Module 4 : Propagation Delays in MOS
Lecture 17 : Pseudo NMOS Inverter

Objectives

In this lecture you will learn the following

- Introduction
- Different Configurations with NMOS Inverter
- Worries about Pseudo NMOS Inverter
- Calculation of Capacitive Load

17.1 Introduction

The inverter that uses a **p**-device pull-up or load that has its gate permanently ground. An **n**-device pull-down or driver is driven with the input signal. This roughly equivalent to use of a depletion load is Nmos technology and is thus called '**Pseudo-NMOS**'. The circuit is used in a variety of CMOS logic circuits. In this, PMOS for most of the time will be linear region. So resistance is low and hence RC time constant is low. When the driver is turned on a constant DC current flows in the circuit.

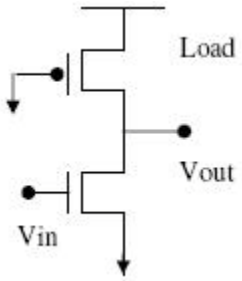
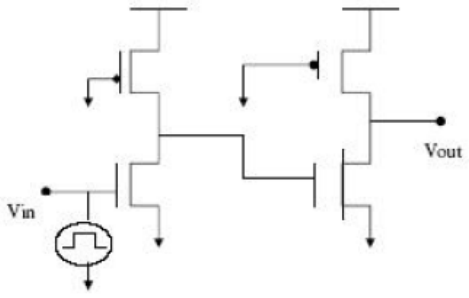


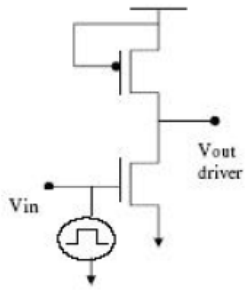
Fig 17.1: CMOS Inverter Circuit

17.2 Different Configurations with NMOS Inverter

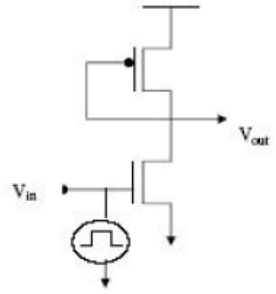
Cascade pseudo NMOS invertor:



Saturated n-mosNMOS invertor:

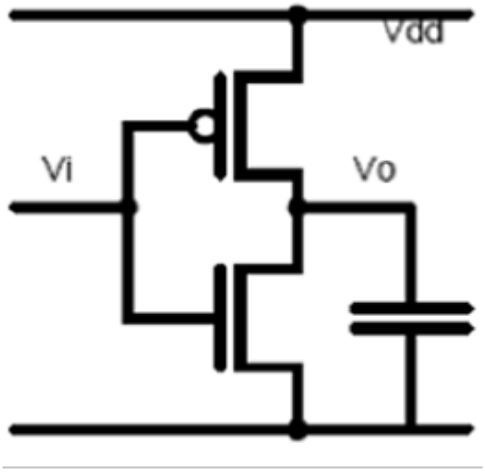


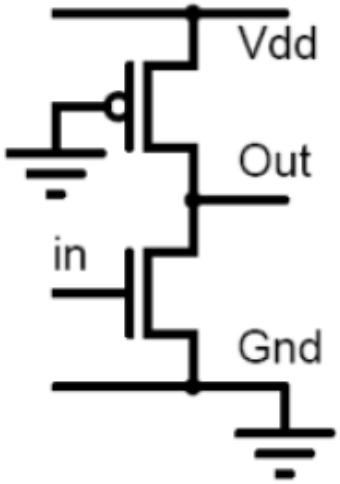
More saturated NMOS Load invertor:



17.3 CMOS Summary

Logic consumes no static power in CMOS design style. However, signals have to be routed to the n pull down network **as well as** to the p pull up network. So the load presented to every driver is high. This is exacerbated by the fact that n and p channel transistors cannot be placed close together as these are in different wells which have to be kept well separated in order to avoid latchup.





17.4 Pseudo nMOS Design Style

The CMOS pull up network is replaced by a single pMOS transistor with its gate grounded. Since the pMOS is not driven by signals, it is always ‘on'. The effective gate voltage seen by the pMOS transistor is Vdd. Thus the overvoltage on the p channel gate is always Vdd -VTp. When the nMOS is turned ‘on', a direct path between supply and ground exists and static power will be drawn. However, the dynamic power is reduced due to lower capacitive loading.

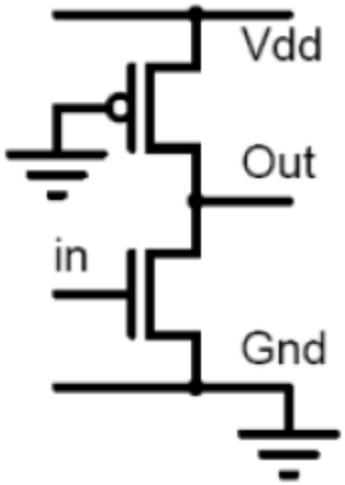
17.5 Static Characteristics

As we sweep the input voltage from ground to, we encounter the following regimes of operation:

- nMOS ‘off’
- nMOS saturated, pMOS linear
- nMOS linear, pMOS linear
- nMOS linear, pMOS saturated

17.6 Low Input

- When the input voltage is less than VTn.
- The output is 'high' and no current is drawn from the supply.
- As we raise the input just above VTn, the output starts falling.
- In this region the nMOS is saturated, while the pMOS is linear.



17.7 nMOS saturated, pMOS linear

The input voltage is assumed to be sufficiently low so that the output voltage exceeds the saturation voltage Vi - VTn. Normally, this voltage will be higher than VTp, so the p channel transistor is in linear mode of operation. Equating currents through the n and p channel transistors, we get

$$K_p\left[(V_{dd}-V_{Tp})(V_{dd}-V_o)-\frac{1}{2}(V_{dd}-V_o)^2\right]=\frac{K_n}{2}(V_i-V_{Tn})^2$$

defining $V_1 \equiv V_{dd}-V_o$ and $V_2 \equiv V_{dd}-V_{Tp}$, we get

$$\frac{1}{2}V_1^2-V_2V_1+\frac{\beta}{2}(V_i-V_{Tn})^2=0$$

$$\frac{1}{2}V_1^2-V_2V_1+\frac{\beta}{2}(V_i-V_{Tn})^2=0$$

The solutions are:

$$V_1 = V_2 \pm \sqrt{V_2^2 - \beta(V_i - V_{Tn})^2}$$

substituting the values of V1 and V2 and choosing the sign which puts V0 in the correct range, we get

$$V_o = V_{Tp} + \sqrt{(V_{dd} - V_{Tp})^2 - \beta(V_i - V_{Tn})^2}$$

As the input voltage is increased, the output voltage will decrease.
The output voltage will fall below Vi – Vtnwhen

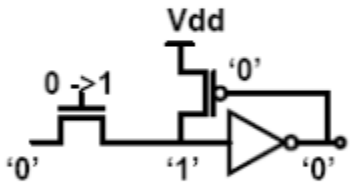
$$V_i > V_{Tn} + \frac{V_{Tp} + \sqrt{V_{Tp}^2 + (\beta + 1)V_{dd}(V_{dd} - 2V_{Tp})}}{\beta + 1}$$

The nMOS is now in its linear mode of operation. The derived equation does not apply beyond this input voltage.

Need for ratioing

The use of pMOS pullup brings up another problem.

Consider the equivalent circuit when the inverter output is 'low' and the pMOS is 'on'.



If the final output is 'low', the pMOS pullup is 'on'. Now if the multiplexer output wants to go 'low', it has to fight the pMOS pullup - which is trying to keep this node 'high'.

In fact, the multiplexer n transistor and the pull up p transistor constitute a pseudo nMOS inverter.

Therefore, the multiplexer output cannot be pulled low unless the transistor geometries are appropriately ratioed.

A. N. Chandorkar

nMOS linear, pMOS saturated

As the input voltage is raised still further, the output voltage will fall below V_{Tp}. The pMOS transistor is now in saturation regime. Equating currents, we get

$$K_n\left[(V_i - V_{Tn})V_o - \frac{1}{2}V_o^2\right] = \frac{K_p}{2}(V_{dd} - V_{Tp})^2$$

which gives

$$\frac{1}{2}V_o^2 - (V_o - V_{Tn})V_o + \frac{(V_{dd} - V_{Tp})^2}{2\beta}$$

This can be solved to get

$$V_o = (V_i - V_{Tn}) - \sqrt{(V_i - V_{Tn})^2 - (V_{dd} - V_{Tp})^2/\beta}$$

Noise Margins

We find points on the transfer curve where the slope is -1. When the input is low and output high, we should use

$$V_o = V_{Tp} + \sqrt{(V_{dd} - V_{Tp})^2 - \beta(V_i - V_{Tn})^2}$$

Differentiating this equation with respect to *Vi* and setting the slope to -1, we get

$$V_{iL} = V_{Tn} + \frac{V_{dd} - V_{Tp}}{\sqrt{\beta(\beta + 1)}}$$

and

$$V_{oH} = V_{Tp} + \sqrt{\frac{\beta}{\beta + 1}}\,(V_{dd} - V_{Tp})$$

When the input is high and the output low, we use

$$V_o = (V_i - V_{Tn}) - \sqrt{(V_i - V_{Tn})^2 - (V_{dd} - V_{Tp})^2/\beta}$$

Differentiating with respect to V_i and setting the slope to -1, we get

$$V_{iH} = V_{Tn} + \frac{2}{\sqrt{3\beta}}(V_{dd} - V_{Tp})$$

and

$$V_{oL} = \frac{(V_{dd} - V_{Tp})}{\sqrt{3\beta}}$$

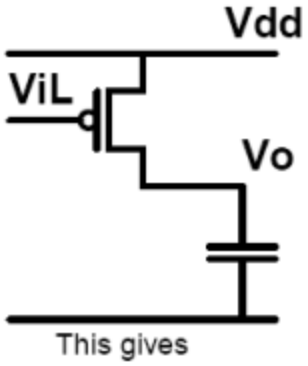
Ratioed Logic

To make the output 'low' value lower than V_{Tn}, we get the condition

$$\beta > \frac{1}{3}\left(\frac{V_{dd} - V_{Tp}}{V_{Tn}}\right)^2$$

- This condition on values of β places a requirement on the ratios of widths of n and p channel transistors.
- The logic gates work properly only when this equation is satisfied.
- Therefore this kind of logic is also called 'ratioed logic'.
- In contrast, CMOS logic is called ratioless logic because it does not place any restriction on the ratios of widths of n and p channel transistors for static operation.
- The noise margin for pseudo nMOS can be determined easily from the expressions for V_{iL}, V_{oL}, V_{iH}, V_{oH}.

Rise Time

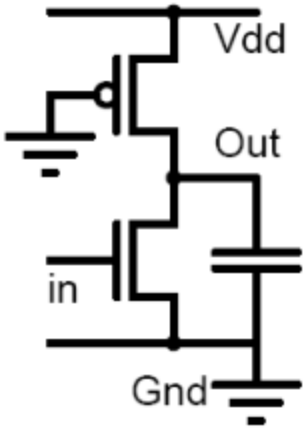


When the input is low, the nMOS is off and the output rises from 'low' to 'high'.

The situation is identical to the charge up condition of a CMOS gate with the pMOS being biased with its gate at 0V.

$$\tau_{rise} = \frac{C}{K_p(V_{dd} - V_{Tp})}\left[\frac{2V_{Tp}}{V_{dd} - V_{Tp}} + \ln\frac{V_{dd} + V_{oH} - 2V_{Tp}}{V_{dd} - V_{oH}}\right]$$

Fall Time



Calculation of fall time is complicated by the fact that the pMOS load continues to dump current in the output node, even as the nMOS tries to discharge the output capacitor.
The nMOS needs to sink the discharge current as well as the drain current of the pMOS transistor.
Simplifying assumption:
pMOS current remains constant at its saturation value through the entire discharge process.

(This will result in a slightly pessimistic value of discharge time).

Fall Time

If we assume that the pMOS current remains constant at its saturation value,

$$I_p = \frac{K_p}{2}(V_{dd} - V_{Tp})^2$$

We can write the KCL equation at the output node as:

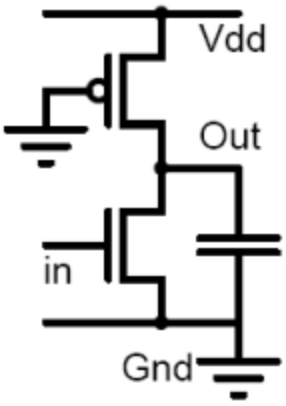
$$I_n - I_p + C\frac{dV_o}{dt} = 0$$

which gives

$$\frac{\tau_{fall}}{C} = -\int_{V_{dd}}^{V_{oL}} \frac{dV_o}{I_n - I_p}$$

We define $V_1 \equiv V_i - V_{Tn}$ and $V_2 \equiv V_{dd} - V_{Tp}$.

Fall Time



The integration range can be divided into two regimes.

- nMOS is saturated when $V_1 \leq V_o < V_{dd}$.
- It is in the linear regime when $V_{oL} < V_o < V_1$.

Fall Time

$$\frac{\tau_{fall}}{C} = -\int_{V_{dd}}^{V_1} \frac{dV_o}{\frac{1}{2}K_n V_1^2 - I_p} - \int_{V_1}^{V_{oL}} \frac{dV_o}{K_n(V_1 V_o - \frac{1}{2}V_o^2) - I_p}$$

so,

$$\frac{\tau_{fall}}{C} = \frac{V_{dd} - V_1}{\frac{1}{2}K_n V_1^2 - I_p} + \int_{V_{oL}}^{V_1} \frac{dV_o}{K_n(V_1 V_o - \frac{1}{2}V_o^2) - I_p}$$

Pseudo nMOS Inverter design

- We design the basic inverter and then scale device sizes based on the logic function being designed.
- The load device size is calculated from the rise time.

$$\tau_{rise} = \frac{C}{K_p(V_{dd} - V_{Tp})}\left[\frac{2V_{Tp}}{V_{dd} - V_{Tp}} + \ln\frac{V_{dd} + V_{oH} - 2V_{Tp}}{V_{dd} - V_{oH}}\right]$$

- Given a value of τ_{rise}, operating voltages and technological constants, K_p and hence, the geometry of the p channel transistor can be determined.

Pseudo nMOS Inverter design

- Geometry of the n channel transistor can be determined from static considerations.

$$V_{oL} = (V_{iH} - V_{Tn}) - \sqrt{(V_{iH} - V_{Tn})^2 - (V_{dd} - V_{Tp})^2/\beta}$$

- We take $V_{oL} = V_{Tn}$, and calculate β.
- But $\beta \equiv K_n/K_p$ and K_p is already known.
- This evaluates K_n and hence, the geometry of the n channel transistor.

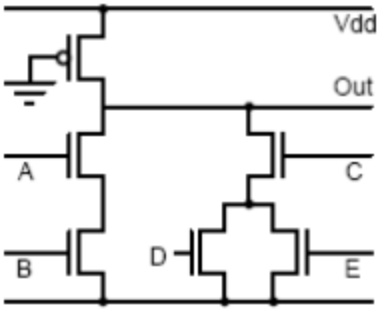
Conversion to other logic

- Once the basic pseudo nMOS inverter is designed, other logic gates can be derived from it.
- The procedure is the same as that for CMOS, except that it is applied only to nMOS transistors.
- The p channel transistor is kept at the same size as that for an inverter.

Conversion to other logic

- The logic is expressed as a sum of products with a bar (inversion) on top.
- For every '.' in the expression, we put the corresponding n channel transistors in series.
- For every '+', we put the n channel transistors in parallel.
- We scale the transistor widths up by the number of devices put in series.
- The geometries are left untouched for devices put in parallel.

$\overline{A.B+C.(D+E)}$ in pseudo-nMOS



- A and B are in series.
- The pair is in parallel with C which is in series with a parallel combination of D and E.

Implementation of $\overline{A.B+C.(D+E)}$ in pseudo-nMOS logic design style.

Complementary Pass gate Logic

- This logic family is based on multiplexer logic.
- Given a boolean function $F(x_1, x_2, \ldots, x_n)$, we can express it as:

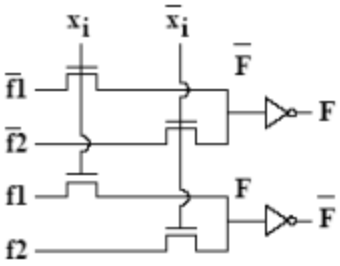
$$F(x_1, x_2, \ldots, x_n) = x_i \cdot f1 + \overline{x_i} \cdot f2$$

  where f1 and f2 are reduced expressions for F with x_i forced to 1 and 0 respectively.
- Thus, F can be implemented with a multiplexer controlled by x_i which selects f1 or f2 depending on x_i.
- f1 and f2 can themselves be decomposed into simpler expressions by the same technique.

Complementary Pass gate Logic

- To implement a multiplexer, we need both x_i and $\overline{x_i}$.
- Therefore, this logic family needs all inputs in true as well as in complement form.
- In order to drive other gates of the same type, it must produce the outputs also in true and complement forms.
- Thus each signal is carried by two wires.
- This logic style is called "Complementary Passgate Logic" or CPL for short.

Basic Multiplexer Structure



Pure passgate logic contains no 'amplifying' elements. Therefore, each logic stage degrades the logic level.

Hence, multiple logic stages cannot be cascaded.

We include conventional CMOS inverters to restore the logic level.

Ideally, the multiplexer should be composed of complementary pass gate transistors.

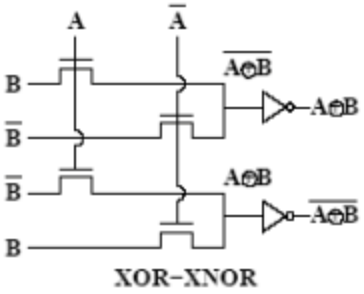
However, we shall use just n channel transistors as switches for simplicity.

Logic Design using CPL

- For any logic function, we pick one input as the control variable.
- Multiplexer inputs are decided by re-evaluating the function, forcing this variable to 1 and zero respectively.
- Since both true and complement outputs are generated by CPL, we need fewer types of gates.
- For example, we do not need separate gates for AND and NAND functions.
- The same applies to OR-NOR, and XOR-XNOR functions.

Implementation of XOR and XNOR

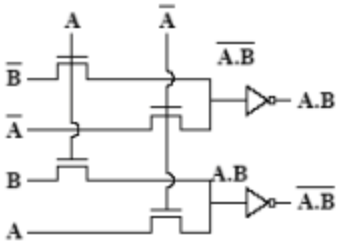
To take an example, let us consider the XOR-XNOR functions.



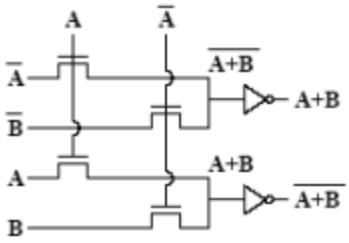
XOR–XNOR

- Because of the inverter, for XOR output, We calculate the XNOR function given by $A\overline{B}+\overline{A}.\overline{B}$.
- If we put A = 1, this reduces to B and for A = 0, it reduces to $\overline{B}$.
- For the XNOR output, we generate the XOR expression = $A\overline{B}+\overline{A}B$.
- The expression reduces to B for A = 1 and to B for A = 0.

Implementation of AND-NAND and OR-NOR



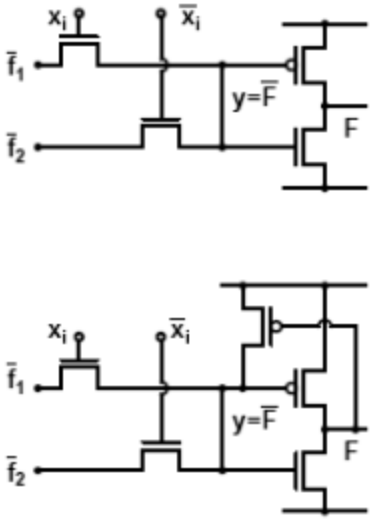
AND–NAND



OR–NOR

- For AND, the mux should output $\overline{A.B}$ to be inverted by the buffer. This reduces to $\overline{B}$ when A = 1 and to 1 (= $\overline{A}$) when A = 0.
- Implementation of NAND, OR and NOR functions follows along the same lines.

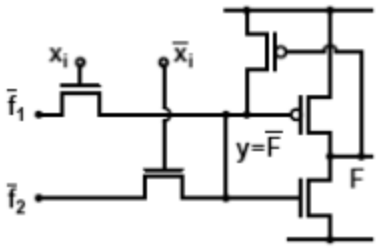
Buffer Leakage Current



- The high output of the multiplexer (y) cannot rise above V_{dd} - V_{Tn} because we use nMOS multiplexers.
- Consequently, the pMOS transistor in the buffer inverter never quite turns off.
- This results in static power consumption in the inverter.

This can be avoided by adding a pull up pMOS with the inverter.

Use of Pullup PMOS



- When the multiplexer output (y) is 'low', the inverter output (F) is high. The pMOS is off and has no effect.
- When the multiplexer output (y) goes 'high', the inverter output falls and turns the pMOS on.

Now, even though the multiplexer nMOS turns ‘off’ as y approaches V_{dd} - V_{Tn}, the pMOS remains ‘on’ and takes the inverter input (y) all the way to V_{dd}.

This avoids leakage in the inverter.

Recap

In this lecture you have learnt the following

- Introduction
- Different Configurations with NMOS Inverter
- Worries about Pseudo NMOS Inverter
- Calculation of Capacitive Load

Congratulations, you have finished Lecture 17.