A logic family is a collection of different IC chips that have similar input, output and internal circuit characteristics, but that perform different logic functions.

Chips from the same family can be interconnected to perform any desired logic function. On the other hand, chips from differing families may not be compatible; they may use different power-supply voltages or may use different input & output conditions to represent logic values.

The most successful bipolar logic family is transistor-transistor logic (TTL). First introduced in the 1960s, TTL now is actually a family of logic families that are compatible with each other but differ in speed, power consumption, and cost.

Metal-Oxide Semiconductor field-effect transistor (MOSFET), or simply MOS transistor, were difficult to fabricate in the early days, and it wasn't until the 1960s that a wave of developments made MOS based logic & memory circuits practical. Even then, MOS circuits lagged bipolar circuits considerably in speed and were attractive only in selected applications because of their lower power consumption and higher levels of integration.

Beginning in the mid-1970s, advances in the design of MOS circuits, in particular Complementary MOS (CMOS) circuits, vastly increased their performance & popularity. By far the majority of new large-scale integrated circuits, such as microprocessors & memories, use CMOS. CMOS circuits now account for the vast majority of the world wide IC market.

CMOS logic is both the most capable & the easiest to understand commercial digital logic technology.

As a consequence of the industry's transition from TTL to CMOS over long period of time, many CMOS families were designed to be somewhat compatible with TTL.

CMOS LOGIC

The basic building blocks in CMOS logic circuits are MOS transistors. In any logic circuit, there is a range of voltages that is interpreted as a logic 0, and another, nonoverlapping range that is interpreted as a logic 1.

A typical CMOS logic circuit operates from a 5-volt power supply. Such a circuit may interpret any voltage in the range 0-1.5 V as a logic 0, and in the range 3.5 - 5.0V as a logic 1. Thus, the definitions of LOW & HIGH for 5-volt CMOS logic are as shown in figure 1.

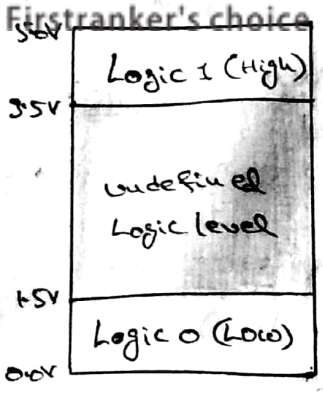


fig 1: Logic levels for typical CMOS logic circuits

Voltages in the intermediate range (1.5 - 3.5V) are not expected to occur except during signal transitions, and yield undefined logic values (i.e., a circuit may interpret them as either 0 or 1).

MOS Transistors:

A MOS transistor can be modeled as a 3-terminal device that acts like a voltage controlled resistance. As suggested by figure 2, an i/p voltage applied to one terminal controls the resistance between the remaining two terminals. In digital logic applications, a MOS transistor is operated so its resistance is always either very high (and the transistor is OFF) or very low (and the transistor is ON).

There are two types of MOS transistors, n-channel & p-channel; the names refer to the type of semiconductor material used for the resistance-controlled terminals.

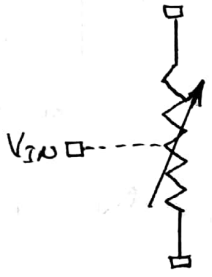
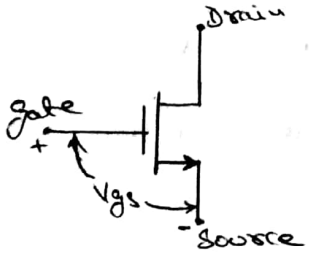


fig 2 The MOS transistor as a voltage controlled resistance.

The circuit symbol for an n-channel MOS (NMOS) transistor is shown in figure 3:



Voltag-controlled resistance
increase V_{gs} => decrease R_{ds}
i.e $V_{gs} \propto \frac{1}{R_{ds}}$

Note: normally $V_{gs} \geq 0$

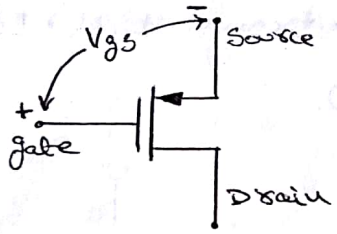
fig 3 Circuit symbol for an n-channel MOS (NMOS) transistor

The terminals are called gate, source, & drain.

As you might guess from the orientation of the circuit symbol, the drain is normally at a higher voltage than the source.

The voltage from gate to source (V_{gs}) in an NMOS transistor is normally zero or positive. If $V_{gs} = 0$, then the resistance from drain to source (R_{ds}) is very high, at least a megaohm ($10^6 \Omega$) or more. As we increase V_{gs} (i.e., increase the voltage on the gate), R_{ds} decreases to a very low value, 100 or so less in some devices.

The circuit symbol for a p-channel MOS (PMOS) transistor is shown in figure 4.9.



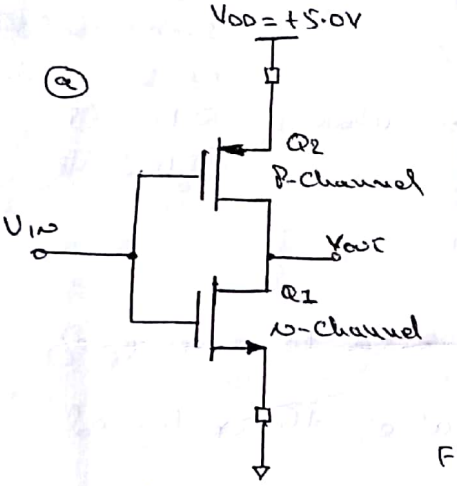
Voltage - Controlled resistance
decrease V_{gs} ⇒ decrease R_{ds}
i.e; $V_{gs} \propto R_{ds}$
Note: normally, $V_{gs} \leq 0$

Fig 4 circuit symbol for a p-Channel MOS (PMOS) transistor

Operation is analogous to that of an NMOS transistor, except that the source is normally at a higher voltage than the drain, and V_{gs} is normally zero or negative. If V_{gs} is zero, then the resistance from source to drain (R_{ds}) is very high. As we algebraically decrease V_{gs} (i.e., decrease the voltage on the gate), R_{ds} decreases to a very low value.

NMOS and PMOS transistors are used together in a complementary way to form CMOS logic.

The simplest CMOS circuit, a logic inverter, requires only one of each type of transistor, connected as shown in figure 5(a). The power-supply voltage, V_{DD}, typically may be in the range 2-6 V and is most often set at 5.0V for compatibility with TTL circuits.



(b)

V_{IN}	Q_1	Q_2	V_{OUT}
0.0(L)	OFF	ON	5.0(H)
5.0(H)	ON	OFF	0.0(L)

(c)

Fig:5 CMOS inverter (a) circuit diagram (b) functional behaviour (c) Logic Symbol

Case-I

V_{IN} is 0.0V.

In this case, the bottom, n-Channel transistor Q1 is OFF, since its V_{gs} is 0. but the top, P-channel transistor Q_2 is ON, since its V_{gs} is a large -ve (-5.0V). Therefore, Q_2 presents only a small resistance between the power-supply terminal (V_{DD}, +5.0V) and the o/p terminal (V_{OUT}), and the o/p voltage is 5.0V.

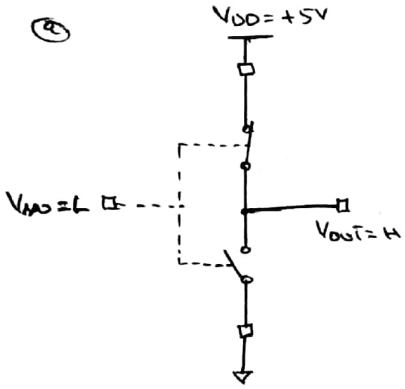
Case-II

V_{IN} is 5.0V.

Here, Q_1 is ON, since its V_{gs} is a large +ve value (+5.0V) but Q_2 is OFF, since its V_{gs} is 0. Thus, Q_1 presents a small resistance between the o/p terminal & ground, so the o/p voltage is 0V.

(6)

Another way to visualize CMOS operation uses switches. As shown in figure 6, the n-channel (bottom) transistor is modeled by a normally-open switch, and the p-channel (top) transistor by a normally-closed switch. Applying a HIGH voltage changes each switch to the opposite of its normal state, as shown in (b).



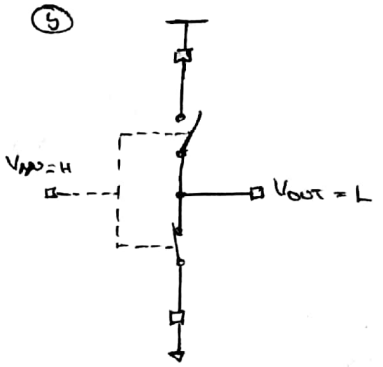


Fig 6 Switch model for CMOS inverter
(a) Low i/p
(b) High i/p

The switch model gives rise to a way of drawing CMOS circuits that makes their logical behaviour more readily apparent. As shown in fig 7, different symbols are used for the p- & n-channel transistors to reflect their logical behaviour. The n-channel transistor (Q_1) is switched 'on', & current flows between source & drain, when a HIGH voltage is applied to its gate; this seems natural enough. The p-channel transistor (Q_2) has the opposite behavior. It is "on" when a LOW voltage is applied; the inversion bubble on its gate indicates this inverting behaviour.

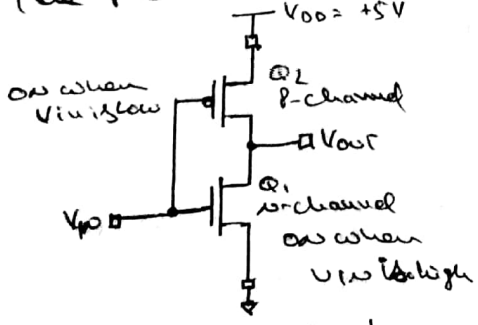
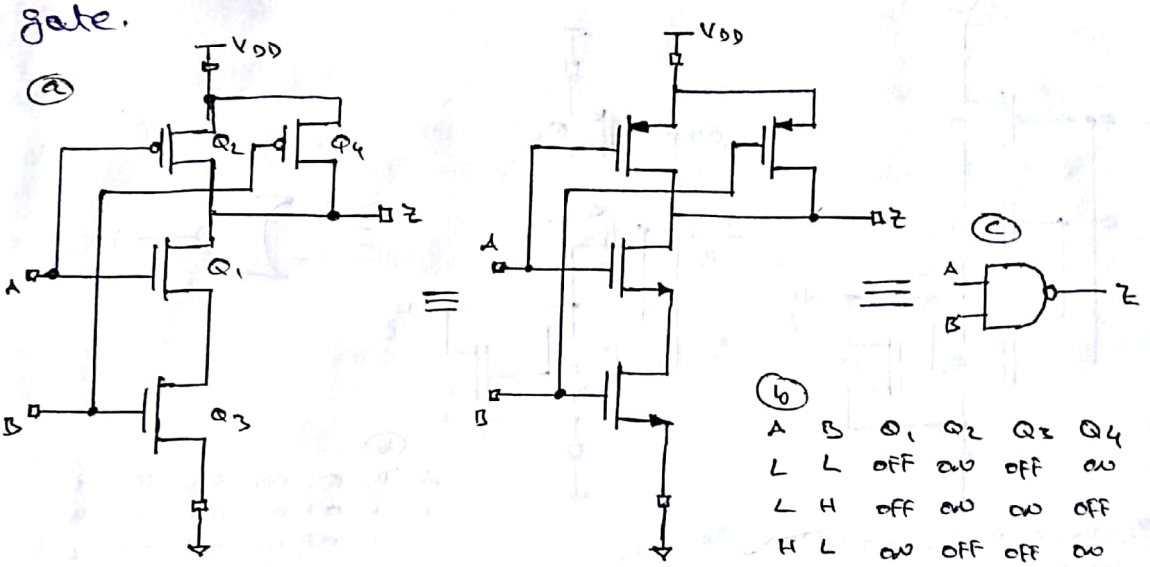


Fig 7: CMOS inverter logical operation

(6)

CMOS NAND gate

NAND gate can be constructed using CMOS. A k-i/p gate uses k p-channel & k-n-channel transistors. Figure 5.3 shows a 2-i/p CMOS NAND gate.

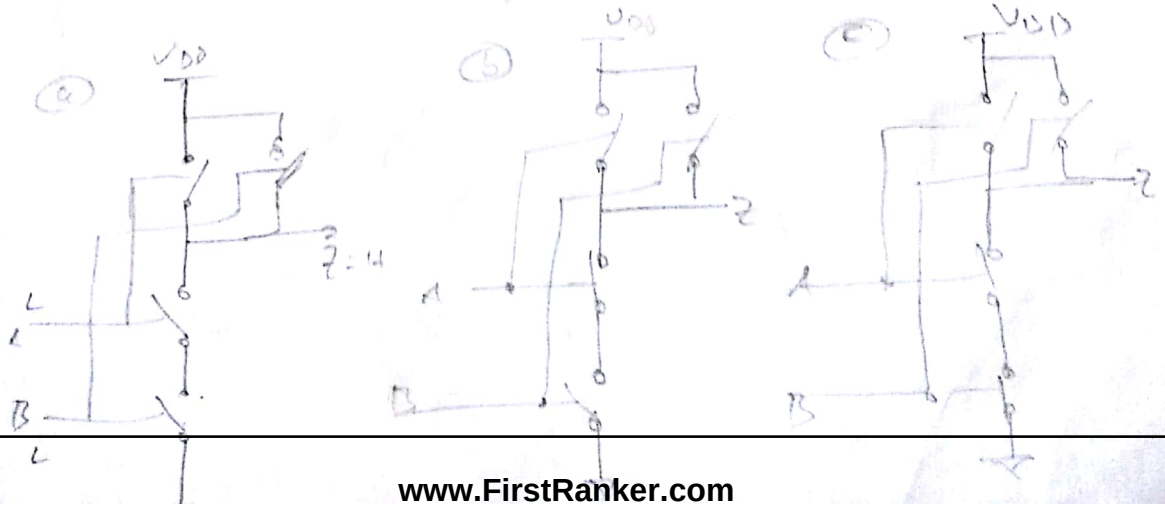


b

A	B	Q_1	Q_2	Q_3	Q_4	Z
L	L	OFF	ON	OFF	ON	H
L	H	OFF	ON	ON	OFF	H
H	L	ON	OFF	OFF	ON	H
H	L	ON	OFF	ON	OFF	L

Fig 5: CMOS 2-input NAND gate (a) circuit (b) functiontable (c) logic symbol

If either i/p is LOW, the o/p Z has a low-impedance connection to V_{DD} through the corresponding ON P-channel transistor, and the path to ground is blocked by the corresponding "OFF" n-channel transistor. If both i/ps are HIGH, the path to V_{DD} is blocked, and Z has a low-impedance connection to ground. Fig 5.4 shows the switch model for the NAND gate's operation.



CMOS NOR gate

NOR gate can be constructed using CMOS.

A K-input gate uses K n-channel & K p-channel transistors. Figure 10 shows a CMOS NOR gate.

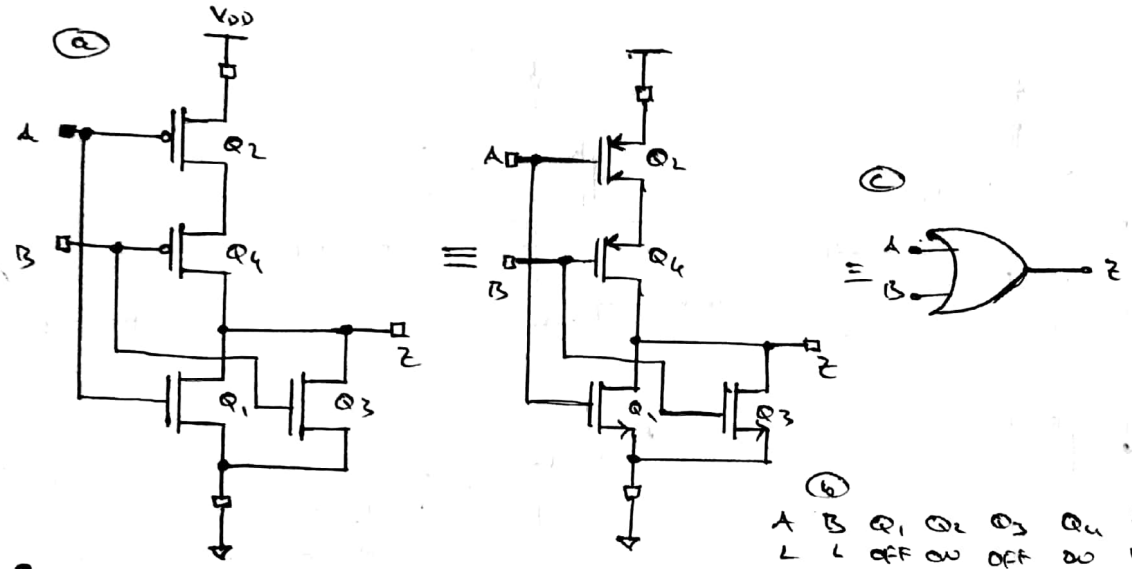


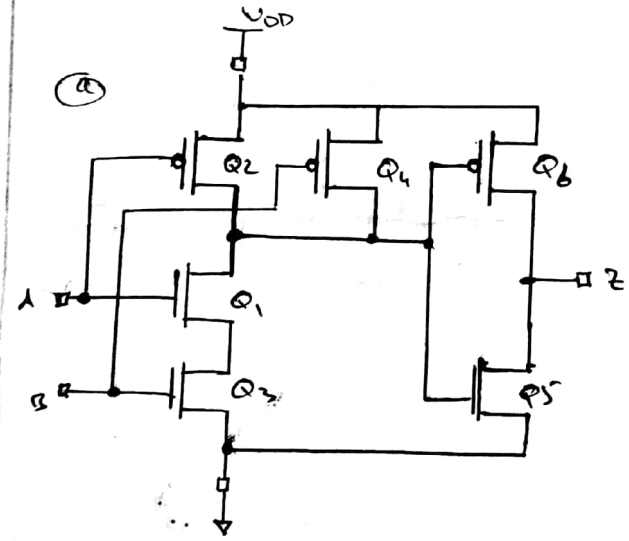
Fig ⑩: CMOS 2-input NOR gate
- ⓐ circuit diagram
- ⓑ function table
- ⓒ logic symbol

If both i/ps are Low, then the o/p Z has a Low impedance connection to V_{DD} through the "ON" p-channel transistors, and the path to ground is blocked by the "OFF" n-channel transistors. If either input is High, the path to V_{DD} is blocked, and Z has a Low-impedance connections to ground. Figure shows the switch model for the NOR gate's operation.

Non-Inverting gates

In CMOS, and in most other logic families the simplest gates are inverters, and the next simplest are NAND gates & NOR gates.

CMOS non inverting buffers and AND and OR gates are obtained by connecting an inverter to the o/p of the corresponding inverting gate. Thus, fig shows a non inverting buffer and figure shows an AND gate. Combining NOR with an inverter yields an OR gate as shown in fig



(a)

(b)

A	B	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Z
L	L	OFF	ON	OFF	ON	ON	OFF	L
L	H	OFF	ON	ON	OFF	ON	OFF	L
H	L	ON	OFF	OFF	ON	ON	OFF	L
H	H	ON	OFF	ON	OFF	OFF	ON	H

(c)

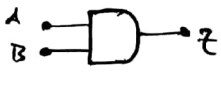


Fig : CMOS 2-input AND gate (a) circuit diagram (b) Function table (c) logic symbol

Electrical Behaviour of CMOS Circuits

CMOS Steady State Electrical Behaviour

The steady-state behaviour is the behaviour of the CKT when i/p & o/p's are not changing.

Logic levels & Noise margin

The CMOS device manufacturers specify the four voltage parameters in the data sheet. They are:

$V_{IH(min)}$ - High level input voltage.

It is a minimum voltage level required for a logical 1 at an i/p. Any voltage below this level will not be accepted as a High by the logic CKT.

$V_{IL(max)}$ - low level input voltage

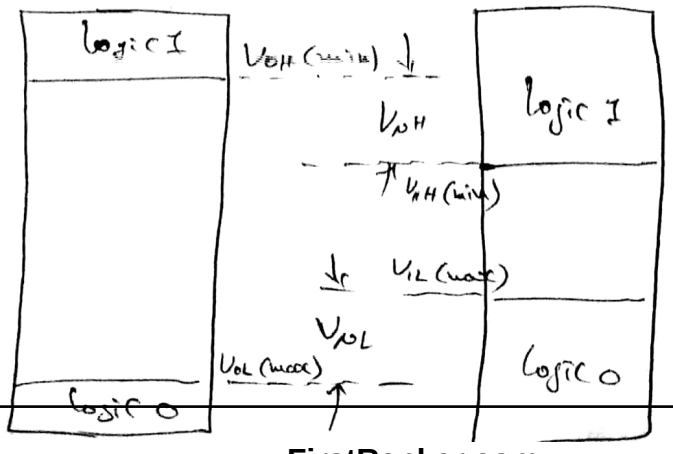
It is a maximum voltage level required for a logic 0 at an i/p. Any voltage above this level will not be accepted as a low by the logic CKT.

$V_{OH(min)}$ - High-level o/p voltage

It is a minimum voltage level at a logic CKT o/p for the logic 1 state under defined load conditions.

$V_{OL(max)}$ - low-level o/p voltage

It is a maximum voltage level at a logic CKT o/p for the logical 0 state under defined load conditions.



$V_{NL} = V_{IL(max)} - V_{OL(max)}$

Voltage levels & noise margins

The power-supply voltage V_{CC} & ground are often called the power supply rails.

CMOS levels are typically a fraction of the power supply rails.

$V_{OH(min)}$: $V_{CC} - 0.1V$

$V_{IH(min)}$: 70% of V_{CC}

$V_{IL(max)}$: 30% of V_{CC}

$V_{OL(max)}$: Ground + 0.1V

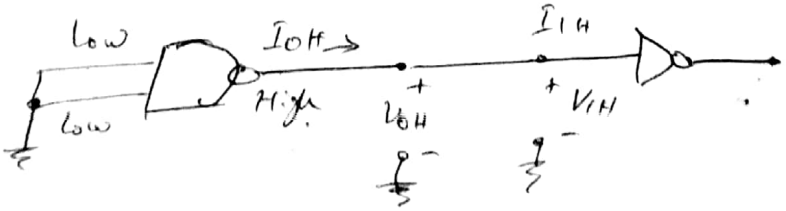
Like voltage levels, device manufacturers also specify current parameters. There are,

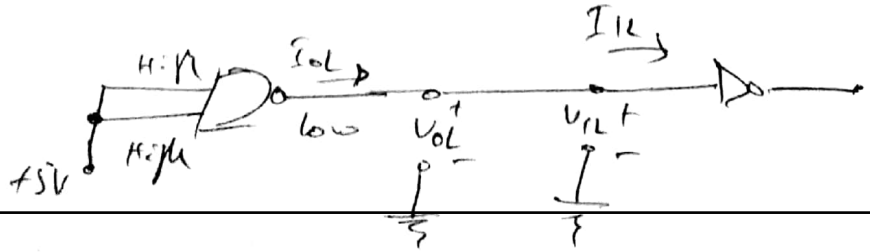
I_{IH} - High level input current : It is a current that flows into an i/p when a specified high-level voltage is applied to that i/p.

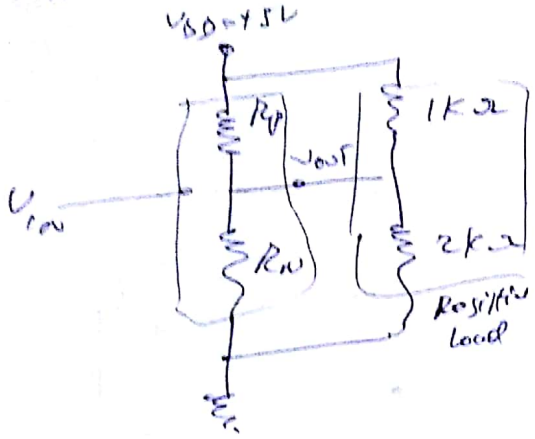
I_{IL} - low level input current : It is a current that flows into an i/p when specified low-level voltage is applied to that i/p

I_{OH} - High level o/p current : It is a current that flows from an o/p in the logical 1 state under specified load conditions.

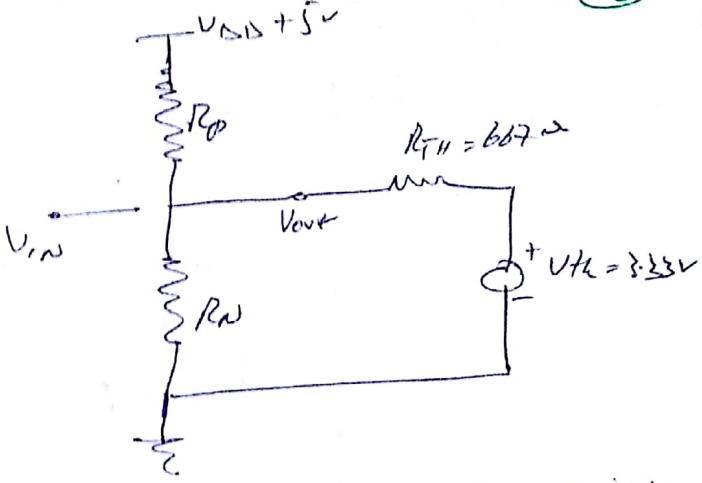
I_{OL} - Low level o/p current : It is the current that flows from an o/p in the logical 0 state under specified load conditions.







Resistive load of CMOS Inverter

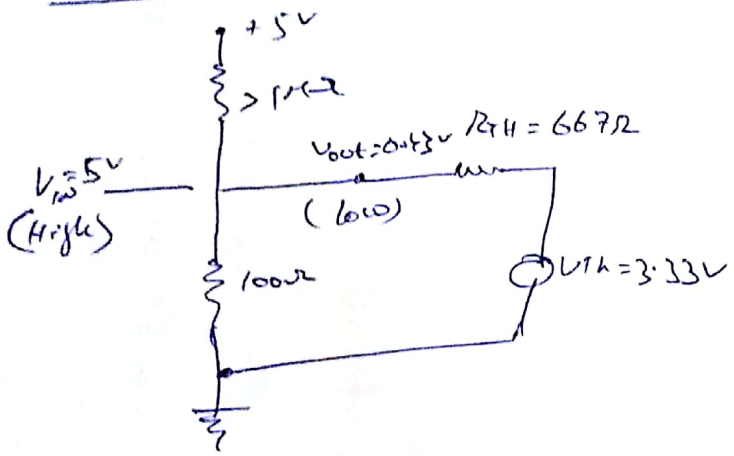


Resistive load modeled by a Thevenin equivalent CKT

$$V_{Th} = \frac{2K}{2K+1K} \times 5V = 3.33V$$

$$R_{Th} = 1K \| 2K = 667\Omega$$

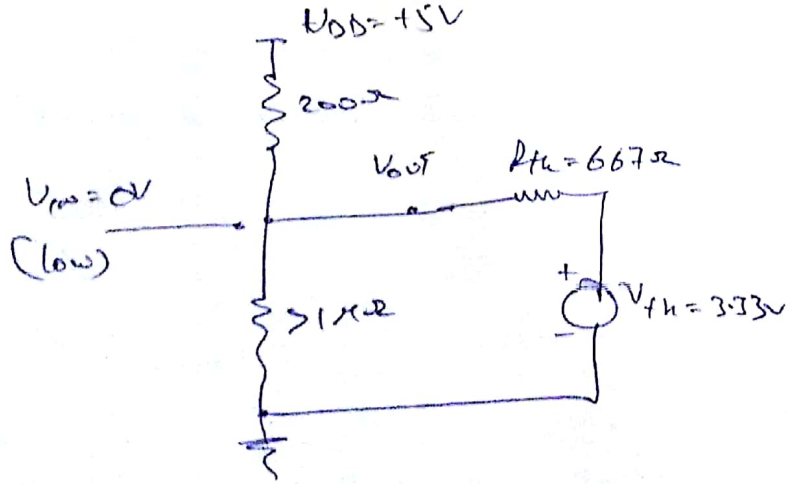
Condition 1: Input High



$$V_{out} = \frac{V_{TH} \times 100}{(100+667)} = \frac{3.33V \times 100}{100+667}$$

$$= 0.43V$$

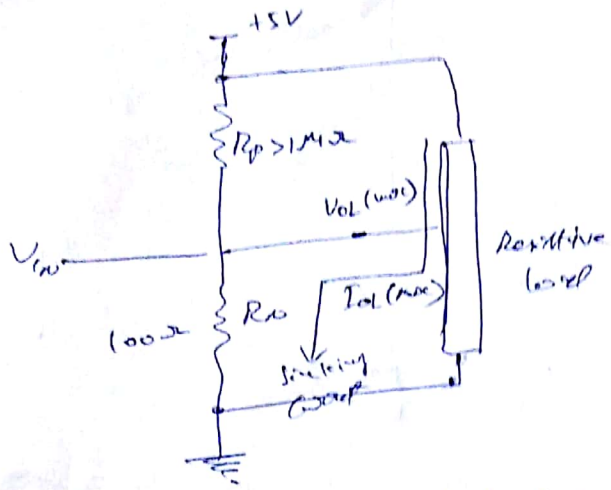
Condition 2: Input Low



$$V_{out} = \frac{(V_{DD}-V_{Th})R_{TH}}{R_{TH}+200} + V_{Th}$$

$$= \frac{(5-3.33) \times 667}{667+200} + 3.33$$

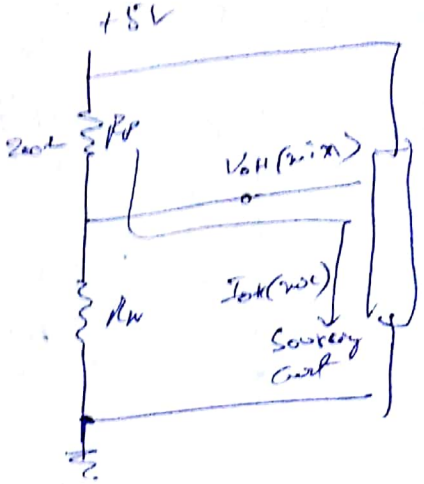
$$= 4.61V$$



$$\text{Sinking current} = \frac{V_{OL(max)}}{R_N}$$

$$= \frac{0.43V}{100}$$

$$= 4.3mA$$

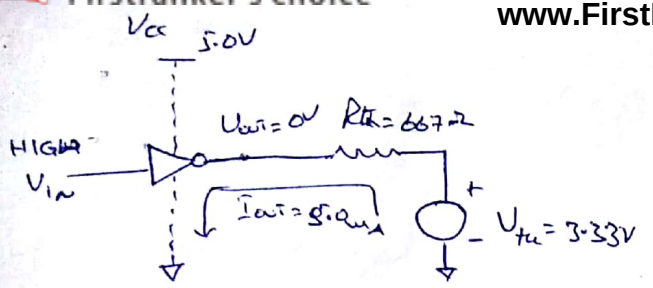


$$\text{Sourcing current} = \frac{V_{DD} - V_{OH(min)}}{R_p}$$

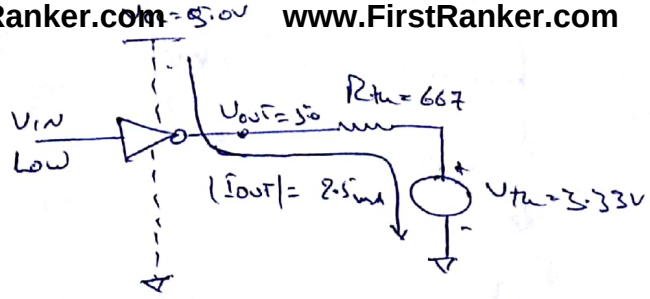
$$= \frac{5 - 4.61}{200}$$

$$= 1.95mA$$

Strategy: Thevenin equivalent circuit



(a) output low



(b) output HIGH

Assume ON resistance of transistors equal to zero i.e no voltage drop across in it, for very good worst case estimates of dp current.

For low output

$$I_{out} = \frac{V_{th}}{R_{th}} = \frac{3.33}{667} = 5\ mA$$

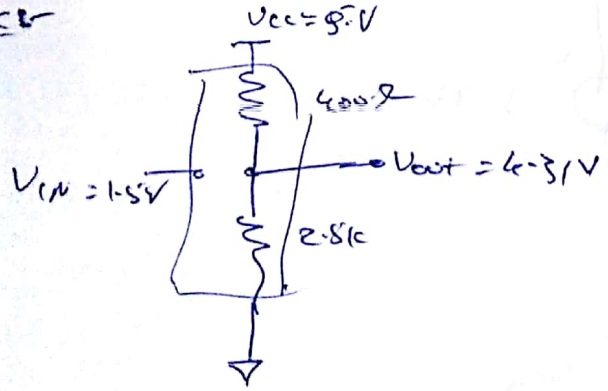
For High output

$$I_{out} = \frac{5 - 3.33}{667} = 2.5\ mA$$

Circuit Behaviour with non 'ideal' i/p

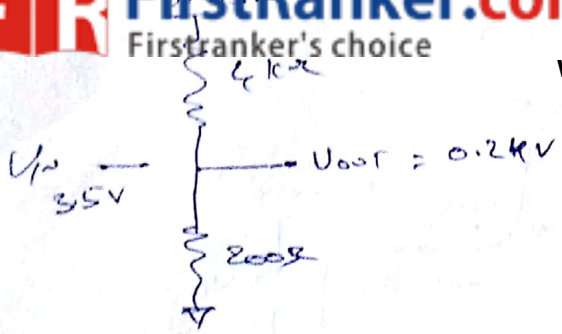
If the i/p voltage is not close to the power supply rail, then the ON transistor may not be fully 'on' & its resistance may increase. Like wise, the 'off' transistor may not be fully 'off' & its resistance may be quite a bit less than one MΩ. These two effects combine to move the output voltage away from the power supply rail.

Ex:-



$$V_{out} = \frac{(2.5k)5}{2.5k + 400} = 4.31V$$

$V_{out} = \frac{200\,(5V)}{200+4k} = 0.24V$



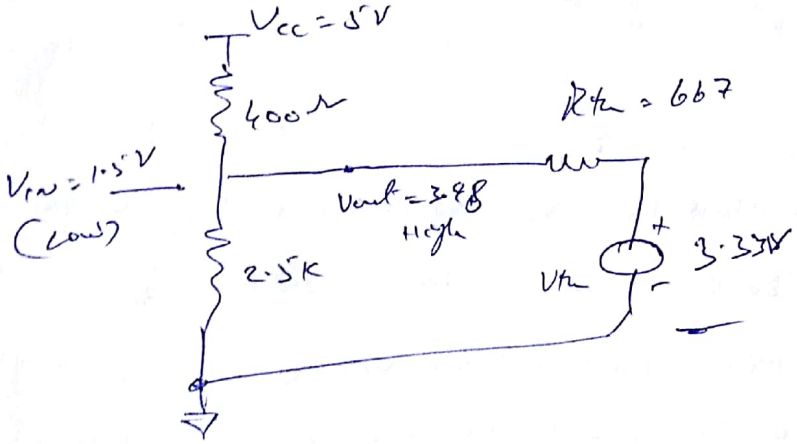
Whats worse is that the o/p structure is now consuming non trivial amount of power. The current flow with the 1.5V i/p is

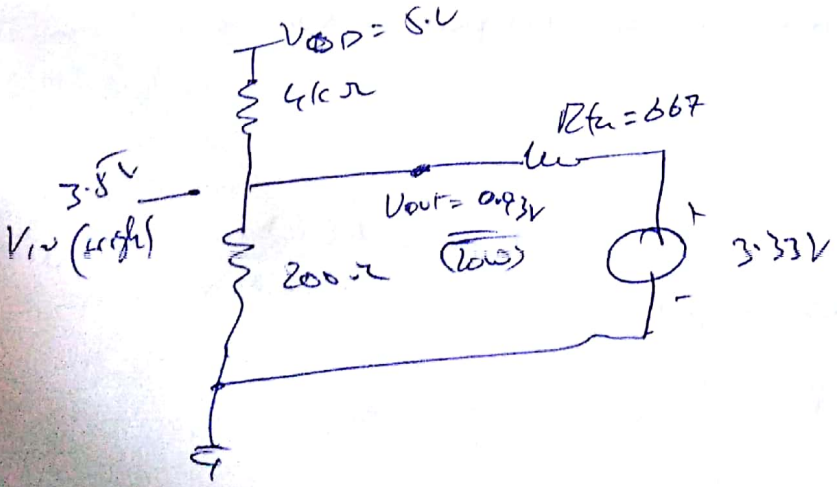
$$I_{switched} = \frac{5}{400\Omega + 2.5k} = 1.72\,mA$$

& power consumption is

$$P_{switched} = 5.0V \times I_{switched} = 5 \times 1.72\,mA = 8.62\,mW$$

The o/p voltage of CMOS inverter deteriorates further with a resistive load.





... resistances are not ... the ... by following eqns.

$$R_P(ON) = \frac{V_{DD} - V_{OH}(min)}{|I_{OH}(max)|}$$

$$R_N(ON) = \frac{V_{OL}(max)}{I_{OL}(max)}$$

Let us consider the parameters of CMOS HC-Series Listed in the table

Parameter	CMOS load	Parameter	TTL load
$I_{OL}(max)_C$	0.02 mA	$I_{OL}(max)_T$	4.0 mA
$V_{OL}(max)_C$	0.1 V	$V_{OL}(max)_T$	0.33 V
$I_{OH}(max)_C$	−0.02 mA	$I_{OH}(max)_T$	−4.0 mA
$V_{OH}(min)_C$	4.4 V	$V_{OH}(min)_T$	3.84 V

From the table

$$R_P(ON) = \frac{5 - 3.84}{|-4.0 \times 10^{-3}|} = 290\Omega \quad 175\Omega$$

$$R_N(ON) = \frac{0.33}{4 \times 10^{-3}} = 82.5\Omega$$

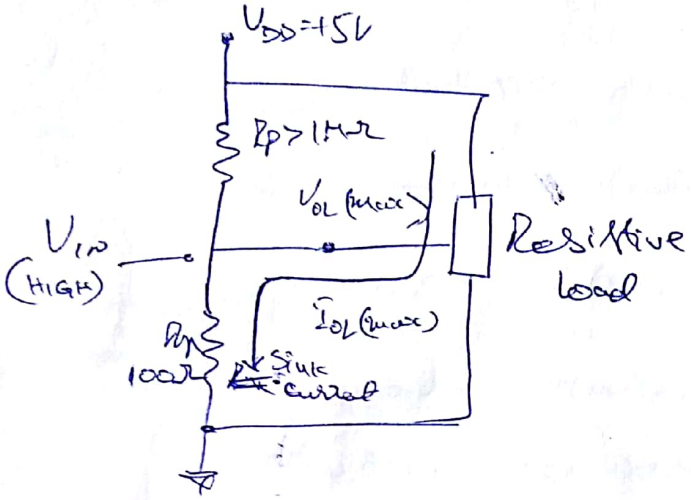
For TTL load

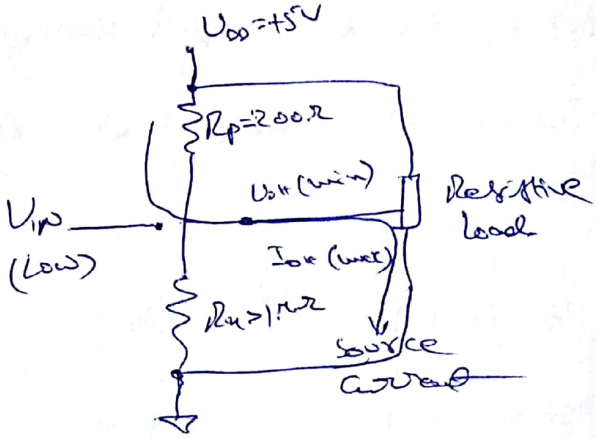
$$R_P(ON) = \frac{5 - 4.4}{0.02 \times 10^{-3}} = 0.03$$

$$R_N(ON) = \frac{0.1}{0.02 \times 10^{-3}} = 0.005$$

$I_{OL(max)}$: The maximum current that the o/p can sink in the Low state while still maintaining an o/p voltage smaller than $V_{OL(max)}$

$I_{OH(max)}$: The maximum current that the o/p can source/supply in the HIGH state while still maintaining an o/p voltage greater than $V_{OH(min)}$





Sink current: A device o/p is said to sink current when current flows from the power supply, through the load, and through the device o/p to ground.

$$\text{Sink current} = I_{Sink} = I_{OL(max)} = \frac{V_{OL(max)}}{R_N} \quad (\text{or}) \quad \frac{V_{out}}{R_N}$$

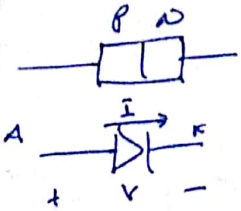
$$= \frac{0.43V}{100} = \underline{4.3\ mA}$$

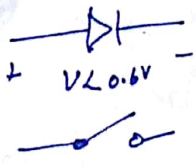
Source current: The o/p is said to source current when current flows from the power supply, out of the device o/p, & through the load to ground.

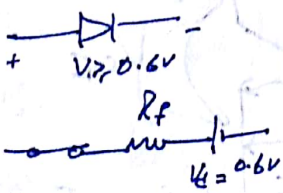
$$\text{Source current} = I_{source} = I_{OH(max)} = \frac{V_{out}}{R_p} = \frac{V_{OH(min)}}{R_p}$$

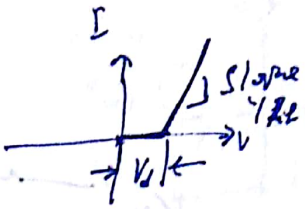
$$= \frac{V_{DD} - V_{out}}{R_p}$$

$$= \frac{5 - 4.61}{200} = \frac{0.39}{200} = \underline{1.95\ mA}$$





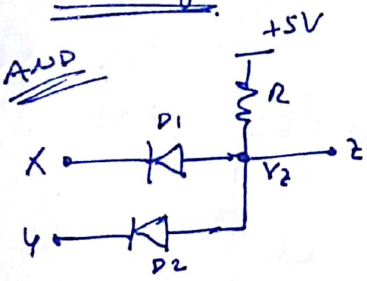




0-2 −0
2-3 −1
3-5 −1

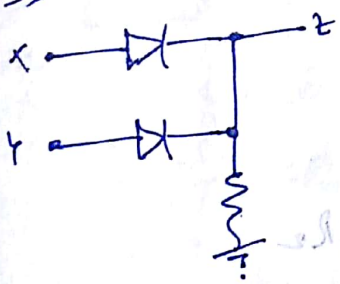
Diode logic

AND



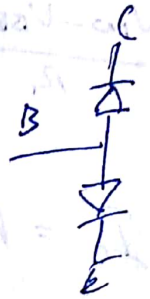
X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

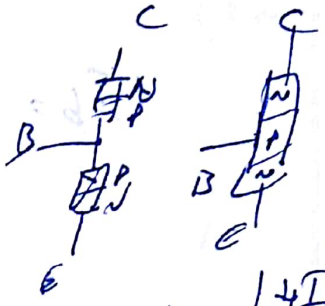
OR

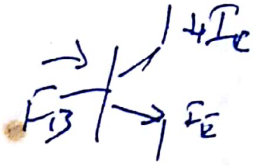


X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

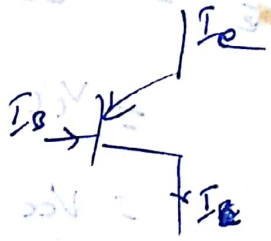
Transistor logic



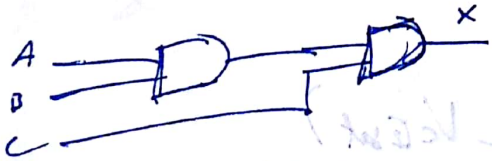


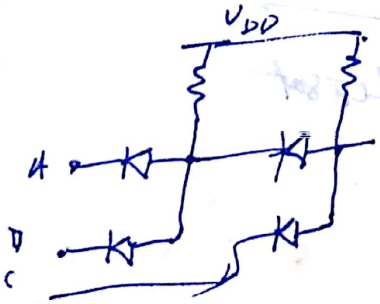


$I_E = I_B + I_C$



$I_C = I_B + I_E$



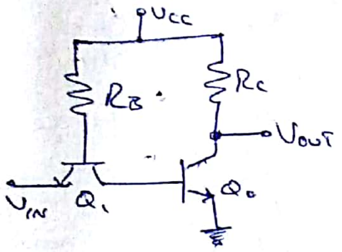


DCTL

RTL

DTL

Simple TTL Inverter



⇒ V_{IN} is Low → Q_1 is forward biased ... choice of R_B → Q_1 saturated for $V_{IN} = V_{Lo}$

V_{BE_0} i.e. Voltage across Emitter to Base transistor of Q_0 is $V_{BE_0} = V_{IN} + V_{CE1}$ (sat)

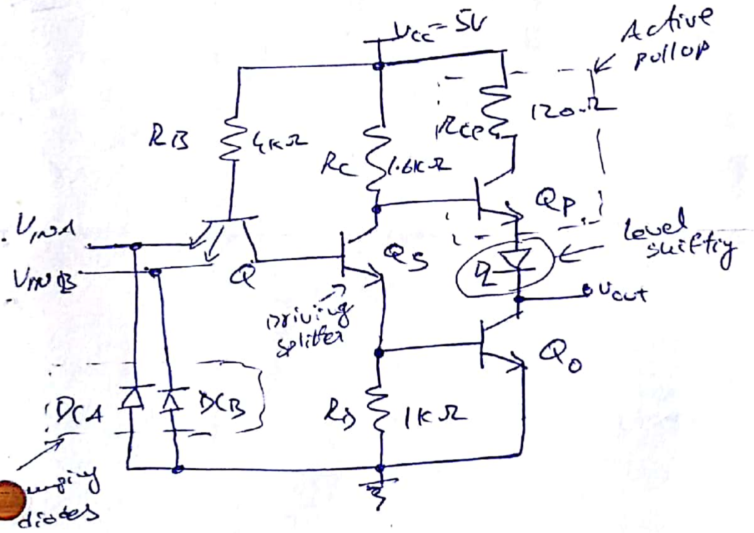
$\doteq V_{Lo} + 0.2V$

$= 0 + 0.2 = 0.2V$ this is insufficient to drive transistor Q_0 hence Q_0 is in cutoff

$\therefore V_{OH} = V_{OUT} = V_{CC}$

⇒ Increasing V_{IN} → V_{BEO} will reach 0.7V (ON) → Q_0 will forward bias

Standard TTL NAND gate (7400)



→ clamping diodes: D_C : To protect i/p from negative spikes

→ Discharge resistor R_D : Provides a discharge path for the base of Q_0 when switching from sat. to cutoff (OL to OH) Faster Speed.

→ Active Pull-up (Q_P & R_{CP}): Provides larger current to load gates w/o for faster charging of load gates and increased fan-out.

→ level shifting Diode (D_L) : Provides voltage isolation (0.7V) b/w Q_0 & Q_P to ensure no simultaneous conduction of both.

LEVEL SHIFTING DIODE

For OL, to ensure Q_P & Q_0 will not conduct simultaneously.

Q_S = sat, Q_P = cutoff ?? & Q_0 = Sat. $V_{out} = OL$

$V_{BEP} = V_{BP} - V_{EP}$ →(1)

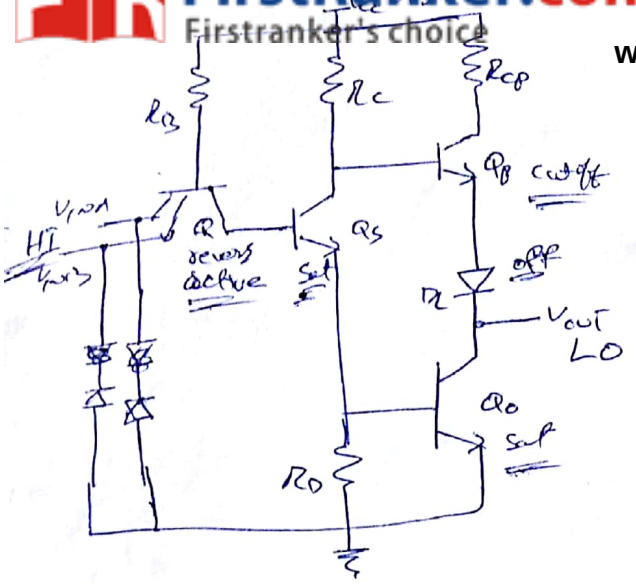
$V_{BP} = V_{BE_0}(sat) + V_{CES}(sat)$ →(2)

$V_{EP} = V_{CE_0}(sat) + V_{DL}(on)$ →(3)

$\therefore V_{BEP} = V_{BE_0}(sat) + V_{CES}(sat) - V_{CE_0}(sat) + V_{DL}(ON)$

$= V_{BE_0}(sat) + V_{DL}(ON) < V_{BE}(FA)$

→ Q_P cannot be conducting → cutoff because of level shifting DL.



Consider if either input of V_{INB} is low (i.e=0) then Q is ON & the base voltage is at its lowest value $V - I_1 R$. Here again R is used generally each element having distinct values in practice. The collector of Q & the base of Q_S are low so that Q_S is off. In this situation the base of Q_P is high turning this transistor ON while the base of Q_O is 0 & this transistor is off. In this situation the o/p is connected to the high voltage line 'V' & goes high.

if V_{INA} & V_{INB} Both are at High level Q is off, turning Q_S ON. This reduces the voltage on the base of Q_P & below its emitter voltage so that it becomes non-conducting. The base of Q_O on the other hand is driven up by an amount $I_2 R$ & becomes Conducting. ~~The base of Q_O on the other hand is driven~~ In this case the o/p is connected to ground through Q_O & the o/p is low.

V_{INA}	V_{INB}	Q	Q_S	Q_P	Q_O	O/P
0	0	ON	OFF	ON	OFF	1
0	1	ON	OFF	ON	OFF	1
1	0	ON	OFF	ON	OFF	1
1	1	OFF	ON	OFF	ON	0

Behavioural Modeling

Entity Declaration:-

1. An entity declaration describes the external interface of the entity, that is it gives the black-box view.
2. It specifies the name of the entity, the names of interface ports, their mode (i.e direction) and the type of ports.

The syntax is

```
entitity entity_name is
    generic (list-of-generics-and-their types);
    Port (list-of-interface-port-names-and-their
    types);
    Entity-item-declarations]

[begin
    entity-statements]

end entity-name;
```

Architecture Body:-

1. It describes the internal view of an entity.
2. It describes the functionality or the structure of the entity
3. The syntax is

```
architecture architecture-name of entity-name
    [architecture_item-declarations]
begin
    Concurrent-statements;
```

Process - statement
block - statement
Cocurrent - Procedure - Call
Concurrent - assertion - statement
Concurrent - signal - assignment - statement
Component - instation - statement
Generate - statement

End architecture_name;

3) Process Statement:-

1. A Process statement contains sequential state-ments that describe the functionality of a portion of an entity in sequential terms.

Syntax is

[Process_label:] Process [(sensitivity_list)]
[Process - item - declarations]
begin
Sequential - statements;
Variable - assignment - statement
Signal - assignment - statement
wait - statement
if - statement
case - statement
loop - statement
null - statement
exit - statement
next - statement
assertion - statement
Procedure - call - statement
Return - statement
end process [Proces - label];

3.2 Variable Assignment statement:-

1. Variables can be declared and used inside a process statement.
2. A variable is assigned a value using the variable assignment statement that typically has the form

Variable-object = expression

Consider the following process statement

```
Process (A)
    Variable events_on_A: Integer:=0;
begin
    events_on_A := events_on_A+1;
end process;
```

3.3. Signal Assignment statement:-

1. Signals are assigned values using a signal assignment statement the simplest form of signal assignment statement is

Signal-object <= expression [after delay-value];

3.4 Wait Statement:-

1. A process may be suspended by means of a sensitivity list.
2. That is, when a process has a sensitivity list.
3. It always suspends after executing the last sequential statement in the process.
4. The wait statement provides an alternative way to suspend the execution of a process.

www.FirstRanker.com

There are three basic forms of the wait statement

wait on Sensitivity_list;
wait until boolean_expression;
wait for time_expression;

5. They may also be combined in a single wait statement.

3.5 If statement:

1. An 'if' statement selects a sequence of statements for execution based on the value of a condition.
2. The condition can be any expression that evaluates to a boolean value.

The general form of an if statement is

```
if boolean_expression
        sequential_statements
elsif boolean_expression then
        sequential_statements
else
        sequential_statements
end if;
```

3.6 Case statement:

The format of a case statement is

```
case expression is
    when choices => seq_statements
    when choice => seq_statements
    --- can have any no. of statements
    when others => seq_statements
end case;
```

3.7 Null Statement :-

1. The statement
   null; is a sequential statement that does not cause any action to take place and execution continues with the next statement.

3.8 Loop statement :-

1. A loop statement is used to iterate through a set of sequential statements.
2. The syntax of a loop statement is
   [loop-label :] iteration-scheme that has the form for identifier. in range.

3.9 Exit statement :-

1. The exit statement is a sequential statement that can be used only inside a loop.
2. It causes execution to jump out of the innermost loop or the loop whose label is specified.
   The syntax for an exit statement is
   exit [loop-label] [when condition];

3.10 Next statement :-

1. The next statement is also a sequential statement that can be used only inside a loop.
2. The syntax is the same as that for the exit statement except that the keyword next replaces the keyword exit.

The Syntax for Next statement

next [loop-label] [when Condition];

3.11 Assertion Statement:-

1. Assertion statements are useful in modeling constrains of an entity.

The Syntax is

assert boolean-expression
[report string-expression]
[severity expression];

3.12 More on Signal Assignment Statement:-

1. Signal assignment statement used inside a process that illustrated the delta delay model.
2. Aside from the delta delay model, there are two other types of delay models that can be used with signal assignments, inertial and transport.

3.12.1 Inertial Delay Model:-

1. In this delay model the delays often found in switching circuits.
2. It represents the time for which an input value must be stable before the value is allowed to propagate to the output. Z <= A after 10ns.

delay = 10ns

3.12.2 Transport Delay Model :-

1. Transport delay models the delays in hardware that do not exhibit any inertial delay.

2. This delay represents pure propagation delay, that is any changes on an input is transported to the output, no matter how small, after the specified delay.

Z ⇐ transport A after 10ns

delay = 10ns

3.12.3 Creating Signal waveforms :-

1. In all examples of signal assignment statements that we have seen so far, we have always assigned a single value to a signal; this need not be so.

Phase1 ⇐ '0', '1' after 8ns, '0' after 13ns, '1' after 50ns;

3.12.4 Signal drivers :-

1. The driver of a signal holds its current value and all its future values as a sequence of one or more transactions, where each transaction identifies the value to appear on the signal along with the time at which the value is to appear.

```
Proc1
begin
    Reset ⇐ 3 after 5ns, 21 after 10ns, 14 after 17ns;
end proc1;
```

3.13 Other Sequential Statements :-

1. There are two other forms of sequential statements.
   (i) Procedure Call Statement
   (ii) Return statement.

3.14 Multiple Processes :-

1. A process statement is a concurrent statement, it is possible to have more than one process in within an architecture body.
2. This makes it possible to capture the behaviour of interfacing processes.