

Number systems and binary codes

- ↳ Binary
- ↳ Octal
- ↳ Decimal
- ↳ Hexadecimal

representation of numbers

signed unsigned

$$+8 = 01000$$

$$8 = 1000 \text{ unsigned}$$

		MSB: 1 → -ve bits	$r^n - N$ → no
		Sign magnitude form	1's complement
g	00 : +0	$+x = +x$	$(r-1)'s \text{ complement}$
01	+1	$+x = +x$	
10	-0	$-x = +x$	
11	-1	$+x = 0101 = x$ $-x = 1010 = \bar{x}$	

Range:

$$-(2^{n-1} - 1) \text{ to } +(2^{n-1} - 1)$$



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

Range:

$$-(2^{n-1} - 1) \text{ to } +(2^{n-1} - 1)$$

If $a_3a_2a_1a_0 = N$ (2's complement)

$$a_3a_2a_3a_2a_1a_0 = N$$

$$\rightarrow a_3a_2a_1a_0 = \stackrel{\text{+ve}}{\curvearrowleft} 2N$$

$$\rightarrow a_3a_2a_1a_0 = \stackrel{\text{-ve}}{\curvearrowleft} 2N$$

$$\rightarrow a_3a_2a_1a_0 = \stackrel{\text{+ve}}{\curvearrowleft} 2N+1$$

$$\rightarrow a_3a_2a_1a_0 = \stackrel{\text{-ve}}{\curvearrowleft} 2N+1$$

If $a_3a_2a_1a_0 = N$ (1's complement)

$$a_3a_2a_3a_2a_1a_0 = N$$

$$\rightarrow a_3a_2a_1a_0 = \stackrel{\text{+ve}}{\curvearrowleft} 2N$$

$$\rightarrow a_3a_2a_1a_0 = \stackrel{\text{-ve}}{\curvearrowleft} 2N+1$$

$$\rightarrow a_3a_2a_1a_0 = \stackrel{\text{+ve}}{\curvearrowleft} 2N+1$$

$$\rightarrow a_3a_2a_1a_0 = \stackrel{\text{-ve}}{\curvearrowleft} 2N$$

BCD

defined from 0000 to 1001

8021, BCD

→ weighted/non weighted

→ 8021, XS-3

BINARY CODES → self complementary / non self complementary

→ cyclic (Gray code)

consists of 1-bit change

Types of weighted codes: 8421 / 2421 --

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

$$(45.625)_{10} \rightarrow (?)_2$$

$$\begin{array}{r} 2 \\ \times 2 \\ \times 2 \\ \times 2 \\ \times 2 \\ \hline 1 & 1 & 0 & 1 & 0 & 1 \end{array}$$

$$0.625 \times 2 = 1.250$$

$$0.250 \times 2 = 0.5$$

$$0.5 \times 2 = 1.0$$

$$\text{Ans: } (101101.101)_2$$

$$X = X_{\text{MSB}} \dots X_{\text{LSB}}$$

$$Y = Y_{\text{MSB}} \dots Y_{\text{LSB}}$$

$$Z = Z_{\text{MSB}} \dots Z_{\text{LSB}}$$

$$\text{Overflow: } \overline{X}_{\text{MSB}} \cdot Y_{\text{MSB}} \cdot Z_{\text{MSB}} \rightarrow \text{no overflow}$$

$$+ X_{\text{MSB}} \cdot Y_{\text{MSB}} \cdot \overline{Z}_{\text{MSB}}$$

↑ 1 overflow

Note

need min 5 bit to represent $(-10)_10$ in 1's complement

$$x = -4 \quad \begin{array}{r} 1 \\ 0 \\ 1 \\ 1 \end{array}$$

$$y = +3 \quad \begin{array}{r} 0 \\ 0 \\ 1 \\ 1 \end{array}$$

$$\text{MSB } 1 \rightarrow \text{-ve}$$

$$\text{Ans: } -(-110)_2$$

$$= -(0001)_2$$

$$= -1$$

$$x = +4 \quad \begin{array}{r} 0 \\ 1 \\ 0 \\ 0 \end{array}$$

$$y = -3 \quad \begin{array}{r} 1 \\ 1 \\ 0 \\ 0 \end{array}$$

if carry generated

$$\begin{array}{r} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{array}$$

$$\text{Ans: } +1$$

$$(r-1)'s \quad \begin{array}{r} 9 \\ (62)_10 \end{array}$$

$$\text{Ans: } +9$$

$$(r)'s \quad -x = \bar{x} + 1 = y$$

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

$$x = y - 1$$

eg: 10

$$+\bar{x} = 10 - 0 - 1$$

$$= 01$$

$$x = -(01)_2 = -(10)_1$$

$$= -2$$

-3 = 101
to represent 10 in 4 bits

$$-3 = 1101$$

A = +4 = 0100

$$B = -3 = 1101$$

$$A - B = 0001$$

carry neglected

$$A - B = 0001$$

final carry indicator

$$1 \rightarrow +ve \text{ fans}$$

$$0 \rightarrow -ve \text{ fans}$$

Decimal	XS-3
0	000
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

$$(2)_{XS3} = (01000101)_2$$

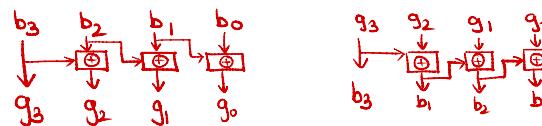
$$28 \quad \begin{array}{r} 2 \\ 9 \\ 0010 \\ 0111 \\ \hline 1001 \end{array}$$

0: +ve over flow

$$\rightarrow (000) \rightarrow 0111$$

Gray code

Dec	Code	Binary	Gray
0	0000	000	000
1	0001	001	001
2	0011	010	011
3	0010	011	010
4	0110	100	110
5	0111	101	111
6	0101	110	101
7	0100	111	100
8	1100		
9	1101		
10	1111		



even parity generator = $x_1 \oplus x_2 \oplus x_3 \oplus x_4$, checker = $x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus P_c \rightarrow 0$ no error
 odd = $\overline{x_1 \oplus x_2 \oplus x_3 \oplus x_4}$, checker = $\overline{x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus P_b} \rightarrow 0$ no error

S code: even parity

↪ $\leq C_2$ ways.

00011
00101
00110
01001
01010
01100
10001
10010
10100
11000

MULTIPLICATION

hexadecimal

$$\begin{array}{r}
 36 \rightarrow 16(2)+4 \\
 33+2 \rightarrow 35 \\
 30+2 \rightarrow 32 \\
 \hline
 \begin{array}{r}
 \overset{2}{A} \overset{2}{B} C \\
 \times \quad 2 \ 3 \\
 \hline
 2 \ 0 \ 3 \ 4 \\
 1 \ 5 \ 7 \ 8 \ 0 \\
 \hline
 1 \ 7 \ 7 \ B \ 4
 \end{array}
 \quad
 \begin{array}{r}
 9 \ 4 \\
 \times \quad 1 \ 2 \\
 \hline
 1 \ 2 \ 8 \\
 + 9 \ 4 \ 0 \\
 \hline
 A \ 6 \ 8
 \end{array}
 \end{array}$$

$$18 \rightarrow 16+2$$

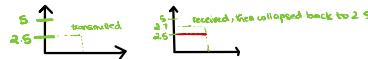
ANALOG VS DIGITAL

continuous
takes ANY value within given limit

↳ minimizes the effect of noise (unwanted signal)

→ DTS (discrete time signal)

Independent axis (x) is recorded @ values marked by lines, rest of it assumed.
 $\Delta t = t_1 - t_0 = t_2 - t_1 = \dots$



analog to digital converter

Analog $\xrightarrow[\text{in}]{\text{ADC}}$ DIGITAL SYSTEM $\xrightarrow{\text{DAC}}$ Analog out

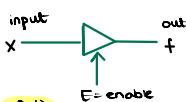
LOGIC GATES AND BOOLEAN ALGEBRA



BUFFER

$2T_{PD}$ delay

TRI STATE BUFFER



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

(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)

$$(A')' = A$$

$$\begin{aligned} A \cdot A &= A \\ A \cdot 0 &= 0 \\ A \cdot 1 &= A \\ A \cdot A' &= 0 \end{aligned}$$

$$\begin{aligned} A + A &= A \\ A + 0 &= A \\ A + 1 &= A \\ A + \bar{A} &= 1 \end{aligned}$$

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

Duality

$x \cdot 0 = 0$	$x+1 = 1$	one times dual gives same expression
$x \cdot \bar{x} = 0$	$x+\bar{x} = 1$	eg. $G_1 = AB + BC + AC$
$x+0 = x$	$x \cdot 1 = x$	$G_1 = (A+B)(B+C)(A+C)$
		$G_2 = (B+A)(C+A)$
		$G_3 = AB + AC + BC$

$$A + B = B + A \text{ (dual)}$$

$$A \cdot B = B \cdot A$$

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

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

$$(x+y)(x \cdot y) = 0 \Rightarrow x \cdot y + x \cdot y = 0$$

$$(x+y)+(z \cdot y) = 1 \Rightarrow (x+z+y) \cdot (x+y \cdot z) = 1 \cdot 1 = 1.$$

$$\Rightarrow (1+y) \cdot (1+x) = 1 \cdot 1 = 1.$$

POS

$$A+B \quad AB' \quad A'B' \quad A'B$$

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

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

\bar{A}				A								
BC	DE	00	01	11	10	BC	DE	00	01	11	10	
00	m ₀	m ₁	m ₃	m ₂	m ₁₆	m ₁₇	m ₁₉	m ₁₈	m ₂₀	m ₂₁	m ₂₃	m ₂₂
01	m ₄	m ₅	m ₇	m ₆	m ₁₂	m ₁₃	m ₁₅	m ₁₄	m ₂₈	m ₂₉	m ₃₁	m ₃₀
11	m ₁₂	m ₁₃	m ₁₅	m ₁₄	m ₈	m ₉	m ₁₁	m ₁₀	m ₂₄	m ₂₅	m ₂₇	m ₂₆
10	m ₁₆	m ₁₇	m ₁₉	m ₁₈	m ₂₀	m ₂₁	m ₂₃	m ₂₂	m ₂₈	m ₂₉	m ₃₁	m ₃₀

$$\begin{aligned} Y &= AB + A'C + BC(A+B') \\ Y &= AB + A'C \end{aligned}$$

$$\begin{aligned} Y &= AB + A'C + BC(A+B') \\ &= AB + ABC + A'BC + A'B'C \\ &= AB(1+C) + A'BC(1+B) \\ &= AB + A'C. \end{aligned}$$

MINIMAL \rightarrow CANONICAL

① 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

$$\text{eg. } Y = A + B'C$$

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

$$\begin{aligned} f(x, y, z) &= E(2, 6, 7) + \pi(1, 3, 6, 7) \\ &= E(2, 6, 7) + E(0, 2, 4, 5) \\ &= E(0, 2, 4, 5, 6, 7) \\ &= \pi(0, 1, 3) \end{aligned}$$

$$f_1 \Rightarrow f \Rightarrow g$$

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

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

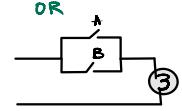
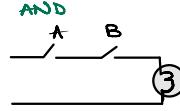
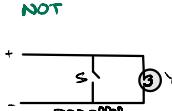
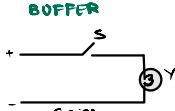
$$\therefore f = E(1, 4)$$

$$g = \pi(1, 4)$$

$$f_1 \Rightarrow f \Rightarrow g$$

$$f = E(0, 1, 2, 3, 4, 5, 7)$$

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



$$A \oplus A = 0$$

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

$$A \oplus 0 = A$$

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

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

$$A \oplus A \oplus \dots \oplus A = A \quad \begin{matrix} n: \text{odd} \\ -0 \\ n: \text{even} \end{matrix}$$

$$A \oplus A = 1$$

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

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

$$A \oplus 1 = A$$

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

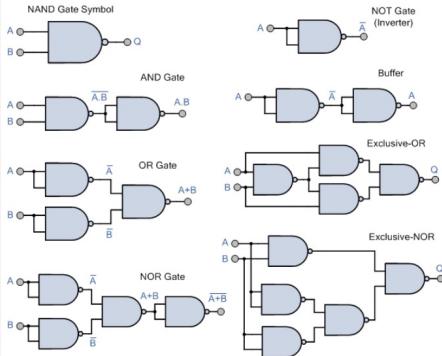
$$A \oplus A \oplus \dots \oplus A = A \quad \begin{matrix} n: \text{odd} \\ -1 \\ n: \text{even} \end{matrix}$$

ODD NO of input

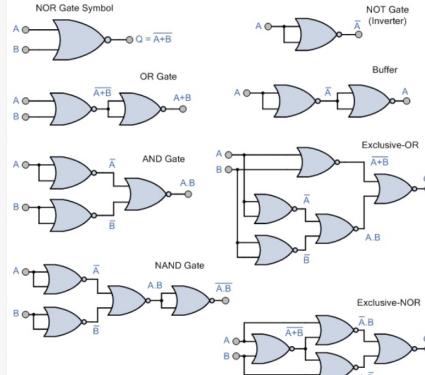
$$\text{XOR} = \text{XNOR}$$

EVEN NO complement

Logic Gates using only NAND Gates



Logic Gates using only NOR Gates



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

What does digital mean?

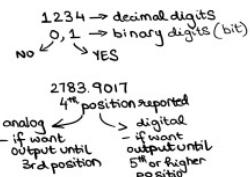
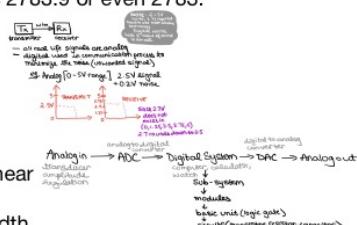
- Very simply, it means that the device or the system works with digits, numbers, discrete quantities.
- These numbers are usually, at least in the interior of the device, binary numbers—0s and 1s.
- We usually see the results as “people” numbers—decimal, most likely—and alphanumeric characters, including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

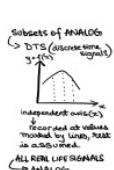
- Quality of service
- Maintenance
- Flexibility
- Upgradability
- Exponential vs Linear
- Delay
- Uses less bandwidth
- Higher efficiency for long distance transmission.
- Encryption: at transmission end we encrypt the intended data user decodes using proper key.



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).



Signal	Analog	Digital	← both parameters are discrete. lower marked limit is constant. e.g. 1,2,3 → assumed as 1
Representation	Analog signal is a continuous signal which represents physical measurements.	Digital signals are discrete time signals generated by digital modulation.	
Technology	Denoted by sine waves	Denoted by square waves	
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 parameters 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.	
Application	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.	

Top-down approach

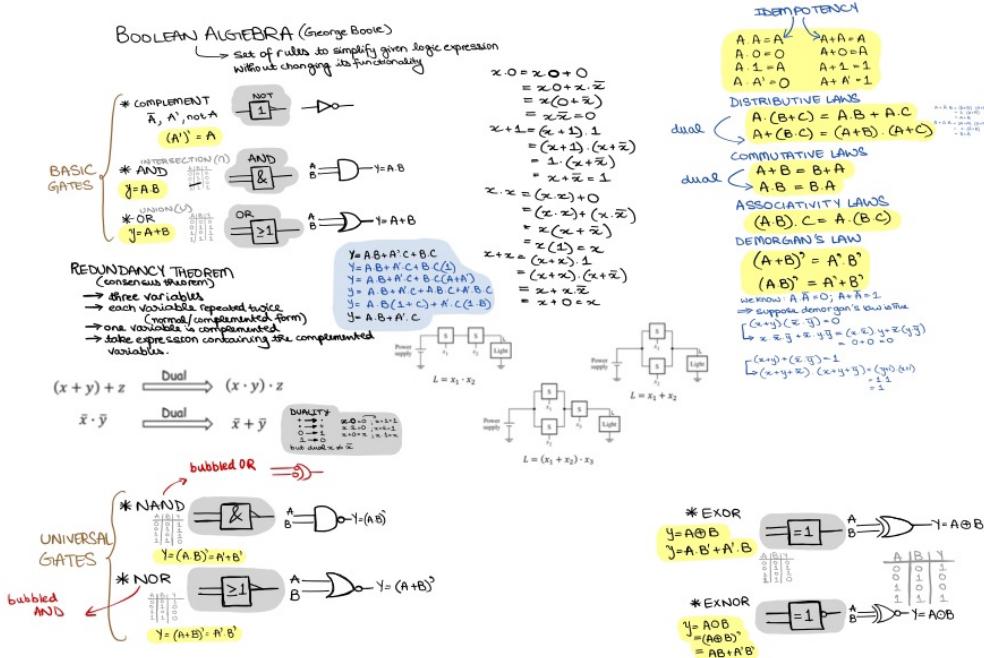
- The top-down approach involves looking at the big picture and breaking that picture up into smaller constituents.
- Each of these smaller parts is a complete entity, which will combine with all these other parts to make the finished whole.
- But the smaller constituents may still be too large for one 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.

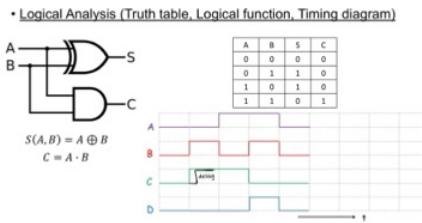
WHAT'S A DIGITAL SYSTEM?

- 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.

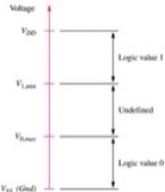


LOGIC CIRCUIT FAMILIES

- 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.

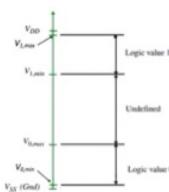
Logic Circuits and Binary Levels

Recognize at the input:



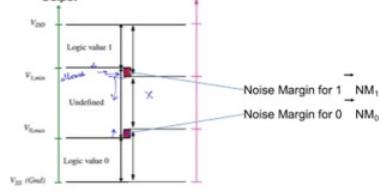
Logic Circuits and Binary Levels

Available at the output:

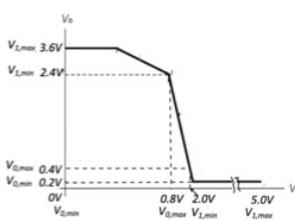


Logic Circuits and Binary Levels

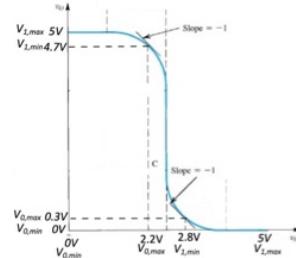
Output Input



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 $A + B$, $\bar{C} + D + E$, 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, \bar{C}, D, E , and F . If the **variable appears both primed and unprimed**, as in G and \bar{G} , each is a literal.
- Sum-of-products** is an expression consisting of product terms linked by or operators $A \cdot B + \bar{C} \cdot D \cdot E + F$.
- Product-of-sums** is an expression consisting of sum terms linked by and operators $(A + B) \cdot (\bar{C} + D + E) \cdot F$.

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
3 m_3	0	1	1	0
4 m_4	1	0	0	1
5 m_5	1	0	1	1
6 m_6	1	1	0	1
7 m_7	1	1	1	1

for when output is high
SUM OF PRODUCTS (SOP)

$A: 1$
 $A': 0$

$$\begin{aligned} & \rightarrow A' \cdot B \cdot C' \\ & \rightarrow A \cdot B' \cdot C' \\ & \rightarrow A \cdot B' \cdot C \\ & \rightarrow A \cdot B \cdot C' \\ & \rightarrow A \cdot B \cdot C \end{aligned}$$

$$F = A' \cdot B \cdot C' + A \cdot B' \cdot C' + A \cdot B \cdot C' + A \cdot B \cdot C \quad \{\text{CANONICAL/STANDARD SOP form}\}$$

$$\begin{aligned} F(A, B, C) &= m_2 + m_4 + m_5 + m_6 + m_7 \\ &= \Sigma m(2, 4, 5, 6, 7) \end{aligned}$$

$$F = A' \cdot B \cdot C' + A \cdot B' \cdot C' + A \cdot B' \cdot C + A \cdot B \cdot C \quad \text{CANONICAL}$$

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

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

$$= A' \cdot B \cdot C' + A$$

$$F = A + B \cdot C' \quad \{A + \bar{A} \cdot B = A + B\}$$

MINIMAL

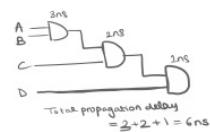
binary	A	B	Y
m_0	0	0	0
m_1	0	1	1
m_2	1	0	0
m_3	1	1	1

$$A' \cdot B$$

$$A \cdot B$$

$$\begin{aligned} Y &= A' \cdot B + A \cdot B \\ &= B \cdot (A' + A) \\ &= B \end{aligned}$$

$$Y = \Sigma m(1, 3)$$



$$f = 1/10 \text{ ns}^{-1}$$

PRODUCT OF SUM (POS)

for when output is low

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

$$\begin{array}{l} A = 0 \\ A' = 1 \end{array}$$

$$A + B + C$$

$$A + B + C'$$

1

$$A + B' + C'$$

1

1

1

1

derive POS from SOP
↳ SOP for low

$$\begin{aligned} Y &= (A \cdot B \cdot C)' \cdot (A \cdot B' \cdot C)' \cdot (A' \cdot B \cdot C)' \\ &= (A+B+C) \cdot (A+B+C') \cdot (A+B'+C') \end{aligned}$$

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

$$Y = \pi M(0, 1, 3) \quad \{ \text{CANONICAL/STANDARD POS form} \}$$

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

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

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

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

$$= A+B \cdot B'+B \cdot C'$$

$$Y = A+B \cdot C'$$

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

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

$$\{ \text{minimal POS form} \}$$

e.g. min terms

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

max terms

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

$2^{(2^n)}$ 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

positive and negative logic OR gate

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

Observation

- pos logic AND gate = -ve logic OR gate
- pos 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 \rightarrow CSOP

$$\begin{aligned} f(x, y, z) &= \bar{x} \cdot z + x \cdot \bar{y} \\ &= \bar{x} \cdot z(y + \bar{y}) + x \cdot \bar{y} \cdot (z + \bar{z}) \\ &= \bar{x} \cdot z \cdot y + \bar{x} \cdot \bar{y} \cdot \bar{z} + x \cdot \bar{y} \cdot z + x \cdot \bar{y} \cdot \bar{z} \end{aligned}$$

$$\begin{aligned} f(a, b, c) &= a + \bar{b} \cdot c \\ &= a \cdot (b + \bar{b}) \cdot (c + \bar{c}) + (a + \bar{a}) \bar{b} \cdot c \\ &= a \cdot b \cdot c + a \cdot b \cdot \bar{c} + a \cdot \bar{b} \cdot c + a \cdot \bar{b} \cdot \bar{c} + \bar{a} \cdot \bar{b} \cdot c \\ &= a \cdot b \cdot c + a \cdot b \cdot \bar{c} + a \cdot \bar{b} \cdot c + a \cdot \bar{b} \cdot \bar{c} + \bar{a} \cdot \bar{b} \cdot c \end{aligned}$$

POS \rightarrow CPPOS

$$\begin{aligned} f(x, y, z) &= (x + z) \cdot (\bar{x} + \bar{y}) \\ &= (x + z + y \cdot \bar{y}) \cdot (\bar{x} + \bar{y} + z \cdot \bar{z}) \\ &= (x + z + y) \cdot (x + z + \bar{y}) \cdot (\bar{x} + \bar{y} + z) \cdot (\bar{x} + \bar{y} + \bar{z}) \end{aligned}$$

$$\begin{aligned} f(a, b, c, d) &= a + d \\ &= a + d + b \cdot \bar{b} + c \cdot \bar{c} \\ &= (a + d + b \cdot \bar{b} + c) \cdot (a + d + b \cdot \bar{b} + \bar{c}) \\ &= (a + b + c + d) \cdot (a + \bar{b} + c + d) \cdot (a + b + \bar{c} + d) \cdot (a + \bar{b} + \bar{c} + d) \end{aligned}$$

COMPLEMENT

$$\bar{z}(C, B, A) = \Sigma_m(1, 3, 5, 6, 7) = \Pi_m(0, 2, 4)$$

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

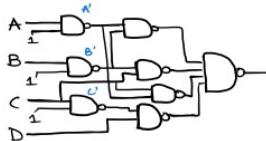
$$f(D, C, B, A) = \sum m(0, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14)$$

	00	01	10	11
00	1			
01	1	1		1
10	1	1	1	
11	1	1	1	1

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

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

$$(f')' = [(A' \cdot B)' \cdot (C' \cdot D)' \cdot (B' \cdot C)' \cdot (A' \cdot C)']'$$



Logic Synthesis

- Implement function using NOR gates

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

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

Logic Synthesis:

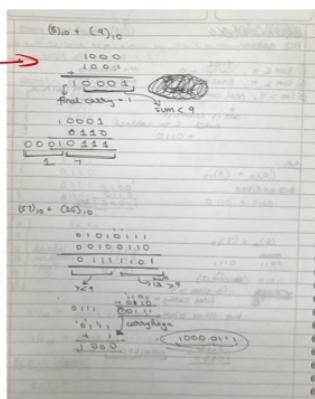
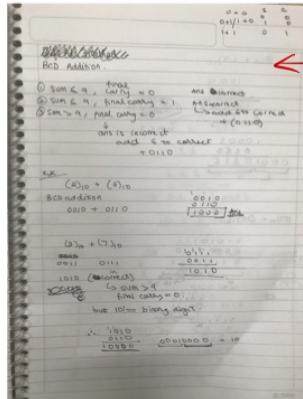
- Implement function using NOT, OR and AND gates in POS

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

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

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 or OR gates) to implement using NOT, OR and AND gates



XS-3

$$(27)_{10} + (39)_{10}$$

$$27 \{ 00100111$$

$$39 \{ 00111001$$

$$+$$

$$00100111$$

$$+00111001$$

$$01010100$$

$$11000110$$

$$G_1$$

$$010111010$$

$$011011100$$

$$11000110$$

$$G_2$$

$$010111010$$

$$011011100$$

$$11000110$$

$$G_1$$

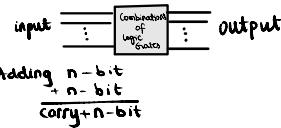
$$010111010$$

$$011011100$$

$$11000110$$

$$G_2$$

Combinational logic circuits

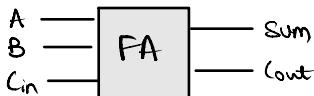
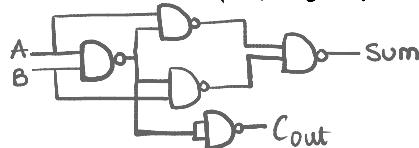


A B Sum Cout

0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$\text{Sum} = A \oplus B$$

$$\text{Cout} = A \cdot B$$



A B Cin Sum Cout

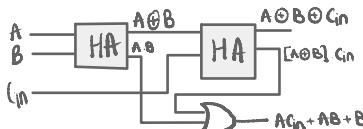
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$\text{Sum} = A \oplus B \oplus \text{Cin}$$

$$= A_0 B_0 \text{Cin}$$

$$\text{Cout} = AB + BC_{\text{in}} + AC_{\text{in}}$$

checkboard on KMAP.

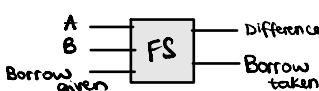
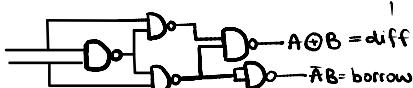
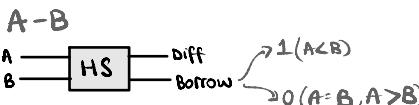


$$\begin{aligned} &ABC_{\text{in}} + ABC_{\text{in}} + AB(C_{\text{in}}) \\ &AB(C_{\text{in}}) + BC(C_{\text{in}}) + BCA(C_{\text{in}}) \\ &AC_{\text{in}} + ABC_{\text{in}} + BCA + BC_{\text{in}} \\ &AC_{\text{in}} + AB + ABC_{\text{in}} + A(BC) \\ &AC_{\text{in}} + AB + BCA \end{aligned}$$

$$\begin{aligned} \text{Sum delay} &= 2 \cdot \text{XOR} \\ \text{Carry delay} &= 1 \cdot \text{XOR} + 1 \cdot \text{AND} + 1 \cdot \text{OR} \end{aligned}$$

if $A=B=1$, $\text{Cout}=1$
 * don't have to wait for second FA's output
 CRUCIAL for delay problems

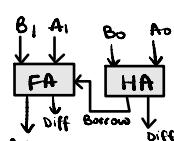
MAJORITY INPUT GATES
 eg - AND/OR
 \downarrow
 $0:0$



A B Diff Borrow

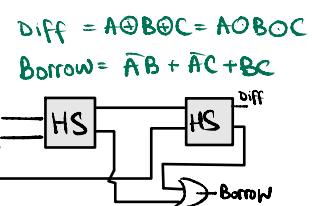
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

$$\begin{aligned} \text{Diff} &= A \oplus B \\ \text{Borrow} &= \bar{A}B + \bar{A}\bar{C} + BC \end{aligned}$$

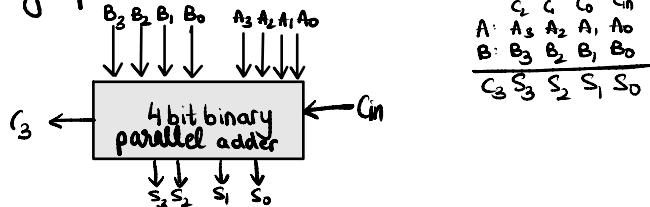


A B Cin Diff Borrow

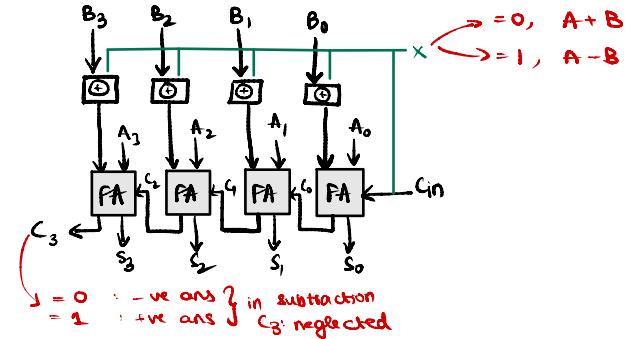
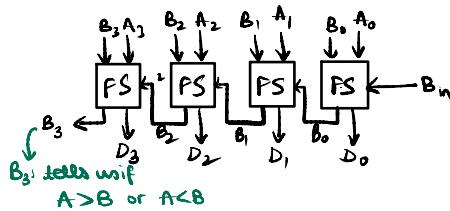
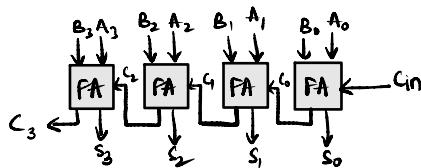
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



Binary parallel adder

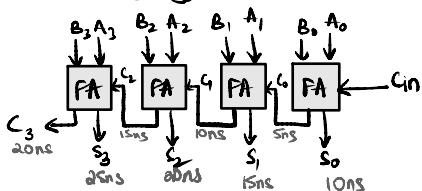


$$\begin{array}{r}
 & C_2 & C_1 & C_0 & C_{in} \\
 A: & A_3 & A_2 & A_1 & A_0 \\
 B: & B_3 & B_2 & B_1 & B_0 \\
 \hline
 C_3 & S_3 & S_2 & S_1 & S_0
 \end{array}$$



FA are identical
 $\rightarrow PD(\text{sum}) = 10ns$
 $\rightarrow PD(\text{carry}) = 5ns$

$$\text{Worst case delay} = (n-1) \text{Carry delay} + \max_{(n-\text{bit})} (\underbrace{\text{CD}/SD}_{\text{cum delay}})$$



DECIMAL ADDER (1 bit FA)

BCD ADDER

1 dec digit = 0 to 9

2nd dec digit = 0 to 9

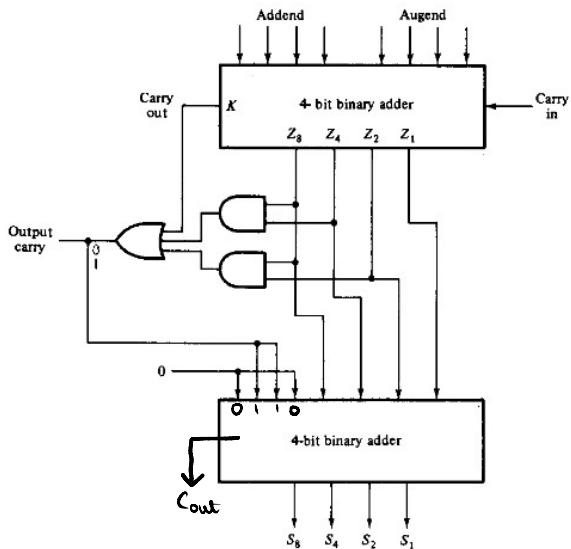
$$C_m = 0/1$$

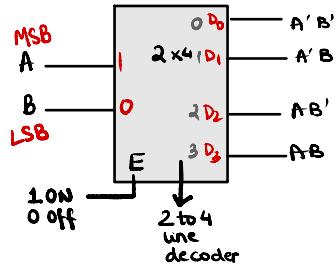
$$\text{Max sum} = 19$$

TABLE 5-1
Derivation of a BCD Adder

K	Binary Sum				C	BCD Sum				Decimal
	Z_8	Z_4	Z_2	Z_1		S_6	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
1	0	0	0	0	1	0	0	0	0	10
1	0	0	0	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





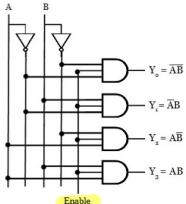
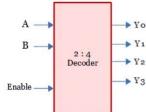
+ OR gate → SOP

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



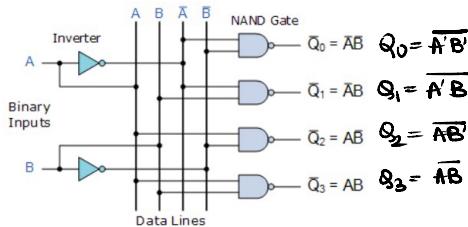
ACTIVE HIGH DECODER

www.electrical4you.com



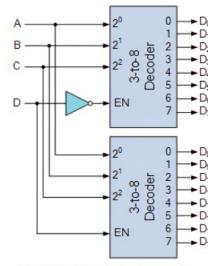
EN	Inputs			Outputs			
	A	B	X	Y ₃	Y ₂	Y ₁	Y ₀
0	X	X	X	0	0	0	0
1	0	0	0	0	0	0	1
1	0	1	0	0	0	1	0
1	1	0	0	0	1	0	0
1	1	1	1	1	0	0	0

ACTIVE LOW DECODER.



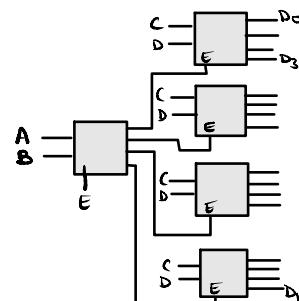
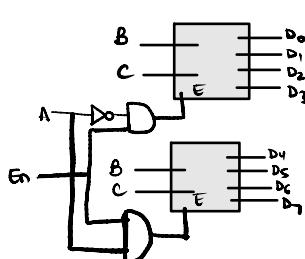
2-to-4 NAND Gate Binary Decoder			
E	A	B	Q ₀ Q ₁ Q ₂ Q ₃
0	0	0	0 1 1 1
0	0	1	1 0 1 1
1	0	0	1 1 0 1
1	1	0	1 1 1 0
0	x	x	1 1 1 1

A 4-to-16 Binary Decoder Configuration



3x8 Decoder using 2x4 decoder.

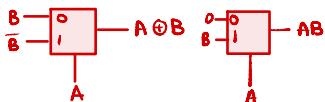
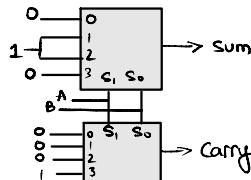
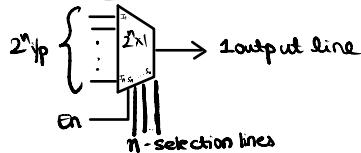
A	B	C	D ₀
0	0	0	D ₀
0	0	1	D ₁
0	1	0	D ₂
0	1	1	D ₃
1	0	0	D ₄
1	0	1	D ₅
1	1	0	D ₆
1	1	1	D ₇



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

MUX

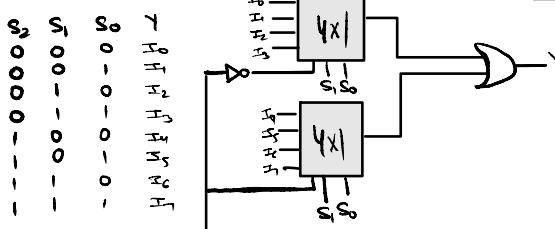
$2^n \times 1$ MUX



8×1 MUX implemented using 4×1 .

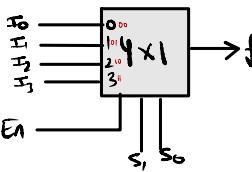
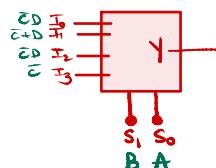
$$\frac{8}{4} = 2$$

$$\frac{2}{4} = 0.5$$



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

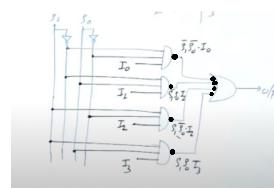
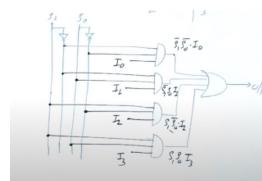
$AB\backslash CD$	$\bar{C}\bar{D}$	$\bar{C}D$	$C\bar{D}$	CD
$\bar{A}\bar{B}$	1			
$\bar{A}B$	0	1	1	
AB	1	1		
$A\bar{B}$	0	1		



E	S ₁	S ₀	O/P
1	0	0	I_0
1	0	1	I_1
1	1	0	I_2
0	1	1	I_3

highz (no ans)

$$f = (\bar{S}_1 \bar{S}_0 I_0 + \bar{S}_1 S_0 I_1 + S_1 \bar{S}_0 I_2 + S_1 S_0 I_3) \cdot E_n$$



Encoder ($n \rightarrow m$ line) $m = \log_2 n$

En	w_7	w_6	w_5	w_4	w_3	w_2	w_1	w_0	y_2	y_1	y_0
1									0	0	0
	1									0	1
		1							0	1	0
			1						0	1	1
				1					1	0	0
					1					0	1
						1				1	0
							1			1	1
0	x	x	x	x	x	x	x	x	high z		

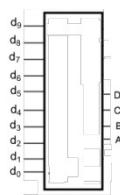
Active high

Active low: singular zeros instead of 1s

10:4 Decimal-Binary Encoder:

$$m = \lceil \log_2 10 \rceil = \lceil 3.322 \rceil = 4$$

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	0
0	0	0	1	0	0	0	0	0	0	0	1	0	1
0	0	1	0	0	0	0	0	0	0	0	1	1	0
0	1	0	0	0	0	0	0	0	0	0	1	1	1
1	0	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

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

$$y_1 = w_3 + w_2$$

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

