Number systems and binary codes

- Binary
- Octal
- Decimal
- Hexadecimal

representation of numbers

signed / unsigned

+8 = 01000
-8 = 11000

8 = 1000 unsigned

MSB: 1 → -ve

bits, $r^n - N$ ← no

Sign magnitude form

eg: 00: +0
01: +1
10: -0
11: -1

Range: $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$

magnitude / sign

1s complement (r-1)'s complement

$+x = +x$
$-x = \overline{+x}$

eg: $+x = 0101 = x$
$-x = 1010 = \overline{x}$

(or)

$-x =$ 1111 − 0101 = 1010

binary	decimal
000	+0
001	+1
010	+2
011	+3
100	−3
101	−2
110	−1
111	−0

Range: $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$

2s complement (r)'s complement

$+x = +x$
$-x = \overline{+x} + 1$
= ones complement + 1 (LSB)

+7 = 0111
−7 = 1000 + 0001 = 1001

+8 = 01000
−8 = 10111 + 00001 = 11000

−8 = 1000 {can be represented in 4-bits?

binary	decimal
00	+0
01	+1
10	−2
11	−1

Range: -2^{n-1} to $+(2^{n-1}-1)$

If $a_3a_2a_1a_0 = N$ (2s complement)

$a_3a_3a_3a_2a_1a_0 = N$

$a_3a_2a_1a_0\,0$ = +ve 2N, −ve 2N

$a_3a_2a_1a_0\,1$ = +ve 2N+1, −ve 2N+1

If $a_3a_2a_1a_0 = N$ (1s complement)

$a_3a_3a_3a_2a_1a_0 = N$

$a_3a_2a_1a_0\,0$ = +ve 2N, −ve 2N+1

$a_3a_2a_1a_0\,1$ = +ve 2N+1, −ve 2N

$(15.625)_{10} \rightarrow (?)_2$

2	15	→ 1
2	22	→ 0
2	11	→ 1
2	5	→ 1
2	2	→ 0
2	1	→ 1

0.625 × 2 =	1.	250
0.250 × 2 =	0.	5
0.5 × 2 =	1	0

Ans: $(101101.101)_2$

$X = X_{MSB}\,X\,\ldots\,X_{LSB}$
$Y = Y_{MSB}\,Y\,\ldots\,Y_{LSB}$
$Z = Z_{MSB}\,Z\,\ldots\,Z_{LSB}$

Overflow = $\overline{X}_{MSB}\,Y_{MSB}\,Z_{MSB}$ + $X_{MSB}\,Y_{MSB}\,\overline{Z}_{MSB}$ → 1 overflow, → 0 no overflow

OVERFLOW

when 2 +ve/2 -ve nos are added and if the result is -ve or +ve, it mean overflow has occured

consider 5 bit as -9 represented in 5 bit

−4 = 1100
−5 = 1011
10111 → Ans ✗ because of OVERFLOW

−4 = 11100
−5 = 11011
11011 1
Ans: 10111 = −9
Same issue when +4 + 5

+4 = 00100
+5 = 00101
01001 = +9

Note

need min 5 bit to represent (−10) in 1s complement

1s complement

x = −4: 1011
y = +3: 0011
MSB 1 ⇒ -ve: 1110
Ans = −(1110)
= −(0001)
= −1

x = +4: 0100
y = −3: 1100
if carry generated: 10000 → 1
0001
Ans = +1

(r-1)s: $(62)_{10}$ → 9s complement: 99 − 62 = 37

(r)s: $-x = \overline{+x} + 1 = y$
$\overline{+x} = y - 1$
$x = \overline{y-1}$

eg: 10
$\overline{+x}$ = 10 − 01
= 01
x = −(01) = −(10)
= −2
−3 = 101
to represent it in 4 bits
−3 = 1101

A = +4 = 0100
B = −3 = 1101
10001
carry neglected
A − B = 0001
→ final carry: INDICATOR. 1 → +ve 2s, 0 → -ve 2s

2s: 2 − 9
0010
0111
1001
0: +ve overflow
∴ (1001) → 2s → 0111

BCD

defined from 0000 to 1001

BINARY CODES

- weighted / non weighted (8421, BCD)
- self complementary / non self complementary (8421, XS-3)
- Cyclic (Gray code) — consecutives w/ 1 bit change

Types of weighted codes: 8421 / 2421 — —

✱✱ If sum of all weights = 9, self complementary code

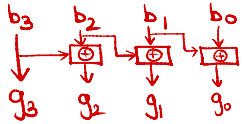
Decimal	XS-3
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

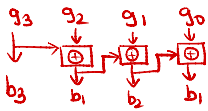
$(12)_{XS3} = (01000101)$

Gray code

Dec	Code
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111

Binary	Gray
000	000
001	001
010	011
011	010
100	110
101	111
110	101
111	100





even parity generator $= x_1 \oplus x_2 \oplus x_3 \oplus x_4$, checker $= x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus P_c$ → 1 error, → 0 no error

odd $= \overline{x_1 \oplus x_2 \oplus x_3 \oplus x_4}$, checker $= \overline{x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus P_0}$ → 1 error, → 0 no error

5 code: even parity

↳ $5C_2$ ways.

00011
00101
00110
01001
01010
01100
10001
10010
10100
11000

MULTIPLICATION

hexadecimal

36 → 16(2)+4

33+2 → 35 → 16(2)+3

32+2 → 32

$$\begin{array}{r} {}^{2}A\,{}^{2}BC \\ \times\ 23 \\ \hline 2034 \\ 15780 \\ \hline 17784 \end{array}$$

$$\begin{array}{r} 94 \\ \times\ 12 \\ \hline 128 \\ +940 \\ \hline A68 \end{array}$$

18 → 16+2

ANALOG vs DIGITAL

↳ minimizes the effect of noise (unwanted signal)

- continuous
- takes ANY value within given limit
  - ↳ DTS (discrete time signal)
    - independant axis (x) is recorded @ values marked by lines, rest of y-vs assumed.

$\Delta t = t_1 - t_0 = t_2 - t_1$

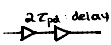
Analog in → ADC (multibit) ⇒ DIGITAL SYSTEM ⇒ DAC → Analog out

Logic gates and boolean algebra

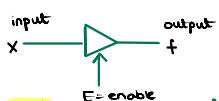
i/p	o/p f_1	f_2	f_3	f_4
0	0	0	1	1
1	0	1	0	1
	0	x	x	1

f_3 → NOT, f_4 → BUFFER

BUFFER — $2\tau_{pd}$: delay

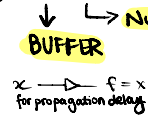


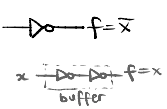
TRISTATE BUFFER



enable	input	output
0	0	High Z
0	1	High Z
1	0	0
1	1	1

when enable = 1, it acts as a buffer
enable = 0, no output
HIGH IMPEDANCE STATE





n variables: $2^{(2^{n-1})}$ Self duals

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

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

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

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

(logical expression)
No. of boolean functions for n-variables $= 2^{2^n}$

n: no of levels
m: no of switch

$n = 2^m$
↳ no of min/max terms

Duality → self dual

one times dual gives same expression

$x \cdot 0 = 0$	$x + 1 = 1$
$x \cdot \bar{x} = 0$	$x + \bar{x} = 1$
$x + 0 = x$	$x \cdot 1 = x$

eg. $G = AB + BC + AC$
$G = (A+B)(B+C)(A+C)$
$= (B+AC)(A+C)$
$G = AB + AC + BC$

$A + B = B + A$ ↶ dual
$A \cdot B = B \cdot A$

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

$(A')' = A$

$A \cdot A = A$
$A \cdot 0 = 0$
$A \cdot 1 = A$
$A \cdot A' = 0$

$x \cdot 0 = x \cdot 0 + 0 = x \cdot 0 + x \cdot \bar{x} = x \cdot (0 + \bar{x}) = x \cdot \bar{x} = 0$

$x + 1 = (x+1) \cdot 1 = (x+1) \cdot (x+\bar{x}) = x + \bar{x} = 1$

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

$x \cdot x = (x \cdot x) + 0 = x \cdot x + x \cdot \bar{x} = x \cdot (x + \bar{x}) = x \cdot 1 = x$

$x + x = (x+x) \cdot 1 = (x+x) \cdot (x+\bar{x}) = x + x\bar{x} = x + 0 = x$

$(A+B)' = A' \cdot B'$

$(A \cdot B)' = A' + B'$

$(x+y)(x' \cdot y') = 0 \Rightarrow xx'y + yxy = 0$

$(x+y) + (\bar{x} \cdot \bar{y}) = 1 \Rightarrow (x + \bar{x} + y) \cdot (x + y + \bar{y})$
$\Rightarrow (1+y) \cdot (1+x) = 1 \cdot 1 = 1$

REDUNDANCY THEOREM

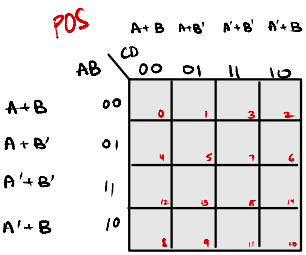
$Y = AB + A'C + BC$

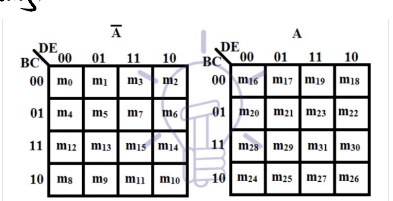
$Y = AB + A'C$

$Y = AB + A'C + BC(A+A')$
$= AB + ABC + A'C + A'BC$
$= AB(1+C) + A'C(1+B)$
$= AB + A'C$

EPI: group of 1s (largest) w/ at least one 1 that CANNOT be paired/combined in any other way.

POS



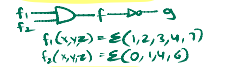


$f(x,y,z) = \Sigma(2,6,7) + \pi(1,3,6,7)$

$= \Sigma(2,6,7) + \Sigma(0,2,4,5)$

$= \Sigma(0,2,4,5,6,7)$

$= \pi(0,1,3)$



$f_1(x,y,z) = \Sigma(1,2,3,4,7)$

$f_2(x,y,z) = \Sigma(0,1,4,6)$

$\therefore f = \Sigma(1,4)$

$g = \pi(1,4)$

$f = \Sigma(0,1,2,3,4,6,7)$

$g = \pi(0,1,2,3,4,6,7)$

MINIMAL → CANONICAL

0. no. of total variables
2. identify the variables absent in each min term
3. use property $1 = A + A'$, $A \cdot 1 = 1$ to add missing min terms

SOP minterms

eg $y = A + B'C$

$y = A(B+B')(C+C') + (A+A')B'C$

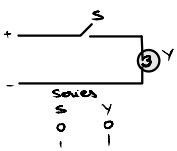
POS maxterms

eg. $F = (A+B+C') \cdot (A'+C)$

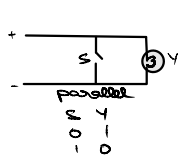
$= (A+B+C') \cdot (A'+C+BB')$

$F = (A+B+C) \cdot (A'+B+C) \cdot (A'+B'+C)$

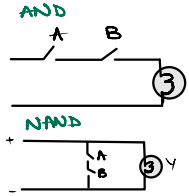
BUFFER



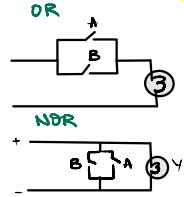
NOT



AND

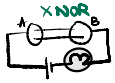


OR



XOR

XNOR



ODD NO of input XOR = XNOR

EVEN NO complement

$A \oplus A = 0$

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

$A \oplus 0 = A$

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

$A \oplus A \oplus A = A$

$A \oplus A \oplus \ldots \oplus A = A$ n-odd

$= 0$ n-even

$A \odot A = 1$

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

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

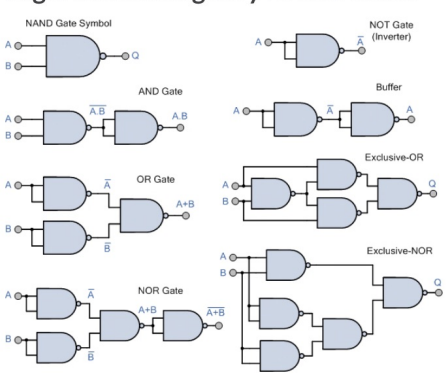
$A \odot 1 = A$

$A \odot A \odot A = A$

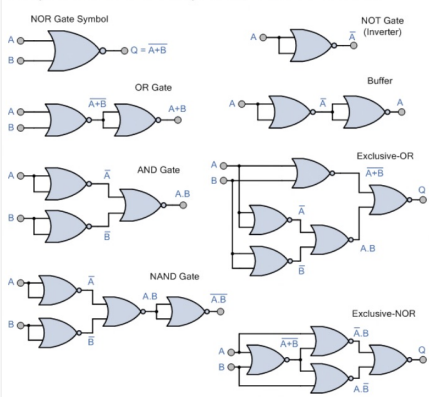
$A \odot A \odot \ldots \odot A = A$ n: odd

$= 1$ n: even

Logic Gates using only NAND Gates



Logic Gates using only NOR Gates



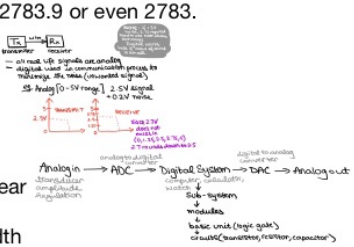
motherboard → subcircuit in a chip → LOGIC GATES → switch circuit → transistor circuit → transistor on a chip

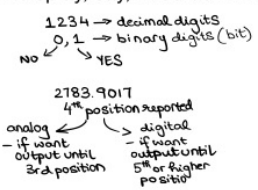
What does digital mean?

- Very simply, it means that the device or the system works with digits, numbers, discrete quantities.
- These numbers are usually, at least in the interior of the device, binary numbers—0s and 1s.
- We usually see the results as "people" numbers—decimal, most likely—and alphanumeric characters, including letters and digits and punctuation.

So what's the opposite of digital? Analog.

- This means that the range of values being processed is continuous.
- A good example is in your car. The speedometer is an analog device because the pointer can point absolutely anywhere on the scale (unless you happen to have a fully digital display as some cars do today).
- The odometer, on the other hand, is digital because it can display only numbers in a fixed range. Miles to the nearest integer, or perhaps to the nearest tenth. It can't display, say, 2783.9017 miles but instead displays 2783.9 or even 2783.





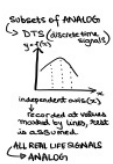
Digital preferred

- Quality of service
- Maintenance
- Flexibility
- Upgradability
- Exponential vs Linear
- Delay
- Uses less bandwidth
- Higher efficiency for long distance transmission.
- Encryption: at transmission end we encrypt the intended data user decodes using proper key.

TRANSDUCER—converts non electrical signal to electrical

What is a signal?

- Function that represents variation of a physical quantity with a parameter.
- Function that is dependent (varies with respect to a parameter) on a parameter.
- In electrical/electronics signal represents variation of electrical quantity (voltage/current with time).



	Analog	Digital
Signal	Analog signal is a continuous signal which represents physical measurements.	Digital signals are discrete time signals generated by digital modulation.
Waves	Denoted by sine waves	Denoted by square waves
Representation	Uses continuous range of values to represent information	Uses discrete or discontinuous values to represent information
Example	Human voice in air, analog electronic devices.	Computers, CDs, DVDs, and other digital electronic devices.
Technology	Analog technology records waveforms as they are.	Samples analog waveforms into a limited set of numbers and records them.
Data transmissions	Subjected to deterioration by noise during transmission and write/read cycle.	Can be noise-immune without deterioration during transmission and write/read cycle.
Response to Noise	More likely to get affected reducing accuracy	Less affected since noise response are analog in nature
Flexibility	Analog hardware is not flexible.	Digital hardware is flexible in implementation.
Uses	Can be used in analog devices only. Best suited for audio and video transmission.	Best suited for Computing and digital electronics.
Applications	Thermometer	PCs, PDAs
Bandwidth	Analog signal processing can be done in real time and consumes less bandwidth.	There is no guarantee that digital signal processing can be done in real time and consumes more bandwidth to carry out the same information.
Memory	Stored in the form of wave signal	Stored in the form of binary bit
Power	Analog instrument draws large power	Digital instrument draws only negligible power
Errors	Analog instruments usually have a scale which is cramped at lower end and give considerable observational errors.	Digital instruments are free from observational errors like parallax and approximation errors.

← both parameters are discrete.

Lower marked limit is estimated.
eg: 1,2,3
1.5 → assumed as 1

Top–down approach

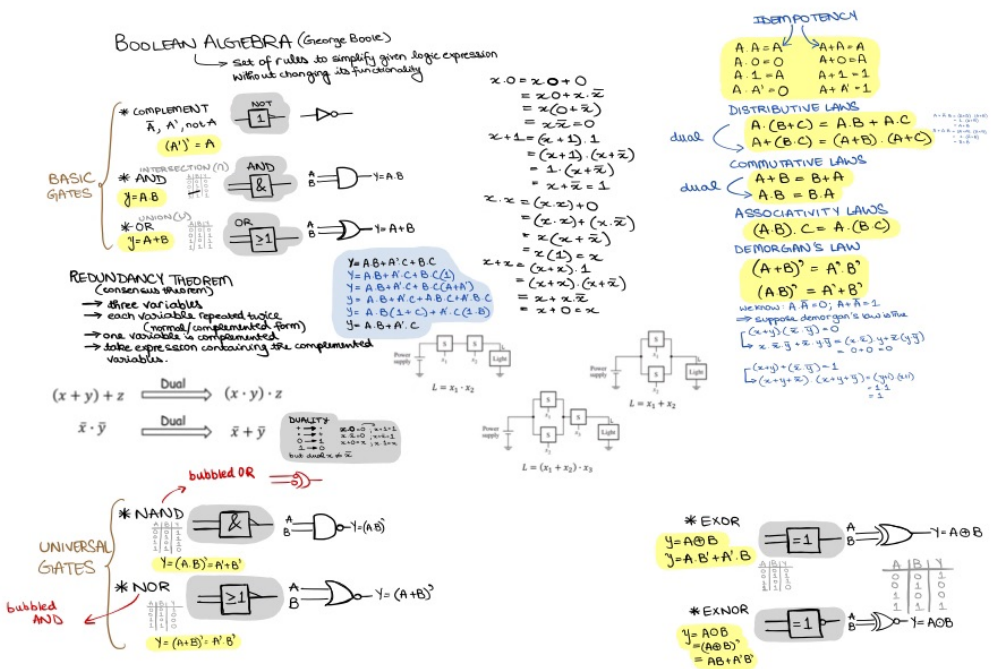
• The top-down approach involves looking at the big picture and breaking that picture up into smaller constituents.
• Each of these smaller parts is a complete entity, which will combine with all these other parts to make the finished whole.
• But the smaller constituents may still be too large for one person or a small design team to handle.
• So we break them up again, getting even smaller constituents.
• Eventually, we'll have these constituents small enough that we can handle them, one by one, fairly easily.

Bottom–up approach

• If we are careful to break the problem at clear boundaries between subsystems, we will find that we have small constituents whose integrity we can be sure of.
• Then we can test each constituent completely, making certain that it does its job 100% correctly.
• When these are assembled into the larger system they were broken out from, the larger system should also work correctly.

WHAT'S A DIGITAL SYSTEM?

• A digital system is one that involves discrete quantities, most often binary digits – digits that are two valued viz. 0 and 1
• What we need to know now is what kinds of things we will find in digital systems.
• The basic units are gates, which are devices that combine binary signals in fixed, simple ways.
• Let us now look at some of the basic operations (gates) that we have.



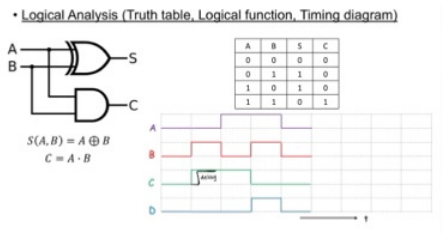
When the truth table for any two or more networks are the same, then they are called as functionally equivalent networks.

• Ideally, we should use the network having lowest cost (say, no of gates, delay, area, power etc.)

— — —

• As you probably now know, combinational circuits are made up of gates and fancier circuits that contain just gates.

• Sequential circuits are made up of both combinational circuits and devices that can “remember” such as flip-flops.



LOGIC CIRCUIT FAMILIES

• Logic circuits come in families.

• The members of a logic family are compatible to the nth degree, which is what the family is all about.

• All the members of a logic family have very similar characteristics.

• All require the same power supply voltages.

• All respond to the same logic signal levels.

• All produce the same signal levels. All of them operate at similar speeds.

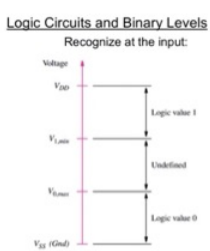
• The members of a logic family are designed to work together simply.

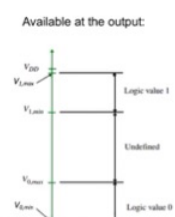
• In most cases, a signal is moved from one member to another “just by running wires.” You don’t have to pay attention to signal levels, voltages, or drive currents.

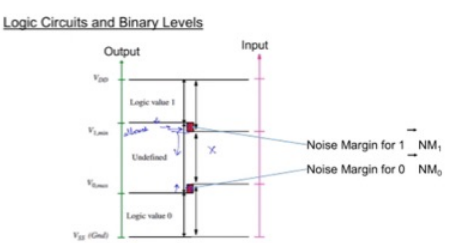
• You do, however, have to consider timing, because a signal that arrives with too much delay may not cause the desired operation.

• Two families are quite common because they are simple, cheap, and versatile.

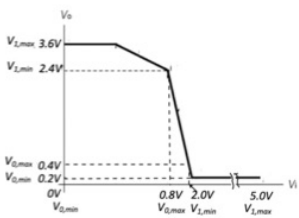
• They are also fairly forgiving of wiring errors.



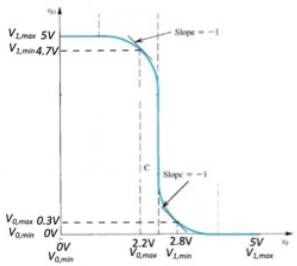




TTL: Transistor Transistor Logic:



CMOS: Complimentary MOS Logic:



TTL Family

• The transistor–transistor logic family (TTL) has been around for over 70 years. The two Ts in the name tell a little about the physical arrangement of the internal components.
• Part labels generally begin with a number, either 74 (commercial grade parts) or 54 (military grade).
• Within the TTL family are several subfamilies, separated by such characteristics as power consumption and speed.
• The particular subfamily is designated by letters after the initial number. Two examples are 74LS series (low-power Schottky), which are simple, require relatively small amounts of power, and operate at moderate speed, fast enough for many applications (such as vending machine logic).
• 74F series (fast), which are faster than the 74 LS series. These, and others that are even faster, may found in such systems as printers.
• The blank after the letters is filled in with two to four digits. But the digits are the same for standard circuits. For example, 74LS00 and 74F00 are both chips containing four two-input NAND gates.
• The TTL family typically requires a 5-volt power supply.

CMOS Family

• The CMOS logic family (complementary metal-oxide semiconductor) is a very low- power family of chips.
• Typical power consumption when the chip isn't doing anything is in the microwatt range.
• The members of this family require essentially zero power except during switching, which means that their average power consumption depends on how many times the output switches per second. In other words, power consumption is proportional to frequency.
• There are three subfamilies here:
• 40 series, which was the original family. Its devices are moderately fast.
• 74HC series (high-speed CMOS), which is popular and has many different types of devices available. But it, like the 40 series, will not interface easily with the members of the TTL family. Yet we sometimes need to do this because the CMOS families do not have as many varieties of logic devices.
• 74HCT (high-speed CMOS, TTL compatible), which resolves that problem. But here too, the variety is not very extensive.
• As with the TTL families, standard numbering is used. So a 4000, a 74HC 00, and a
• 74HCT 00 all have four NAND gates on one chip.
• CMOS has another advantage—lower power supply voltages. While 5 volts is still common, 3.3 volts, 3 volts, and even less than 3 volts are available.

• Product term is a term where the variables are connected by and operators, terms such as $A \cdot B$, $C \cdot D \cdot E$, and even the simple term F.

• Sum term is a term where the variables are connected by or operators, terms such as $A + B$, $\overline{C} + D + E$, and even the simple term F.

• Literal is any appearance of a variable in a term, whether primed or not. In the terms above, the literals are $A, B, \overline{C}, D, E$, and F. If the variable appears both primed and unprimed, as in G and $\overline{G}$, each is a literal.

• Sum-of-products is an expression consisting of product terms linked by or operators $A \cdot B + \overline{C} \cdot D \cdot E + F$.

• Product-of-sums is an expression consisting of sum terms linked by and operators $(A + B) \cdot (\overline{C} + D + E) \cdot F$.

SUM OF PRODUCTS (SOP)

for when output is high

A : 1
A' : 0

minterms

binary		A	B	C	F	
0	m_0	0	0	0	0	
1	m_1	0	0	1	0	
2	m_2	0	1	0	1	→ A'.B.C'
3	m_3	0	1	1	0	
4	m_4	1	0	0	1	→ A.B'.C'
5	m_5	1	0	1	1	→ A.B'.C
6	m_6	1	1	0	1	→ A.B.C'
7	m_7	1	1	1	1	→ A.B.C

F = A'.B.C' + A.B'.C' + A.B'.C + A.B.C' + A.B.C

F(A,B,C) = $m_2 + m_4 + m_5 + m_6 + m_7$

= $\sum m(2,4,5,6,7)$

{CANONICAL/STANDARD SOP form}

F = A'.B.C' + A.B'.C' + A.B'C + A.B.C' + A.B.C CANONICAL

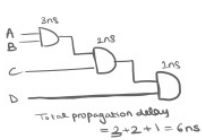
F = A'.B.C' + A.B'.(C' + C) + A.B(C' + C)

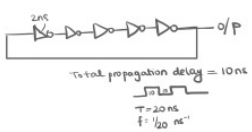
= A'.B.C' + A.(B' + B)

= A'.B.C' + A

F = A + B.C' MINIMAL

{A + Ā.B = A + B}





binary		A	B	y	
m_0		0	0	0	
m_1		0	1	1	A'.B
m_2		1	0	0	
m_3		1	1	1	A.B

y = A'.B + A.B

= B.(A' + A)

y = B

y = $\sum m(1,3)$

PRODUCT OF SUM (POS)

for when output is low

A : 0
A' : 1

binary	A	B	C	F
M_0	0	0	0	0
M_1	0	0	1	0
M_2	0	1	0	1
M_3	0	1	1	0
M_4	1	0	0	1
M_5	1	0	1	1
M_6	1	1	0	1
M_7	1	1	1	1

A+B+C

A+B+C'

A+B'+C'

derive POS from SOP
↳ SOP for low

$Y = (A'.B'.C')' . (A'.B'.C)' . (A'.B.C)'$
$= (A+B+C) . (A+B+C') . (A+B'+C')$

$Y = (A+B+C) . (A+B+C') . (A+B'+C')$

$Y = \Pi M(0,1,3)$

{CANONICAL/STANDARD POS form}

$Y = (A+B+C) . (A+B+C') . (A+B'+C')$

$(X+Y) . (X+Z) = (X+Y.Z)$

$Y = (A+B+C.C') . (A+B'+C')$
$= A+B.(B'+C')$
$= A+B.B'+B.C'$
$Y = A+B.C'$
$Y = (A+B) . (A+C')$

{minimal POS form}

SOP form ⟶ no. of AND gates > OR
POS form ⟶ no. of OR gates > AND

eg.

min terms
$Y(A,B,C) = \Sigma m(0,2,3,6,7)$

max terms
$Y(A,B,C) = \Pi M(1,4,5)$

$2^{(2^{n-1})}$ self duals

2^n = min/max terms ; n variables

$2^{(2^n)}$ = logical expression ; n variables

Self-dual
→ for any logical expression two times dual gives same expression
→ self dual : one time dual gives same expression

POSITIVE and NEGATIVE LOGIC

→ higher voltage corresponds to logic 1
→ lower voltage corresponds to logic 0

→ higher voltage corresponds to logic 0
→ lower voltage corresponds to logic 1

positive and negative logic AND gate

A	B	Y
L	L	L
L	H	L
H	L	L
H	H	H

L→0, H→1 +ve logic AND

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

L→1, H→0 -ve logic NAND

A	B	Y
1	1	1
1	0	1
0	1	1
0	0	1

positive and negative logic OR gate

A	B	Y
L	L	L
L	H	H
H	L	H
H	H	H

L→0, H→1 +ve logic OR

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

L→1, H→0 -ve logic NOR

A	B	Y
1	1	1
1	0	0
0	1	0
0	0	0

OBSERVATION
+ve logic AND gate = -ve logic OR gate
+ve logic OR gate = -ve logic AND gate

Definitions for Logic Synthesis:

- **Normal term:** Product or sum term in which no literal appears more than once, in any form.

$$F_1(x,y,z) = (x+y) + \bar{x}\cdot\bar{z}$$

- **Minterm:** For a function of n variables, minterm is a normal product term with n literals.

$$F_2(x,y,z) = \bar{x}\cdot\bar{y}\cdot z + \bar{x}\cdot y\cdot z + x\cdot\bar{y}\cdot\bar{z} + x\cdot\bar{y}\cdot z$$

- **Maxterm:** For a function of n variables, maxterm is a normal sum term with n literals.

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

SOP → CSOP

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

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

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

$f(a,b,c) = a + \bar{b}\cdot c$

$= a\cdot(b+\bar{b})\cdot(c+\bar{c}) + (a+\bar{a})\bar{b}\cdot c$

$= a\cdot b\cdot c + a\cdot b\cdot\bar{c} + a\cdot\bar{b}\cdot c + a\cdot\bar{b}\cdot\bar{c} + a\bar{b}\cdot c + \bar{a}\,\bar{b}\,c$

$= a\cdot b\cdot c + a\cdot b\cdot\bar{c} + a\cdot\bar{b}\cdot c + a\cdot\bar{b}\cdot\bar{c} + \bar{a}\cdot\bar{b}\cdot c$

POS → CPOS

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

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

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

$f(a,b,c,d) = a + d$

$= a + d + b\cdot\bar{b} + c\cdot\bar{c}$

$= (a+d+b\cdot\bar{b}+c)\cdot(a+d+b\cdot\bar{b}+\bar{c})$

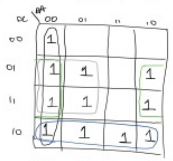
$= (a+b+c+d)\cdot(a+\bar{b}+c+d)\cdot(a+b+\bar{c}+d)\cdot(a+\bar{b}+\bar{c}+d)$

COMPLEMENT

$z(C,B,A) = \Sigma m(1,3,5,6,7) = \Pi m(0,2,4)$

$\bar{z}(C,B,A) = \Sigma m(0,2,4)$

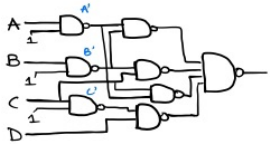
$f(D,C,B,A) = \Sigma m(0,4,5,6,8,9,10,11,12,13,14)$



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

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

$(f')' = \left[(A'.B')'.(C'.D)'.(B'.C)'.(A'.C)'\right]'$



Logic Synthesis

- Implement function using only NOR gates

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

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

Logic Synthesis:

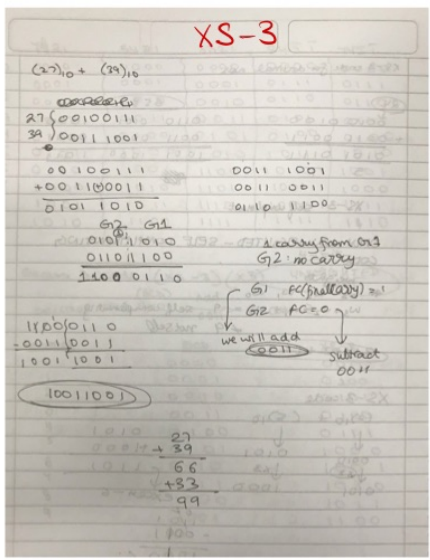
- Implement function using NOT, OR and AND gates in POS

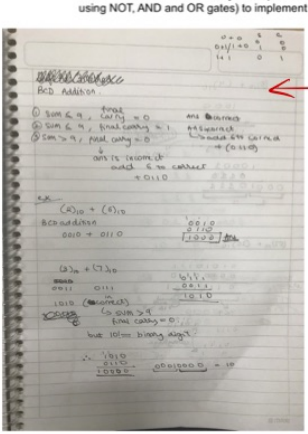
$$f(D,C,B,A) = \prod M(0,2,3,4,8,10,11,12,13,15)$$

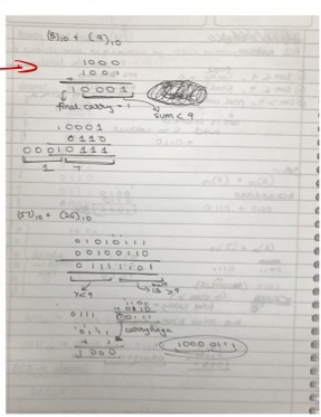
$f(D,C,B,A) = (D+C+B+A)\cdot(D+C+\bar{B}+A)\cdot(D+\bar{C}+B+A)\cdot(D+\bar{C}+\bar{B}+A)\cdot(\bar{D}+C+B+A)\cdot(\bar{D}+C+\bar{B}+A)\cdot(\bar{D}+\bar{C}+B+A)\cdot(\bar{D}+\bar{C}+\bar{B}+A)$

This reduces to $f = (A+B)(C+\bar{B})(\bar{A}+C+D)$

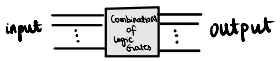
Take over from here and carry out a process (like what we did for implementation using NOT, AND and OR gates) to implement using NOT, OR and AND gates







Combinational logic circuits

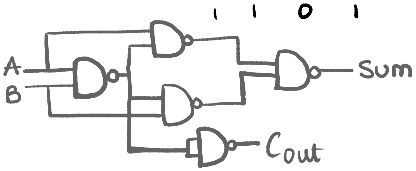


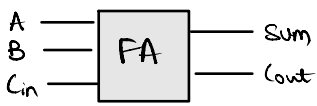
Adding n-bit
+ n-bit
carry+n-bit

1-bit adder / subtractor
- HA: 2 single bit is added
- FA: 3 single bit is added
- HS: 2 single bit subtractor
- FS: 3 single bit subtractor

A	B	Sum	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Sum = A⊕B
Cout = A.B

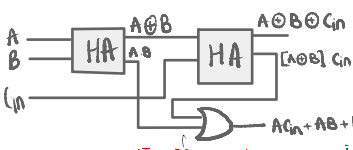




A	B	Cin	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Sum = A⊕B⊕Cin
= A⊙B⊙Cin
Cout = AB + BCin + ACin

checkboard on kMAP.



$A\bar{B}C_{in} + \bar{A}BC_{in} + AB(C_{in} + \bar{C}_{in})$
$A\bar{B}C_{in} + \bar{A}BC_{in} + ABC_{in} + AB\bar{C}_{in}$
$AC_{in} + \bar{A}BC_{in} + AB(C_{in} + \bar{C}_{in})$
$AC_{in} + AB + AB(C_{in} + \bar{C}_{in})$
$AC_{in} + AB + BC_{in}$

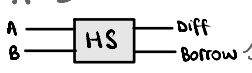
if A.B = 1, Cout = 1

~~XX~~ don't have to wait for second FA's output.
CRUCIAL for delay problems.

MAJORITY INPUT GATES
eg - AND / OR
0 : 0
1 : 1

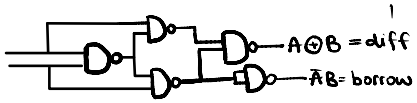
Sum delay = 2 x XOR
Carry delay = 1 x XOR + 1 AND + 1 OR

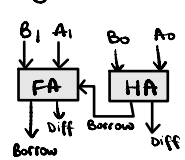
A - B

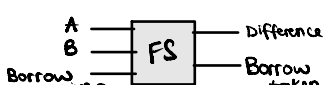


A	B	Diff	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Diff = A⊕B
Borrow = $\bar{A}$.B

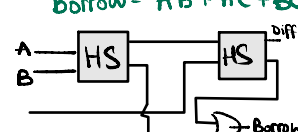




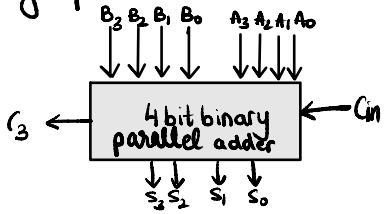


A	B	Cin	Diff	Borrow
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

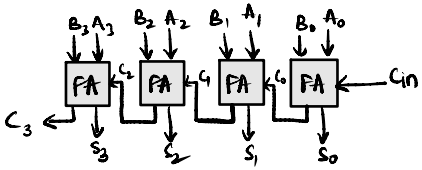
Diff = A⊕B⊕C = A⊙B⊙C
Borrow = $\bar{A}$B + $\bar{A}$C + BC

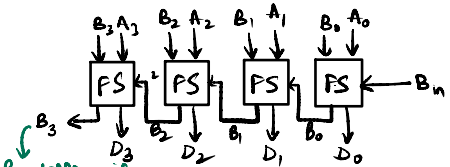


Binary parallel adder

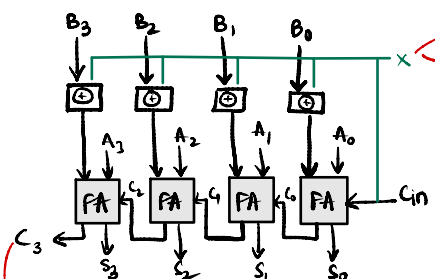


	C_2	C_1	C_0	Cin
A:	A_3	A_2	A_1	A_0
B:	B_3	B_2	B_1	B_0
	C_3	S_3	S_2	S_1	S_0





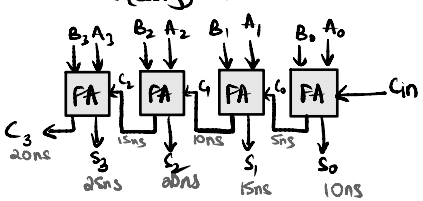
B_3: tells wif A>B or A<B



C_3 = 0 : −ve ans } in subtraction, C_3: neglected
C_3 = 1 : +ve ans }

FA are identical
→ PD (sum) = 10ns
→ PD (carry) = 5ns

worst case delay (n-bit) = (n−1) Carry delay + max (CD/SD)
(CD: carry delay; SD: sum delay)



DECIMAL ADDER (1 bit FA)

BCD ADDER

1 dec digit = 0 to 9

2nd dec digit = 0 to 9

Cn = 0/1

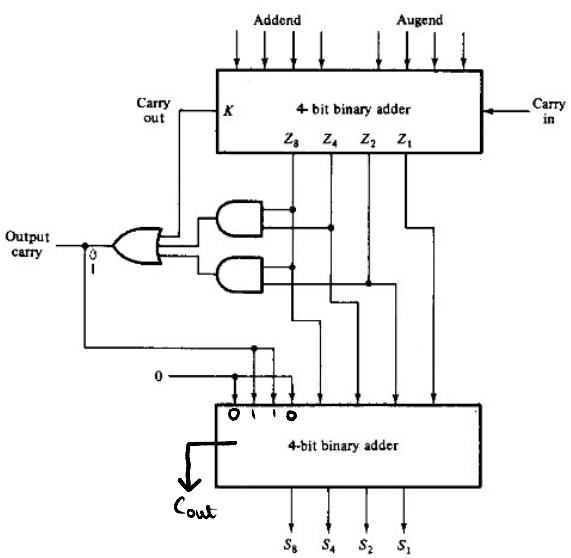
Max sum = 19

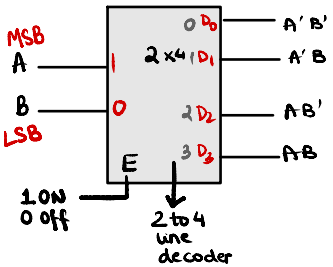
TABLE 5-1
Derivation of a BCD Adder

Binary Sum					BCD Sum					Decimal
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

binary sum, same as BCD

+6=



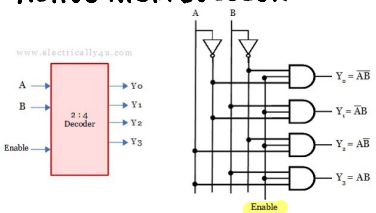


+ OR gate → SOP

In one time, only one output in decoder circuit.
↳ if that ONE output = 1 rest = 0, Active High.

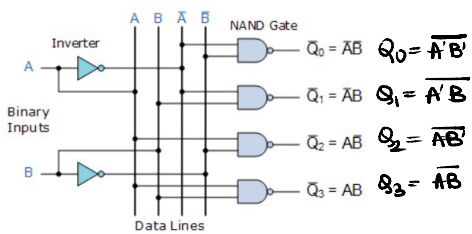
if $\bar{E}$ is used E = 0: enable
E = 1 : disable

ACTIVE HIGH DECODER



Inputs			Outputs			
EN	A	B	Y_3	Y_2	Y_1	Y_0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

ACTIVE LOW DECODER.

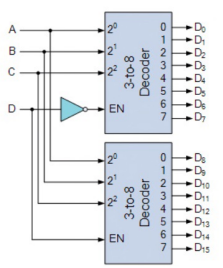


2-to-4 NAND Gate Binary Decoder

Truth Table

E	A	B	Q_0	Q_1	Q_2	Q_3
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	x	x	1	1	1	1

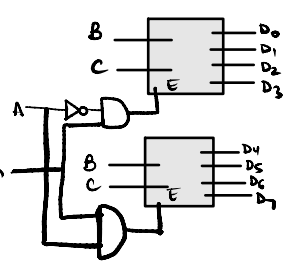
A 4-to-16 Binary Decoder Configuration

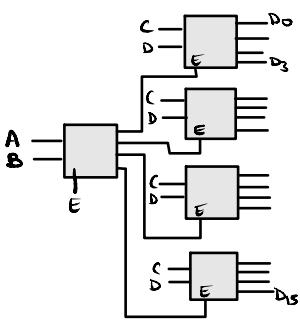


4-to-16 Line Decoder Implemented with two 3-to-8 Decoders

3×8 Decoder using 2×4 decoder

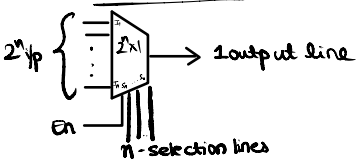
A	B	C	
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

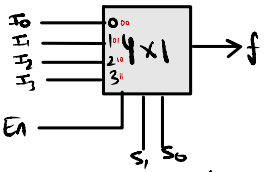




MUX

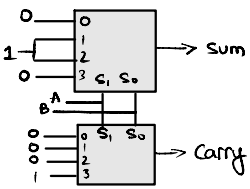
$2^n \times 1$ MUX

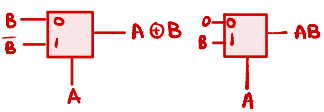


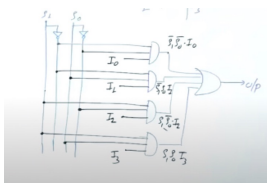


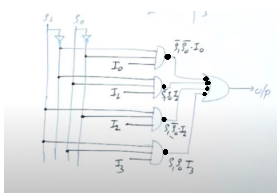
E	S_1	S_0	o/p
1	0	0	I_0
1	0	1	I_1
1	1	0	I_2
1	1	1	I_3
0	x	x	high Z (no ans)

$f = (\bar{S_1}\bar{S_0}I_0 + \bar{S_1}S_0I_1 + S_1\bar{S_0}I_2 + S_1S_0I_3).En$







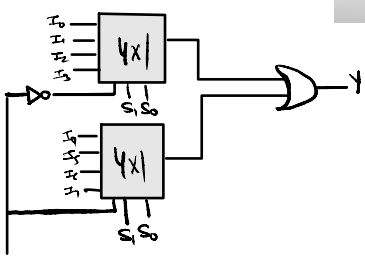


8×1 MUX implemented using 4×1.

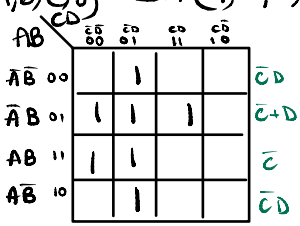
$8/4 = 2$

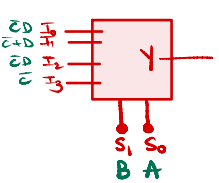
$2/4 = 0.5$

S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

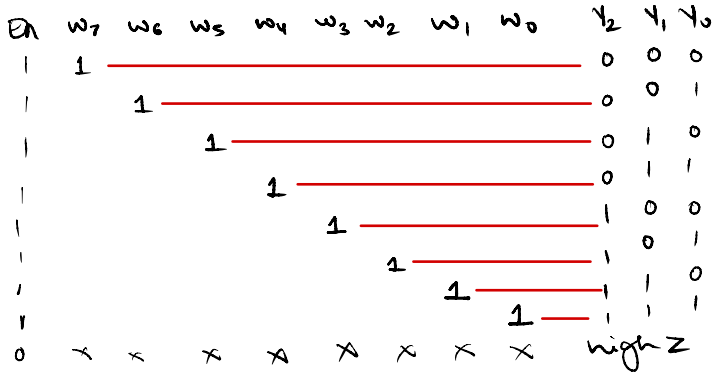


$F(A,B,C,D) = \Sigma m\,(1, 4, 5, 7, 9, 12, 13)$





Encoder (n → m line) $m = \log_2 n$



Active high

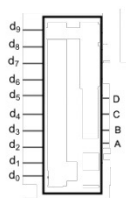
Active low: Singular zeros instead of 1s

10:4 Decimal-Binary Encoder:

$m = [\log_2 10] = [3.322] = 4$

BCD

d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	D	C	B	A
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1



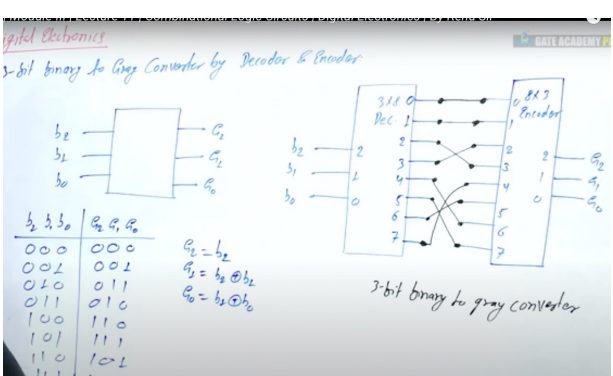
Priority Encoders:

w_3	w_2	w_1	w_0	y_1	y_0	z
0	0	0	0	d	d	0
0	0	0	1	0	0	1
0	0	1	ϕ	0	1	1
0	1	ϕ	ϕ	1	0	1
1	ϕ	ϕ	ϕ	1	1	1

$$z = w_3 + w_2 + w_1 + w_0$$

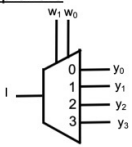
$$y_1 = w_3 + w_2$$

$$y_0 = w_3 + \overline{w_2} w_1$$



DEMUX

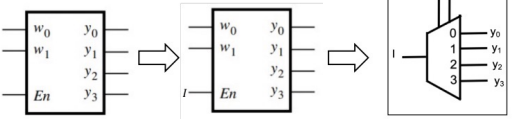
Demultiplexers:



w_1	w_0	y_0	y_1	y_2	y_3
0	0	I	0	0	0
0	1	0	I	0	0
1	0	0	0	I	0
1	1	0	0	0	I

$y_0 = \overline{w_1} \cdot \overline{w_0} \cdot I$
$y_1 = \overline{w_1} \cdot w_0 \cdot I$
$y_2 = w_1 \cdot \overline{w_0} \cdot I$
$y_3 = w_1 \cdot w_0 \cdot I$

Realizing Demultiplexer using a Decoder:



$y_0 = \overline{w_1} \cdot \overline{w_0} \cdot E_n$
$y_1 = \overline{w_1} \cdot w_0 \cdot E_n$
$y_2 = w_1 \cdot \overline{w_0} \cdot E_n$
$y_3 = w_1 \cdot w_0 \cdot E_n$

$y_0 = \overline{w_1} \cdot \overline{w_0} \cdot I$
$y_1 = \overline{w_1} \cdot w_0 \cdot I$
$y_2 = w_1 \cdot \overline{w_0} \cdot I$
$y_3 = w_1 \cdot w_0 \cdot I$

$y_0 = \overline{w_1} \cdot \overline{w_0} \cdot I$
$y_1 = \overline{w_1} \cdot w_0 \cdot I$
$y_2 = w_1 \cdot \overline{w_0} \cdot I$
$y_3 = w_1 \cdot w_0 \cdot I$

A_2, B_2	A_1, B_1	A_0, B_0	A>B	A<B	A=B
$A_2 > B_2$	×	×	1		
$A_2 < B_2$	×	×		1	
$A_2 = B_2$	$A_1 > B_1$	×	1		
$A_2 = B_2$	$A_1 < B_1$	×		1	
$A_2 = B_2$	$A_1 = B_1$	$A_0 > B_0$	1		
$A_2 = B_2$	$A_1 = B_1$	$A_0 < B_0$		1	
$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$			1

XNOR for A=B

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

$\therefore f = (A_2 \odot B_2) \cdot (A_1 \odot B_1) \cdot (A_0 \odot B_0)$

A>B

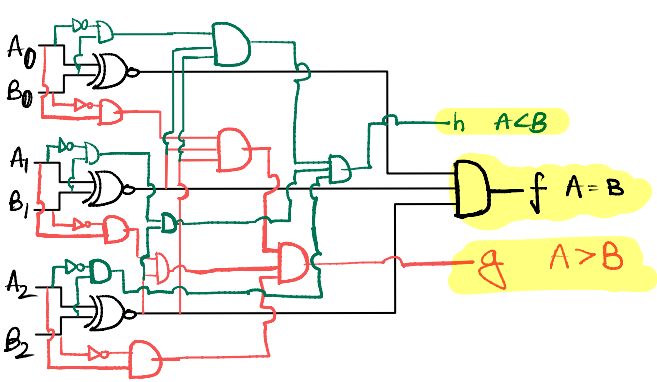
↳ $A_2 > B_2$ (OR) $A_2 = B_2, A_1 > B_1$ (OR) $A_2 = B_2, A_1 = B_1, A_0 > B_0$

$g = (A_2 \cdot \overline{B_2}) + (A_2 \odot B_2) \cdot (A_1 \cdot \overline{B_1}) + (A_2 \odot B_2)(A_1 \odot B_1)(A_0 \overline{B_0})$

$\therefore A < B$

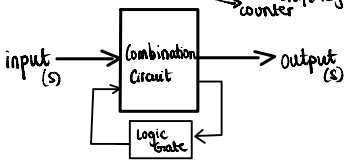
$A_2 < B_2$ (or) $A_2 = B_2, A_1 < B_1$ (OR) $A_2 = B_2, A_1 = B_1, A_0 < B_0$

$h = (\overline{A_2} \cdot B_2) + (A_2 \odot B_2)(\overline{A_1} \cdot B_1) + (B_2 \odot A_2)(A_1 \odot B_1)(B_0 \overline{A_0})$



Sequential circuits

- latch
- flip-flop
- Registers
- Shift registers
- counter

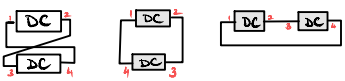


Combinational : present input depends only on present input

Sequential : present output depends on present & previous input

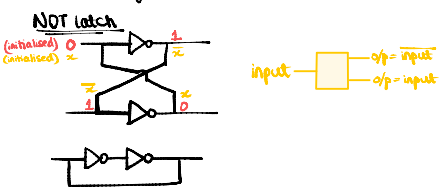
Latch

Structure

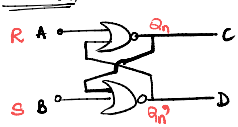


DC → NOT gate → NOT latch
- → NAND gate → NAND latch
- → NOR gate → NOR latch

NOT latch



NOR latch



present output = Q_n
next output = Q_{n+1}
R — DC — Q_n
S — DC — Q_n'

R	S	Q_n	Q_{n+1}	
0	0	0	0	} Q_n
0	0	1	1	
0	1	0	0	} Reset
0	1	1	0	
1	0	0	1	} Set
1	0	1	1	
1	1	0	forbidden	
1	1	1	forbidden	

A	B	C	D	
0	0	Q_n	Q_n'	memory
0	1	1	0	set
1	0	0	1	reset
1	1	0	0	← FORBIDDEN

} SR Latch

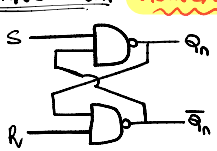
S \ RQn	00	01	11	10
0	0	1	0	0
1	1	1	x	x

$Q_{n+1} = S + \bar{R} Q_n$
characteristic eqn

Excitation table

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

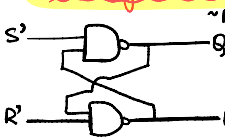
NAND latch — Active low S-R latch



S	R	Q_{n+1}	
0	0	forbidden	
0	1	1	set
1	0	0	reset
1	1	Q_n	memory

Active High S-R latch

~NAND latch w/ inverted input



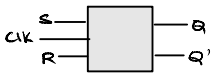
S' — DC — Q_n
R' — DC — Q_n'

Latch → Asynchronous circuit (w/out time, w/out clock)

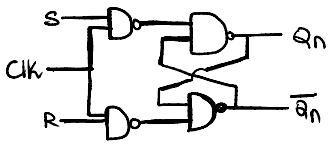
Clocked latch → synchronous circuit (flip flop {FF})

CLOCKED S-R Latch

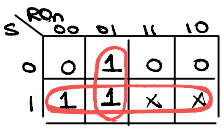
S-R FF (flip flop)
- 2NOR + 2AND
- 4NAND



S-R Latch
- 2NOR gate
- 2NAND gate + 2 NOT gate



Clk	S	R	Q_{n+1}	
1	0	0	Q_n	memory
1	0	1	0	reset
1	1	0	1	set
1	1	1	forbidden	
0	x	x	Q_n	memory

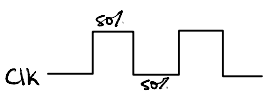


Clk = 1

Clk	S	R	Q_n	Q_{n+1}
1	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	1
1	1	1	0	x
1	1	1	1	x
0	x	x	0	0
0	x	x	1	1

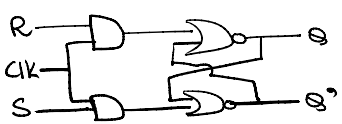
$Q_{n+1} = S + \overline{R}\,Q_n$ {Clk = 1}

$Q_{n+1} = Q_n$ {Clk = 0}

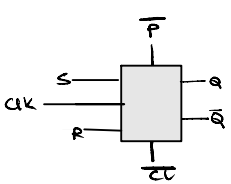


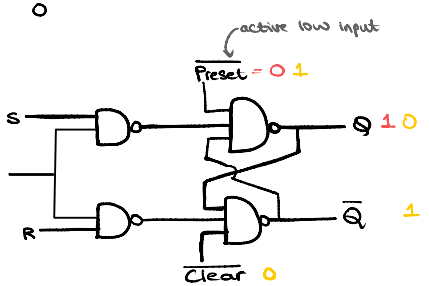
duty cycle = 50%

$f_{clk} = 1/T_{clk}$



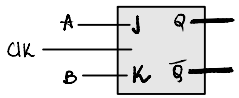
Q_n	Q_{n+1}	S	R
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0



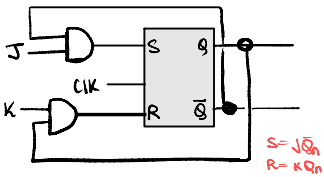


S	R	$\overline{Preset}$	$\overline{Clear}$	F/F operation
x	x	0	0	FORBIDDEN
x	x	0	1	Q=1, FF set
x	x	1	0	Q=0, FF clear
S	R	1	1	Normal FF operation

J-K F/P



clk	J	K	Q_{n+1}
↓	0	0	Q_n
↓	0	1	0
↓	1	0	1
↓	1	1	$\overline{Q_n}$
0	X	X	Q_n



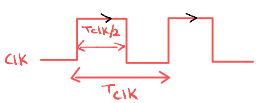
$S = J\overline{Q_n}$

$R = KQ_n$

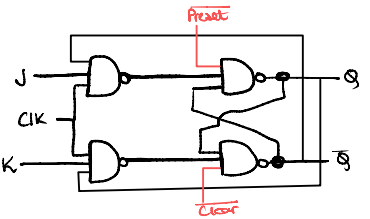
$$Q_{n+1}\big|_{S-R} = S + \overline{R}\,Q_n$$

$$Q_{n+1} = J\overline{Q_n} + (KQ_n)'\,Q_n$$
$$= J\overline{Q_n} + (K' + Q_n')\,Q_n$$

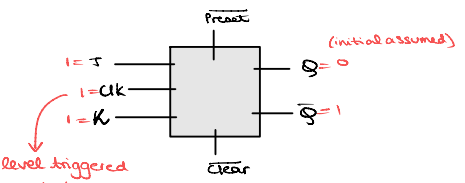
$$Q_{n+1} = J\overline{Q_n} + \overline{K}\,Q_n$$



Race around problem in level triggered clock JK ff.

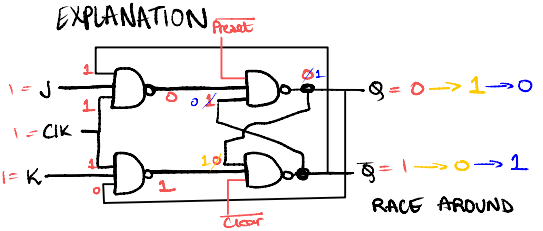


TOGGLE



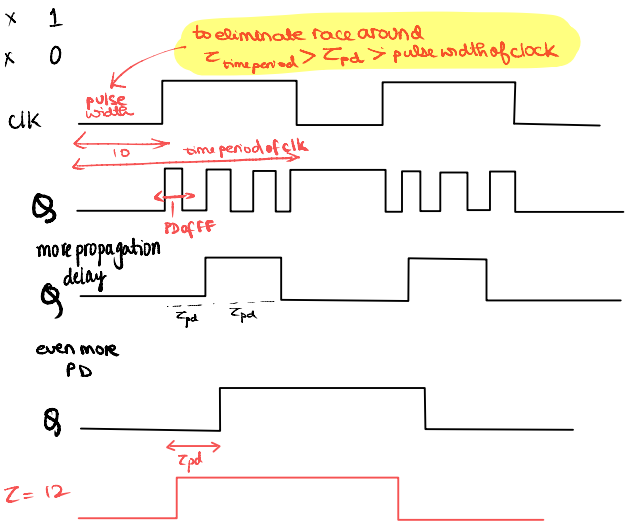
CLK	J	K	Q_n	Q_{n+1}
1	1	1	0	1, 0, 1, 0, ...
1	1	1	1	0, 1, 0, 1, ...

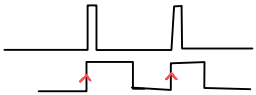
EXPLANATION



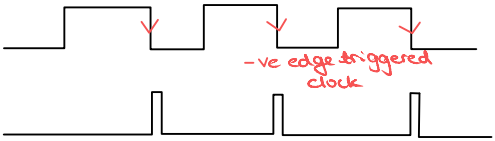
Excitation table

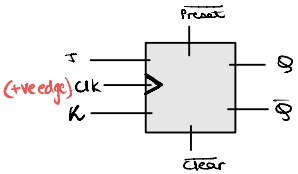
Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

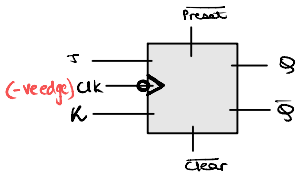


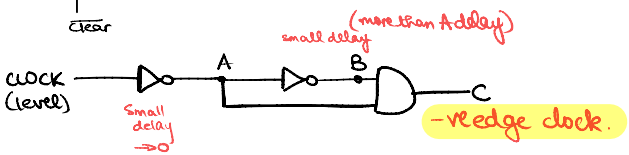


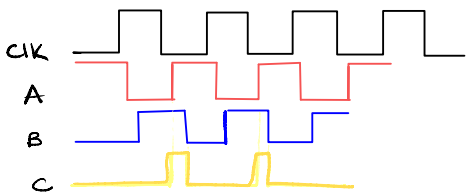
+ve edge triggered clock/
rising edge clock

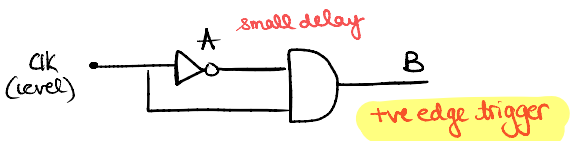


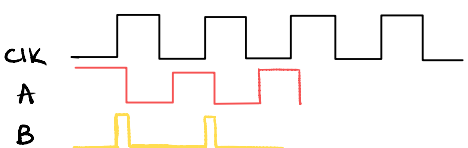






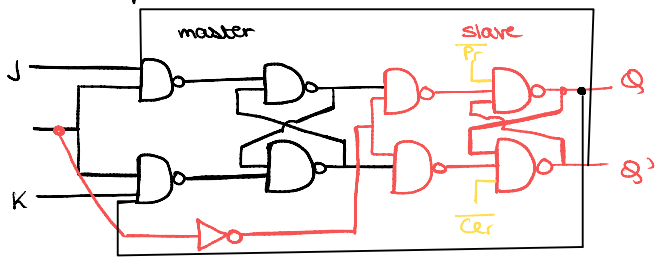


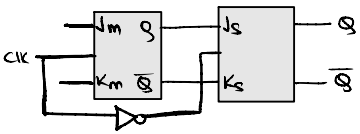


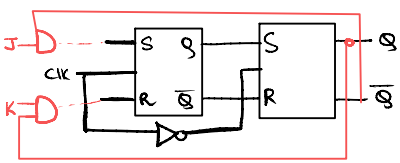


Master-slave J-K/F-F

works as a -ve edge trigger FF



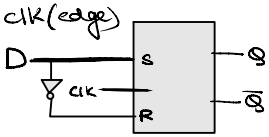




for const input in SR, edge & level have no diff

D/latch if clk (level)

D/FF if clk (edge)

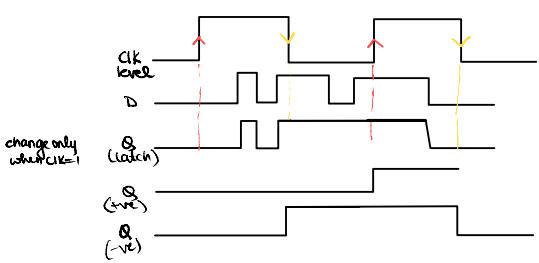


S = D

R = $\overline{D}$

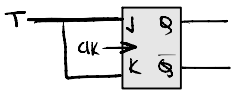
charac-eqn data FF

$Q_{n+1} = D$



T F/F

Toggle flip flop



$J = K = T$

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

$Q_{n+1} = T \oplus Q_n$

$Q_{n+1} \oplus Q_n = T$

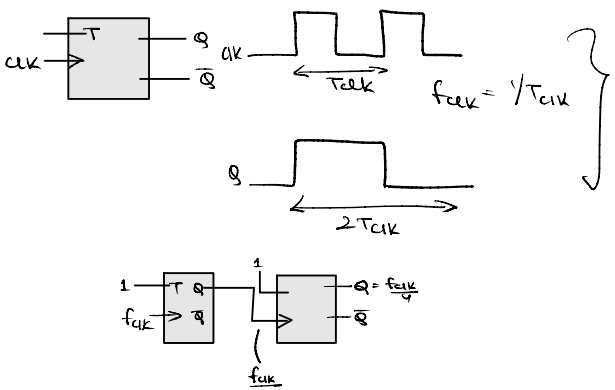
$Q_{n+1}|_{SR} = S + \overline{R}Q_n$

$Q_{n+1}|_{JK} = J\overline{Q}_n + \overline{K}Q_n$ → to construct Register (memory element)

$Q_{n+1}|_{D} = D$

$Q_{n+1}|_{T} = T \oplus Q_n$ → to implement counters

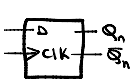
→ frequency divide by 2^n



Design one FF using another

D F/F available

→ implement J/K

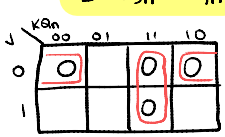


$D = f(J, K, Q_n)$

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

$Q_{n+1} = D$

$D = J\overline{Q}_n + \overline{K}Q_n$



$D = (J + Q_n) \cdot (\overline{K} + \overline{Q_n})$

→ implement T

$D = T \oplus Q_n$

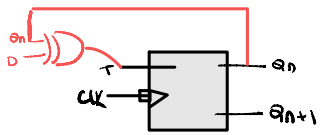
→ implement SR

$D = S + \overline{R}Q_n$

can't implement SR. 11, are shown invalid in approach.

T → D

Q_n	D	Q_{n+1}	$T = Q_n \oplus D$
0	0	0	0
0	1	1	1
1	0	0	1
1	1	1	0



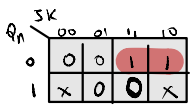
SR → JK.

available : SR

required : JK

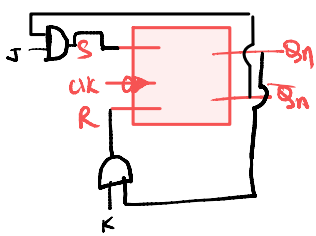
Q_n	J	K	Q_{n+1}	S	R
0	0	0	0	0	X
0	0	1	0	0	X
0	1	0	1	1	0
0	1	1	1	1	0
1	0	0	1	X	0
1	0	1	0	0	1
1	1	0	1	X	0
1	1	1	0	0	1

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



$S = \overline{Q_n} \cdot J$

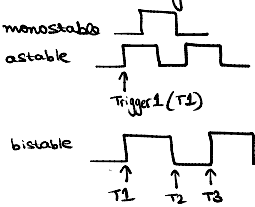
$R = Q_n K$

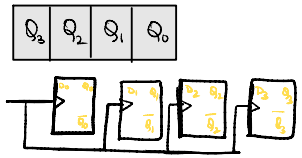


Register : 4bit Register → 4F/F : 4-D/FF

1F/F : store 1 bit binary info

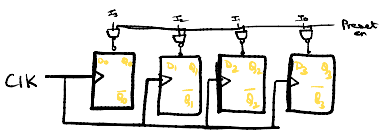
: 1 bit memory, bistable multivibrator





parallel in, parallel out Register

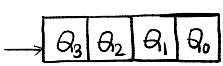
$D_3D_2D_1D_0$ = 1 1 1 1
then apply clock
↳ $Q_3Q_2Q_1Q_0$ = 1 1 1 1
↓
stored until we give clock again

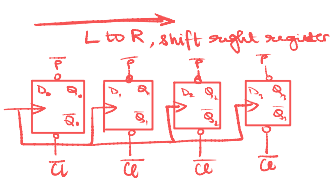


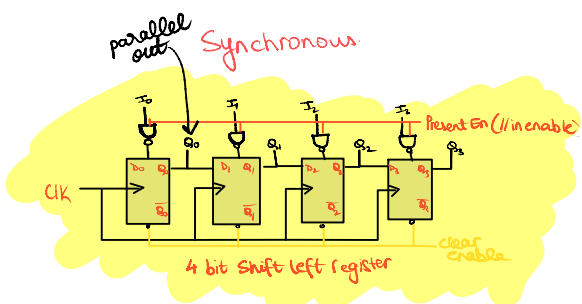
M1 $Q_3Q_2Q_1Q_0$ = x x x x
preset en = 0
Put D_3 D_2 D_1 D_0 = 1 1 1 1
∴ apply clk
$Q_3Q_2Q_1Q_0$ = 1 1 1 1

M2 initial $Q_3Q_2Q_1Q_0$ = 0000 (clr)
$D_3D_2D_1D_0$ = x x x x
$I_3I_2I_1I_0$ = 1 1 1 1
preset en = 1 1 1 1
$Q_3Q_2Q_1Q_0$ = 1 1 1 1

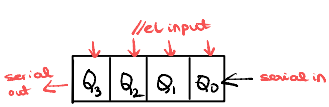
Shift Register → left / right







shift register
- → //el in //el out
- → //el in serial out
- → serial in //el out
- → serial in serial out



① parallel in, parallel out {0 clock is needed}

clear $Q_3 Q_2 Q_1 Q_0 = 0\ 0\ 0\ 0$

input data @ $I_3 I_2 I_1 I_0 = 1010$

② parallel in, serial out {n-1 clock is needed for n bit}

for output, only Q_3 can be accessed

$Q_3 Q_2 Q_1 Q_0 = 1\ 0\ 1\ 0$

clk	1	10 1 x
	2	10 x x
	3	1 x x x

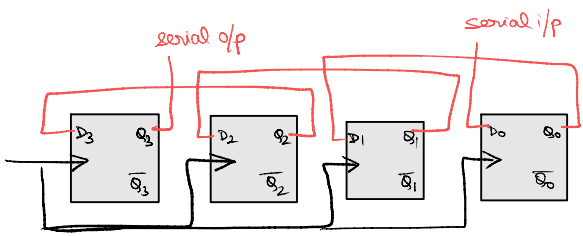
③ serial in, //el out {n clock needed}

1 1 1 0

$Q_3 Q_2 Q_1 Q_0$

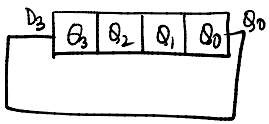
	Q_3	Q_2	Q_1	Q_0
initial	x	x	x	x
Put $D_0=1$, clk=1	1	x	x	x
$D_1=1$, clk=2	1	1	x	x
$D_2=1$, clk=3	1	1	1	x
$D_3=1$, clk=4	0	1	1	1

④ serial in, serial out {n + n-1 clocks}



Application of shift register

1. for parallel to serial data conversion
2. for serial to parallel data conversion
3. used to multiply w/ 2^n {Shift Left Register}
4. used to divide w/ 2^n {Shift Right Register}
5. As a ring counter



serial output & input connected.

{ MAX possible count by n-bit ring counter → n

2bit ring → cannot be started from all bits same

00, 01, 10, 11

	Q_1	Q_0
start	0	0
clk 1	0	0

only one state

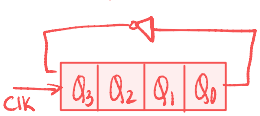
	Q_1	Q_0
start	0	1
clk 1	1	0
clk 2	0	1

2 diff states

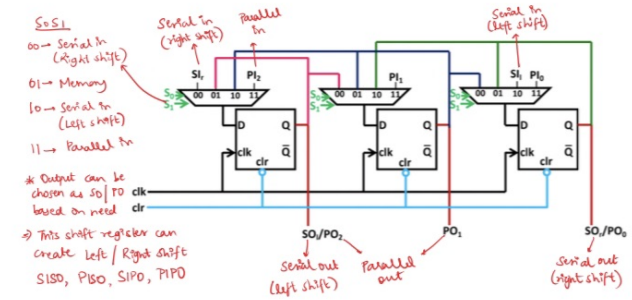
start	1	1
clk 1	1	1

max possible no of used states = 2n

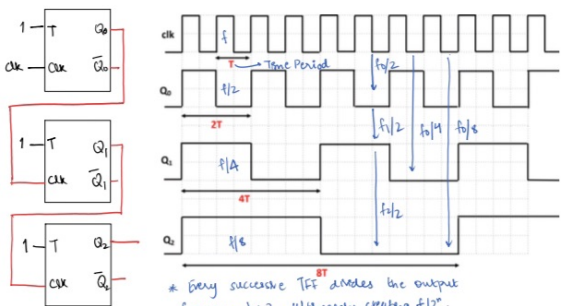
Johnson Counter (twisted ring counter) / (switch tail)
↳ shift register application



	Q_1	Q_0
initial	0	0
clk 1	1	0
2	1	1
3	0	1
4	0	0

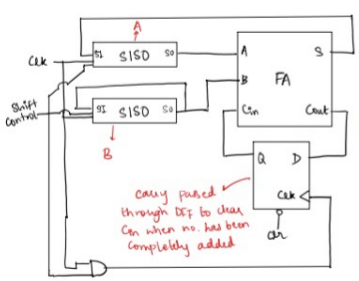


FREQUENCY DIVIDER



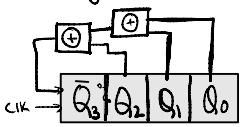
* Every successive TFF divides the output frequency by 2. Ultimately creating f/2ⁿ.

N-BIT ADDER USING SHIFT REGISTERS



* Same shift register is used to store sum & augend wherein sum of each bit is stored at the start of the SISO register.
* Clk AND'ed with shift control activated DFF only when bits are shifted within the shift registers; thus $B_n + A_n$ gets carry from $B_{n-1} + A_{n-1}$

Shift Register as synchronous counter



	Q_3	Q_2	Q_1	Q_0
initial	1	1	1	1
CLK ↓	1	1	1	1

Synchronous : provided to only one F/F (LSB)

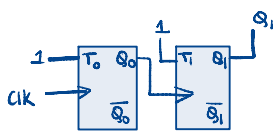
Asynchronous : simultaneously provided to all FF

Counters (T F/F)

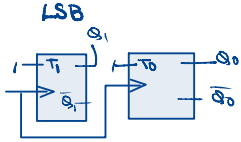
↳ Driven by clock signal, used to count no. of clock cycles. Also known as freq divider circuit

$$f_{o/p} = \frac{f_{CLK}}{N}$$

↳ any integer value



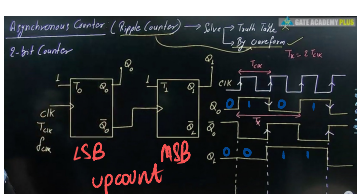
ASYNCHRONOUS Serial/Ripple



SYNCHRONOUS Parallel

up → start 0000 ... 1111

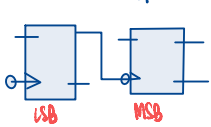
ASYNCHRONOUS

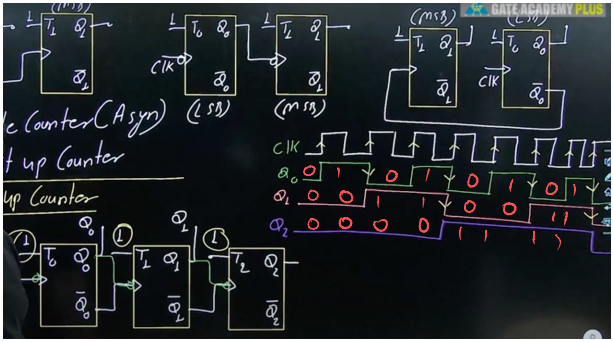


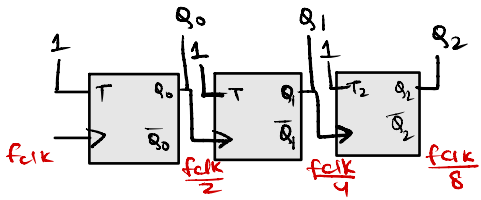
$f_{o/p} = \frac{f_{CLK}}{4}$

Q_1	Q_0
0	0
0	1
1	0
1	1

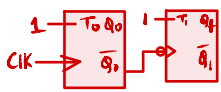
$f_{o/p} = \frac{f_{CLK}}{\text{no. of o/p states}}$

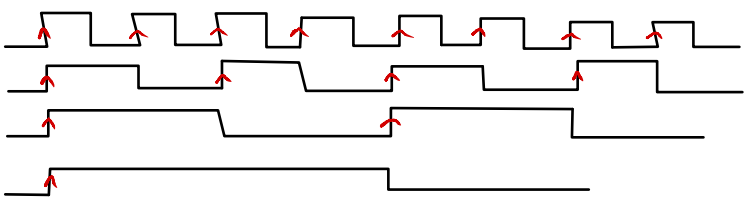






3 bit down counter





MOD-N counter

$T_{clk} \geq$ PD of all the F/F $= n \times$ PD of any one F/F

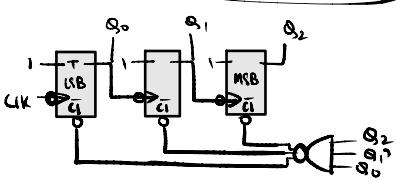
$T_{clk/min}$ = PD of all F/F

$f_{clk} \leq \frac{1}{\text{PD of all the F/F}}$

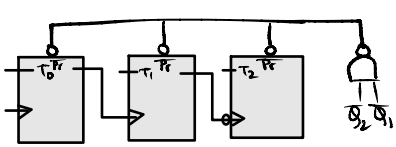
$f_{clk/max} = \frac{1}{n \times \text{PD of any F/F}}$

mod-5

0 → 1 → 2 → 3 → 4

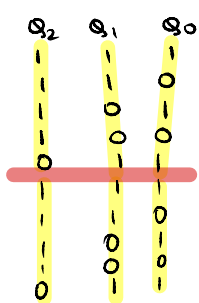


7 → 6 → 5 → 4 → 3 → 2



$f_{Q_2} = f_{Q_1} = f_{Q_0} = \frac{f_{clk}}{5}$

diff waveform



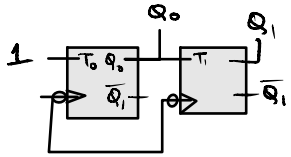
Synchronous Counter

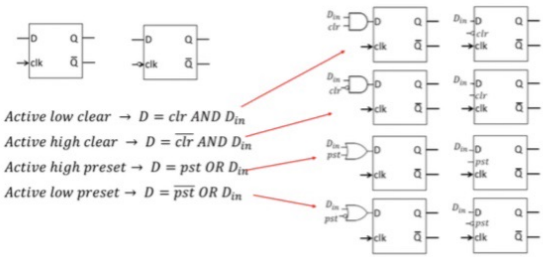
$0 \rightarrow 1 \rightarrow 2 \rightarrow 3$

Q_1	Q_0	Q_1^+	Q_0^+	T_1	T_0
0	0	0	1	0	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	0	0	1	1

$T_0 = 1$

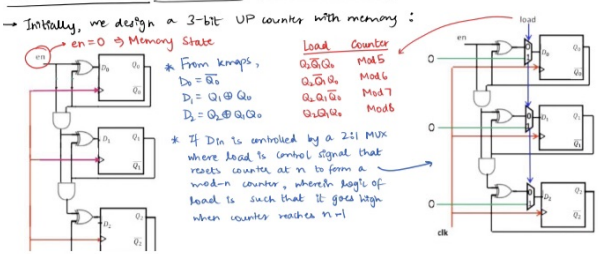
$T_1 = Q_0$





VARIABLE LENGTH SYNCHRONOUS 3-BIT COUNTER

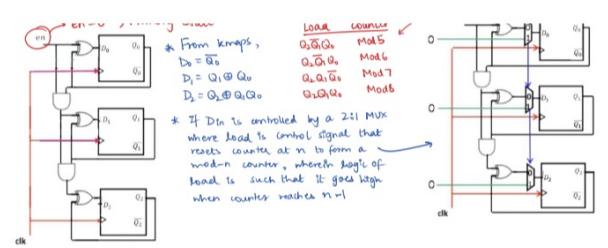
→ Initially, we design a 3-bit UP counter with memory :



* From kmaps,
  $D_0 = \overline{Q_0}$
  $D_1 = Q_1 \oplus Q_0$
  $D_2 = Q_2 \oplus Q_1 Q_0$

Load	Counter
$Q_2\overline{Q_1}Q_0$	Mod5
$Q_2\overline{Q_1}\overline{Q_0}$	Mod6
$Q_2Q_1\overline{Q_0}$	Mod7
$Q_2Q_1Q_0$	Mod8

* If Din is controlled by a 2:1 MUX where load is control signal that resets counter at n to form a mod-n counter, wherein logic of load is such that it goes high when counter reaches n-1



* From kmaps,
  $D_0 = \overline{Q_0}$
  $D_1 = Q_1 \oplus Q_0$
  $D_2 = Q_2 \oplus Q_1 Q_0$

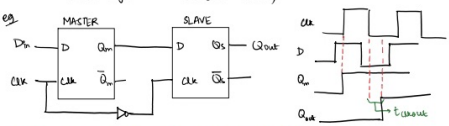
Load	Counter
$Q_2\overline{Q_1}Q_0$	Mod5
$Q_2\overline{Q_1}\overline{Q_0}$	Mod6
$Q_2Q_1\overline{Q_0}$	Mod7
$Q_2Q_1Q_0$	Mod8

* If Din is controlled by a 2:1 MUX where load is control signal that resets counter at n to form a mod-n counter, wherein logic of load is such that it goes high when counter reaches n-1

TIMING DELAY

→ $t_{clk\text{-}out}$: Time taken for final output to see a change after the active clock edge is called ($t_{cm} - t_{cm}$)

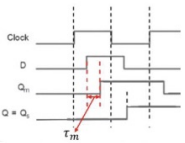
eg



* $t_{clk\text{-}out} \simeq$ Propagation delay of slave latch (T_s)

→ t_{setup}: Time for which Din must remain high before active clock edge of slave latch (i.e. Qm should be stable before slave's active clock edge)

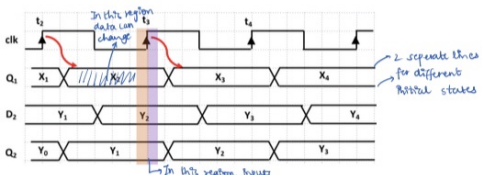
eg For same Master Slave FF above,

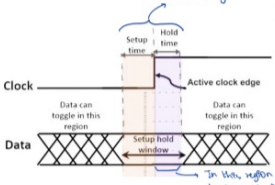


* $t_{setup} \simeq$ Propagation delay of master latch (T_m)

* If input changes in the time period before t_{setup}, the output of slave latch is unpredictable ie. output could be of this value or previous value of input

→ t_{hold}: When the clock signal switches value (for active edge), due to the rise time, there is a delay for the clock to change value; during this time, input must remain constant to avoid ambiguous output (current/prev value)

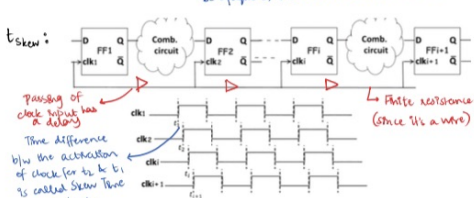




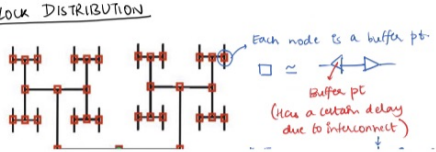
* Reason for "data cannot change":
1. For slave output, we need stable master output
↓
For this we need stable master input during t_{setup}.

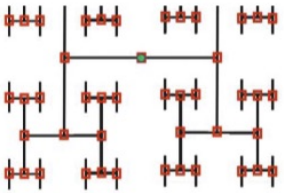
In this region, master is not guaranteed to be opaque ⇒ we need stable master output

→ t_{skew}:



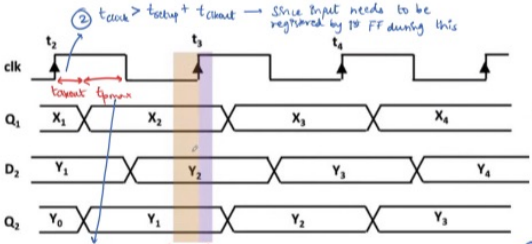
CLOCK DISTRIBUTION





(Has a certain delay due to interconnect)

* For any node, Imagine there were FFs at both ×, then the $t_{interconnect\ delay} = t_{skew}$ for FFs
* This is due to the interconnect having a finite resistance



Possible timing diagram of an arrangement of a combination of 2 FFs

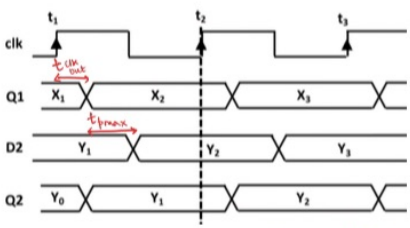
* t_{pmax} → Delay of combinational circuit connecting the 2 FFs

④ Also b/w the 2 FFs, there is a $t_{skew} = t_2 - t_1$

$\Rightarrow t_{clk} > t_{setup} + t_{clkout} + t_{pmax} + t_{skew}$

Maximum logic delay constraint

* This time constraint ensures before t_{setup} is up, the output of the first FF reaches the 2nd FF.
* But this also limits the value of the frequency of the circuit ⇒ circuit can only operate at certain speed.



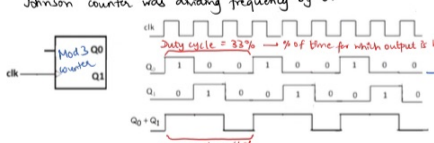
* The delay seen in the change of output at 1st FF and propagation of output to input of 2nd FF through combinational circuit is due to non zero t_{clkout} & t_{pmax}.
* If $t_{clkout} + t_{pmax} = 0$ then the value of D_2 can change in the setup hold window, thus causing an unstable output.
* To overcome this scenario,

Minimum logic delay constraint ← $t_{clkout} + t_{pmin} \geq t_{hold} + t_{skew}$

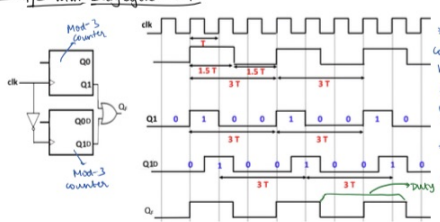
To incorporate clock delay

FREQUENCY DIVISION

→ To build a frequency divider by n, use a mod-n counter similar to how Johnson counter was dividing frequency by 6.



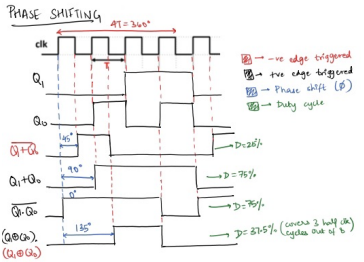
→ f/3 with Duty Cycle = 50%



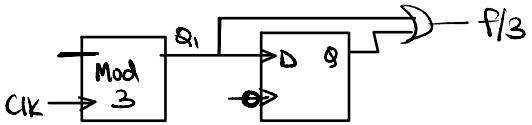
* Output of the combined circuit is high for half the time period of the output i.e. $3T/2 = 1.5T$
⇒ Duty cycle = 50%

* This divider with 50% duty cycle can also be generated using one Mod-3 counter & a DFF (earlier we required 4 DFFs now only 3 DFFs)
* Output of Mod-3 counter can be fed to DFF with opposite active edge thus producing same waveform as Q1D in above timing diagram.

PHASE SHIFTING



* Similarly frequency divider & phase shift can be created for any desired values by using Mod-n counters & taking combinations of input values.

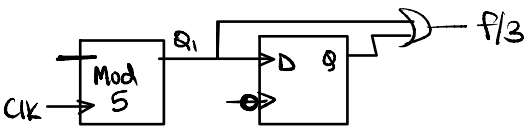


-ve triggered $+T/2$

+ve $+T$

Mod 5

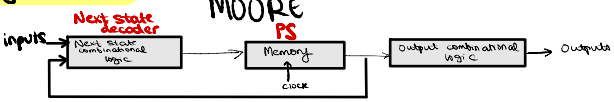
Q_2	Q_1	Q_0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0



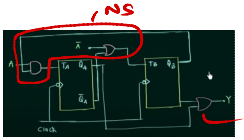
Mealey & Moore → o/p function of Present state only (PS)

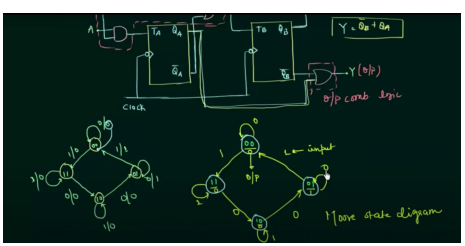
o/p function of PS & Input

MOORE



NS





MEALEY