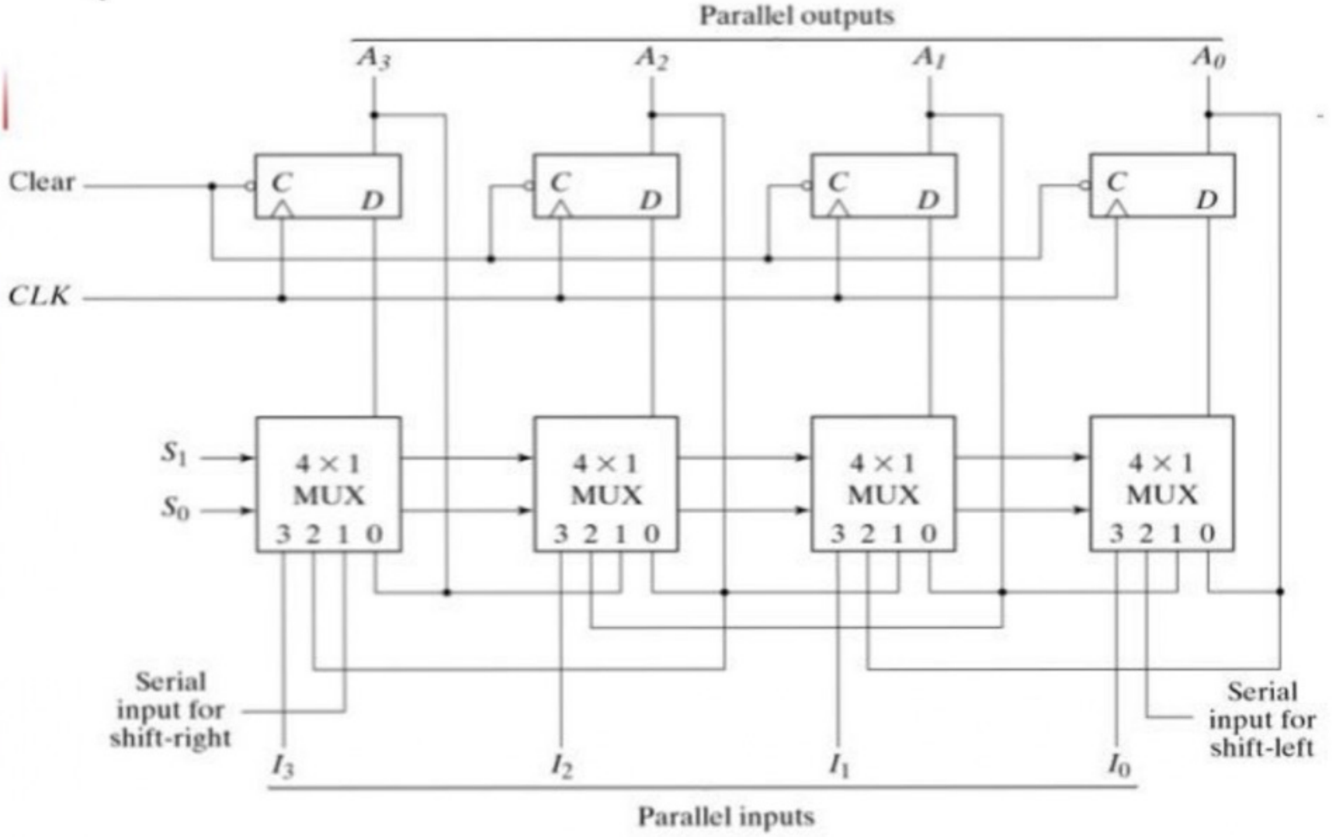


Fig. 6-7 4-Bit Universal Shift Register

Scanned with CamScanner

Universal Shift Register Operation I

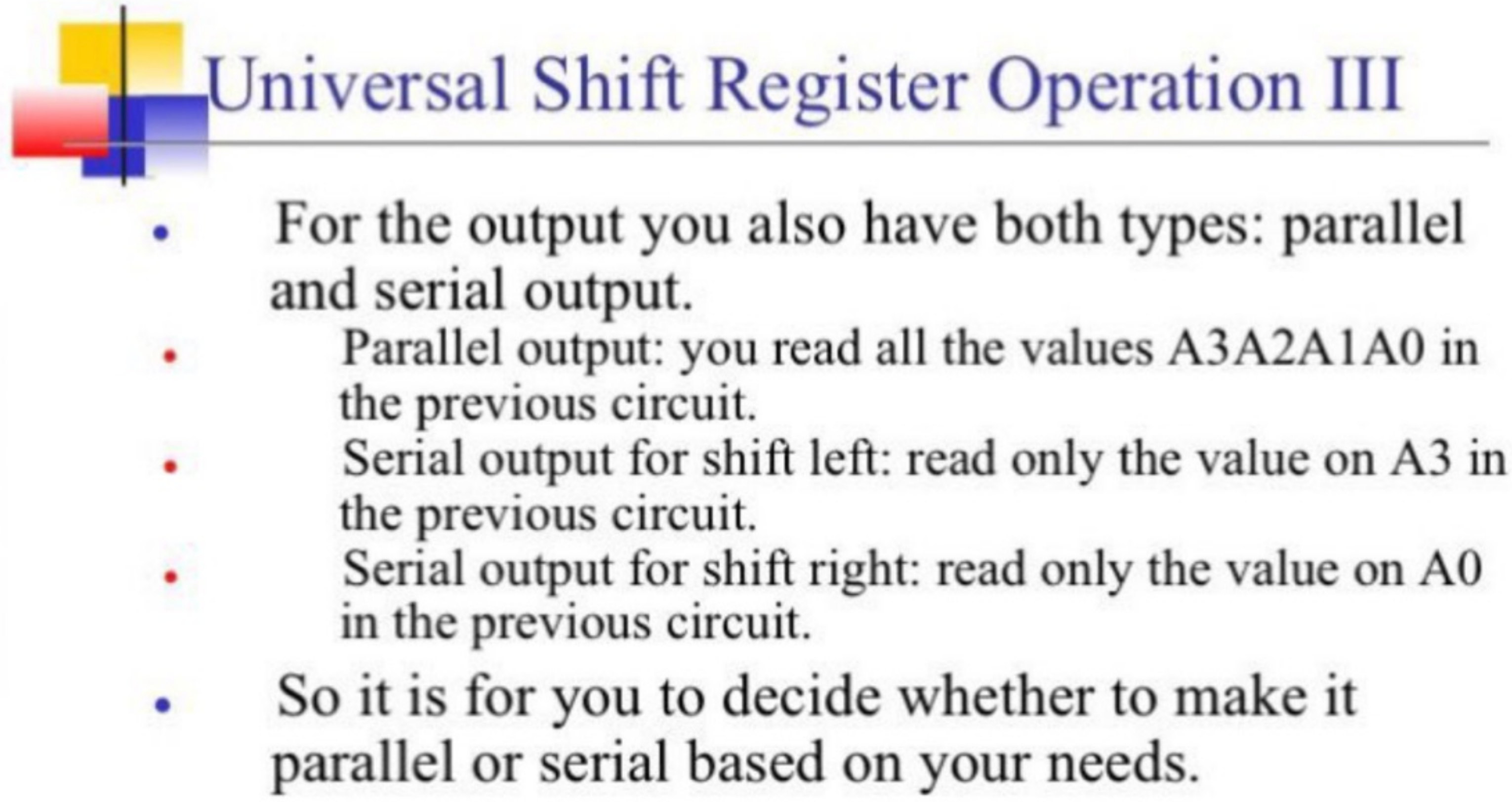
- Values of the selector lines and the mode of register operation based on each option are as follows:

S1	S0	
0	0	□ no change
0	1	□ shift right
1	0	□ shift left
1	1	□ parallel load

Universal Shift Register Operation II

- As you can deduce from the circuit, the mode of operation of the universal register is based on the values of the selector lines of the multiplexers.
- Modes of operation:
  - No change: means no operation, i.e. you want to store the data as it is in the register.
  - Shift left: serial input to shift the data to the left direction.
  - Shift right: serial input to shift the data to right direction.
  - Parallel load: load the input values from the parallel inputs and store them in the register.

Scanned with CamScanner



Universal Shift Register Operation III

- For the output you also have both types: parallel and serial output.
  - Parallel output: you read all the values A3A2A1A0 in the previous circuit.
  - Serial output for shift left: read only the value on A3 in the previous circuit.
  - Serial output for shift right: read only the value on A0 in the previous circuit.
- So it is for you to decide whether to make it parallel or serial based on your needs.

Scanned with CamScanner