Topic 6

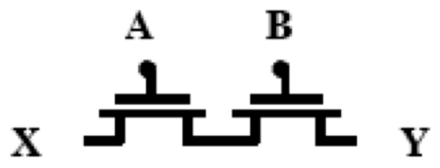
CMOS Static & Dynamic Logic Gates

Peter Cheung
Department of Electrical & Electronic Engineering
Imperial College London

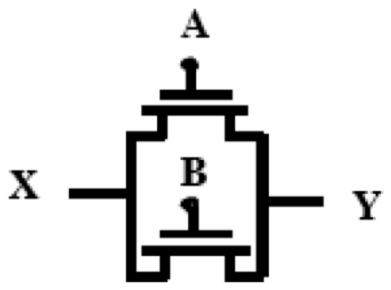
URL: www.ee.ic.ac.uk/pcheung/
E-mail: p.cheung@ic.ac.uk

NMOS Transistors in Series/Parallel Connection

- Transistors can be thought as a switch controlled by its gate signal
- NMOS switch closes when switch control input is high



Y = X if A and B

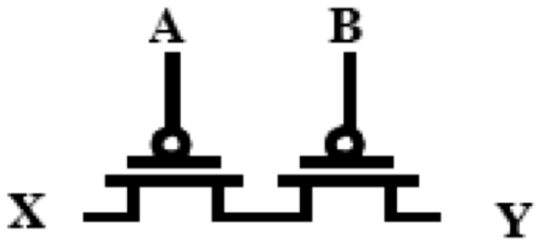


Y = X if A OR B

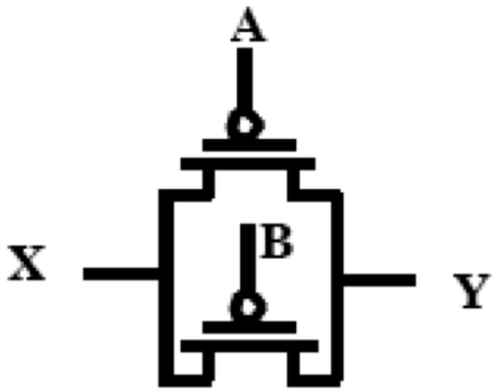
NMOS Transistors pass a “strong” 0 but a “weak” 1

PMOS Transistors in Series/Parallel Connection

PMOS switch closes when switch control input is low



Y = X if A AND B = A + B

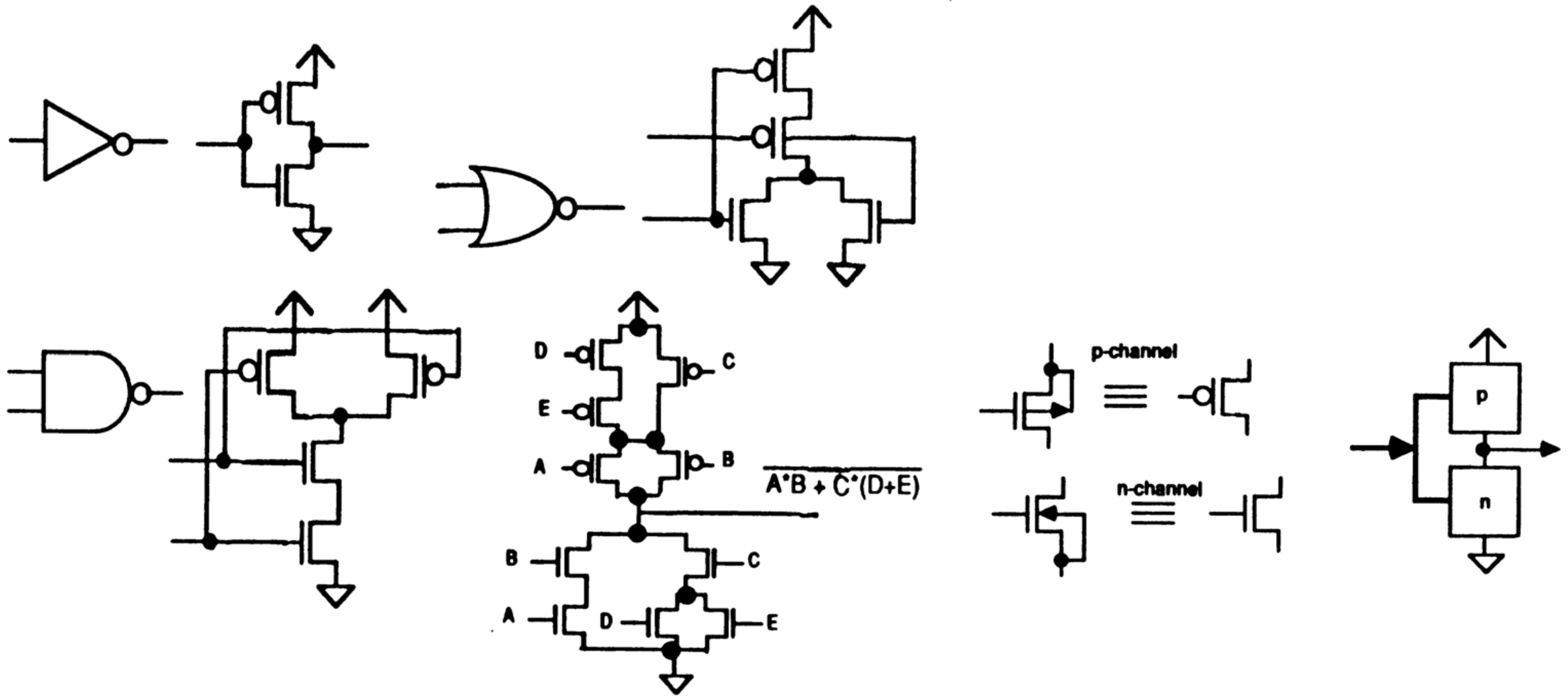


Y = X if A OR B = AB

PMOS Transistors pass a “strong” 1 but a “weak” 0

Static CMOS Circuit

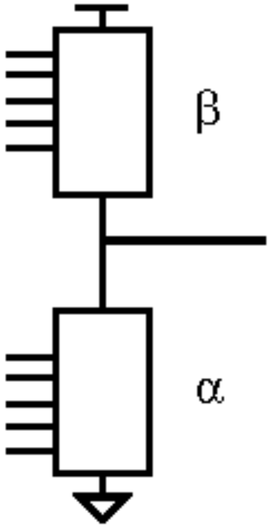
- Basic CMOS combinational circuits consist of:
  - Complementary pull-up (p-type) and pull-down (n-type)



Static CMOS

To build a logic gate $\bar{f}(x_1, \ldots, x_n)$, need to build two switch networks:

The pullup network connects the output to Vdd when f is false.



pMOS only, since only passes 1

The pulldown network connects the output to Gnd when f is true.

nMOS only, since only passes 0

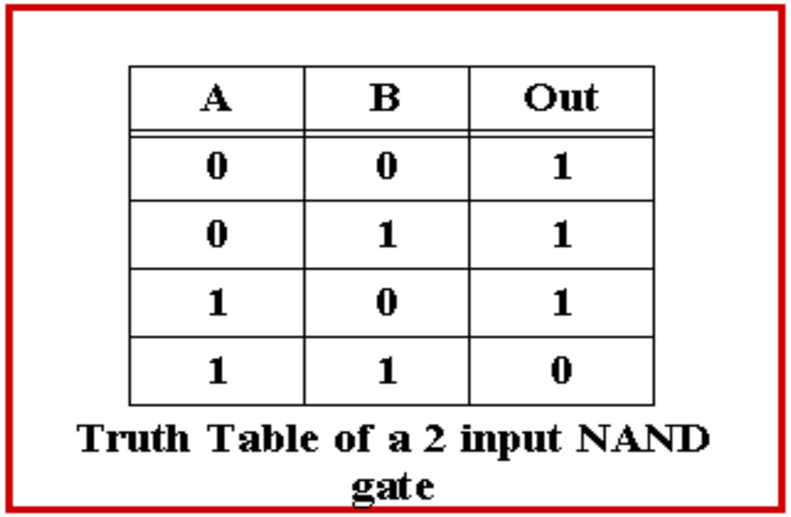
Pulldown

$$\alpha(x_1, \ldots, x_n) = f(x_1, \ldots, x_n)$$

Pullup

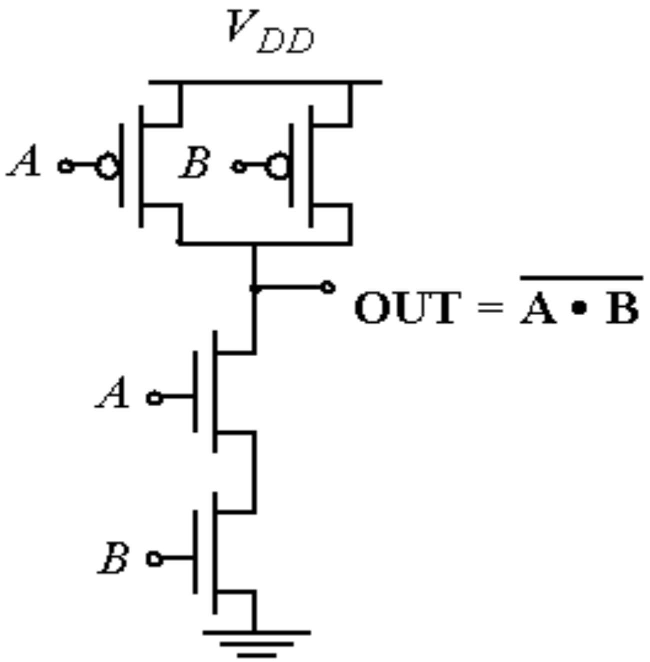
$$\beta(\bar{x}_1, \ldots, \bar{x}_n) = \bar{f}(x_1, \ldots, x_n)$$ (since pMOS invert inputs)

Example Gate: NAND



A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

Truth Table of a 2 input NAND gate

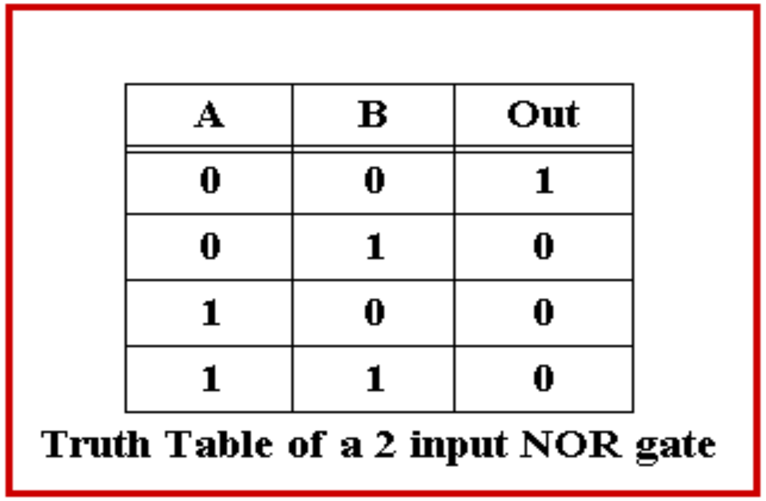


PDN: $G = A\,B \Rightarrow$ Conduction to GND

PUN: $F = \overline{A} + \overline{B} = \overline{AB} \Rightarrow$ Conduction to V_{DD}

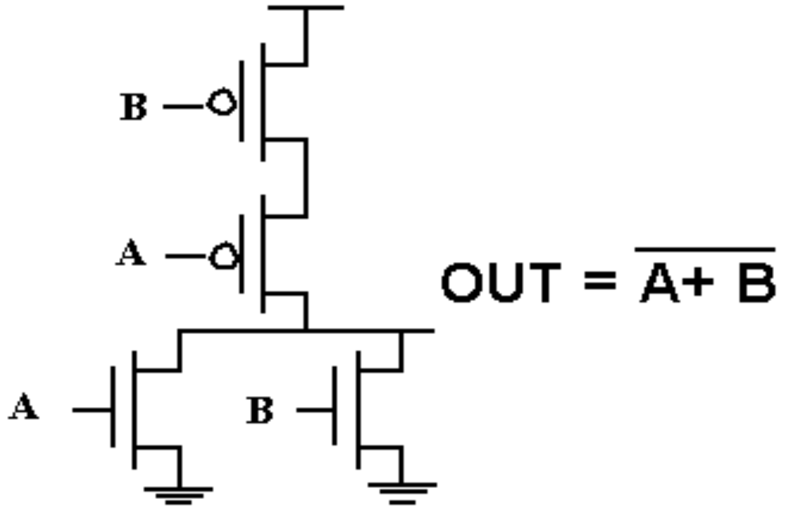
$$\overline{G(In_1, In_2, In_3, \ldots)} \equiv F(\overline{In_1}, \overline{In_2}, \overline{In_3}, \ldots)$$

Example Gate: NOR



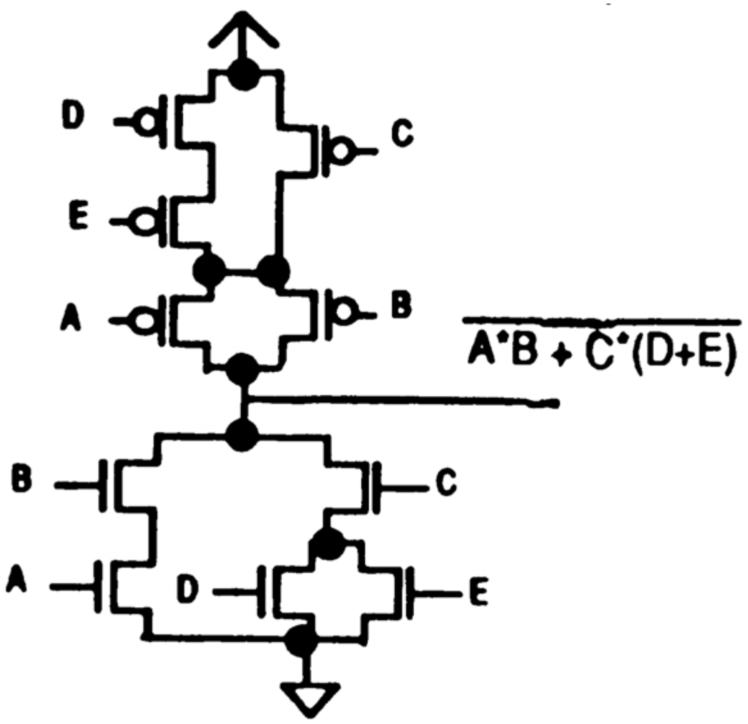
A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

Truth Table of a 2 input NOR gate

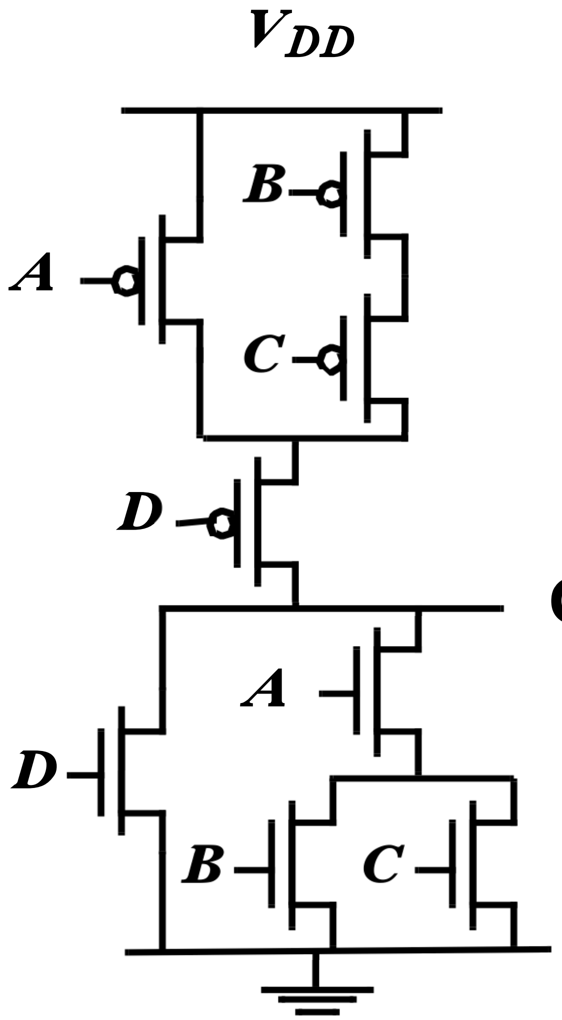


Complex Gate

- We can form complex combinational circuit function in a complementary tree. The procedure to construct a complementary tree is as follow:-
  - Express the boolean expression in an inverted form
  - For the n-transistor tree, working from the inner-most bracket to the outer-most term, connect the **OR** term transistors in parallel, and the **AND** term transistors in series
  - For the p-transistor tree, working from the inner-most bracket to the outer-most term, connect the **OR** term transistors in series, and the **AND** term transistors in parallel



Example Gate: COMPLEX CMOS GATE



$$\text{OUT} = \overline{D + A \cdot (B + C)}$$

Properties of Complementary CMOS Gates

1) High noise margins:

V_{OH} and V_{OL} are at V_{DD} and *GND*, respectively.

2) No static power consumption:

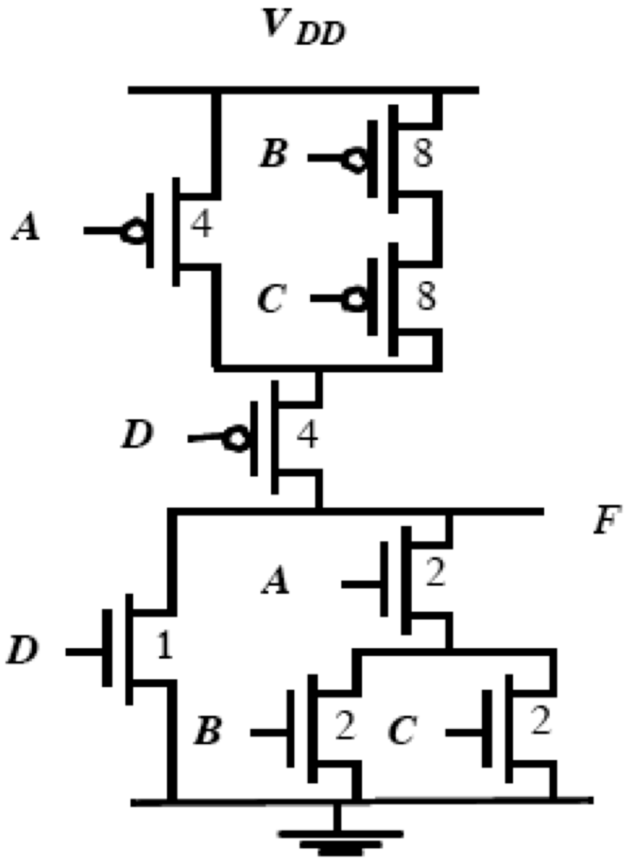
There never exists a direct path between V_{DD} and V_{SS} (*GND*) in steady-state mode.

3) Comparable rise and fall times:

(under the appropriate scaling conditions)

Transistor Sizing

- for symmetrical response (dc, ac)
- for performance

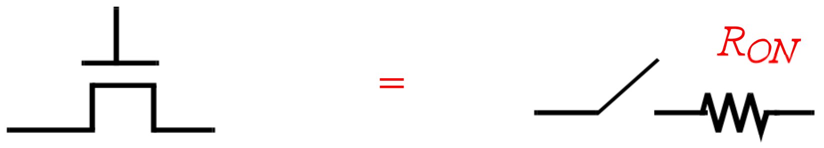


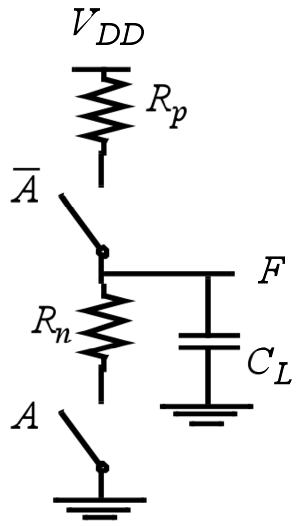
Input Dependent

Focus on worst-case

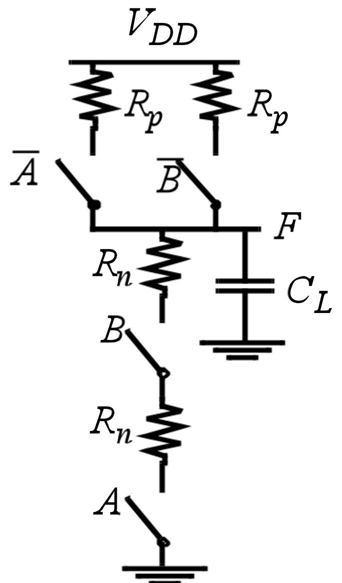
- assume $\mu_n = 2 * \mu_p$ (i.e. n-channel transistors has 2 times the transconductance as that of p-channel.)

Propagation Delay Analysis - The Switch Model

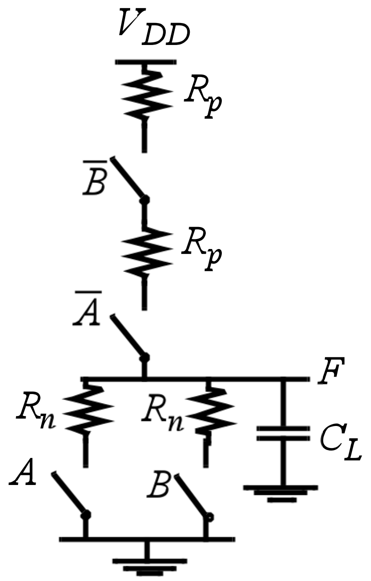




(a) Inverter



(b) 2-input NAND



(c) 2-input NOR

$$t_p = 0.69\, R_{on}\, C_L$$

(assuming that C_L dominates!)

What is the Value of R_{on}?

- *Depends strongly on the operating region*

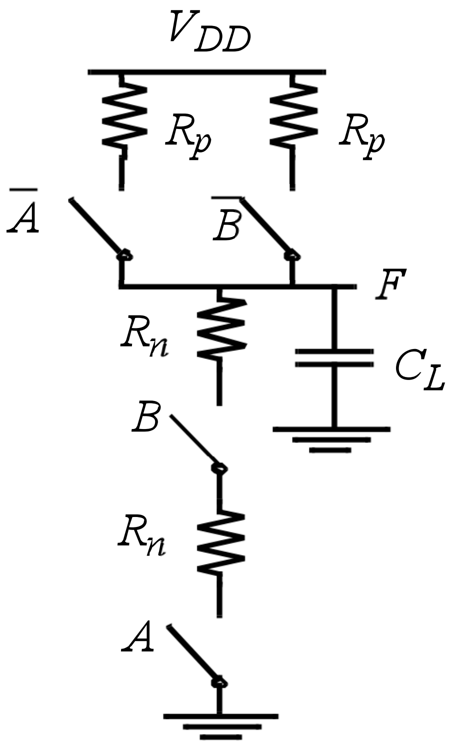
- *For hand analysis use a fixed value of R which it the average of the two end points of the transition*
- *Similar to the previous approach of averaging currents*

EXAMPLE: For t_{pHL} for an inverter, the R_{on} is:

$$R_{on} = \frac{1}{2}(R_{NMOS}(V_{out} = V_{DD}) + R_{NMOS}(V_{out} = V_{DD}/2))$$

$$= \frac{1}{2}\left(\left(\frac{V_{DS}}{I_D}\right)_{V_{out} = V_{DD}} + \left(\frac{V_{DS}}{I_D}\right)_{V_{out} = V_{DD}/2}\right)$$

Analysis of Propagation Delay



2-input NAND

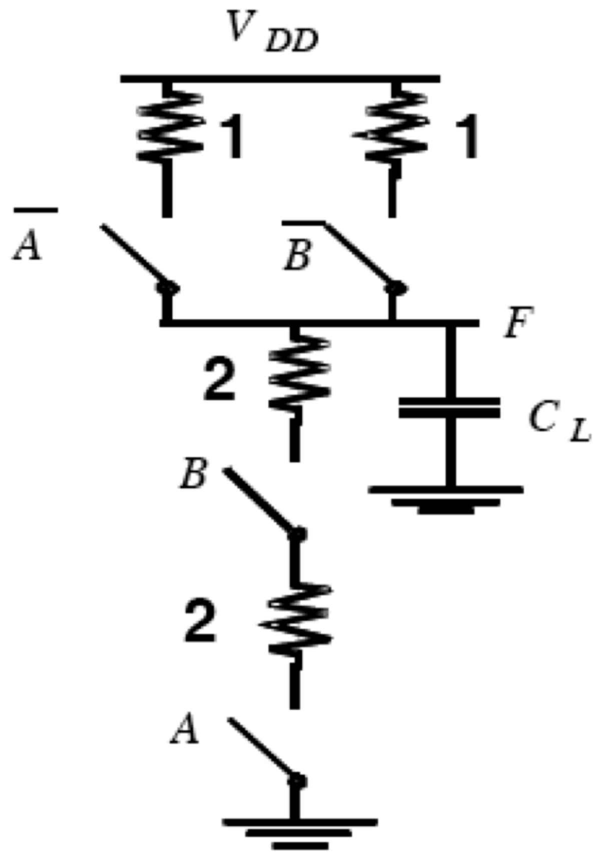
1. Assume R_n=R_p= resistance of minimum sized NMOS inverter
2. Determine "Worst Case Input" transition (Delay depends on input values)
3. Example: t_{pLH} for 2input NAND
   - Worst case when only ONE PMOS Pulls up the output node
   - For 2 PMOS devices in parallel, the resistance is lower

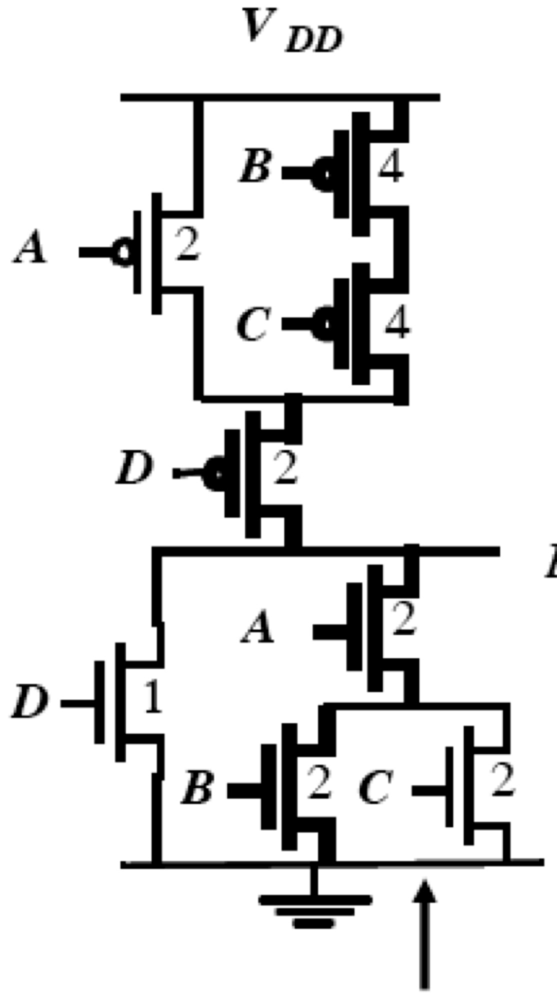
$$t_{pLH} = 0.69 R_p C_L$$

4. Example: t_{pHL} for 2input NAND
   - Worst case : TWO NMOS in series

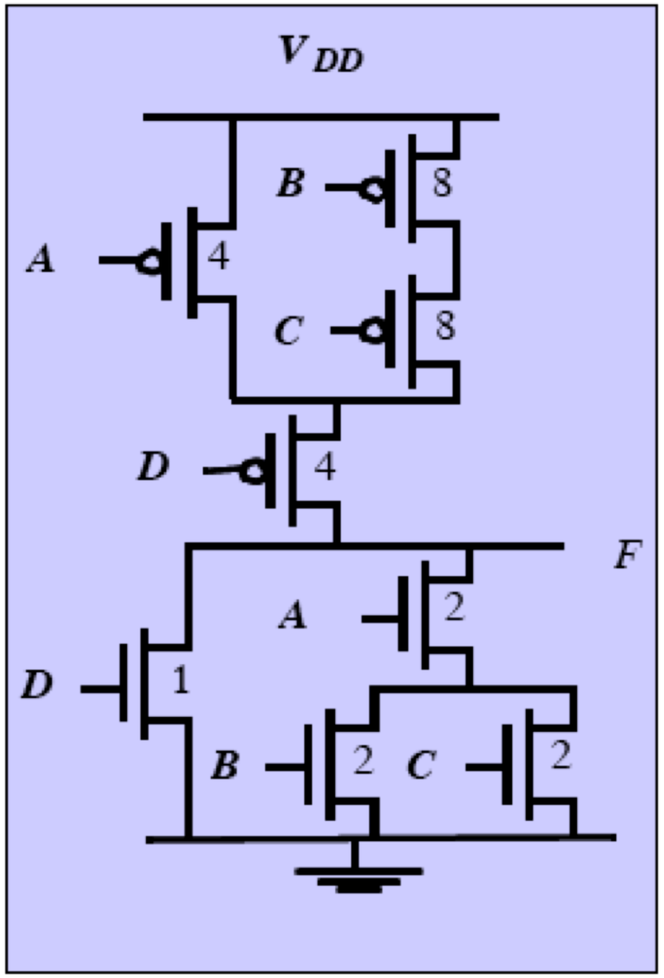
$$t_{pHL} = 0.69(2R_n)C_L$$

Design for Worst Case





Here it is assumed that $R_p = R_n$

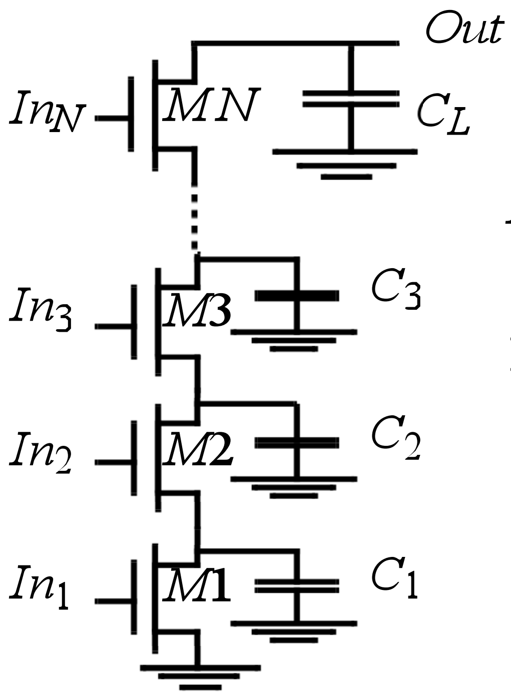


Fast Complex Gate - Design Techniques

- **Transistor Sizing:**

  As long as Fan-out Capacitance dominates

- **Progressive Sizing:**



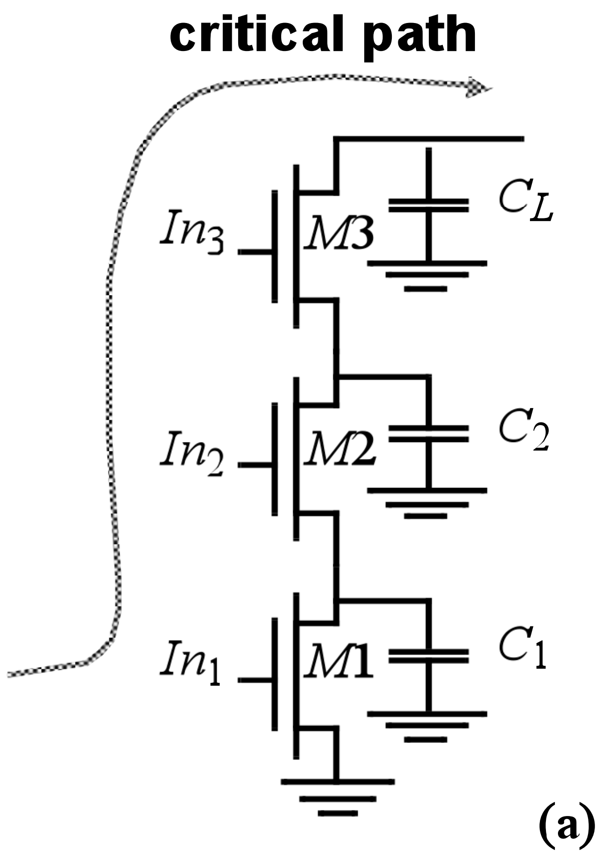
$M1 > M2 > M3 > MN$

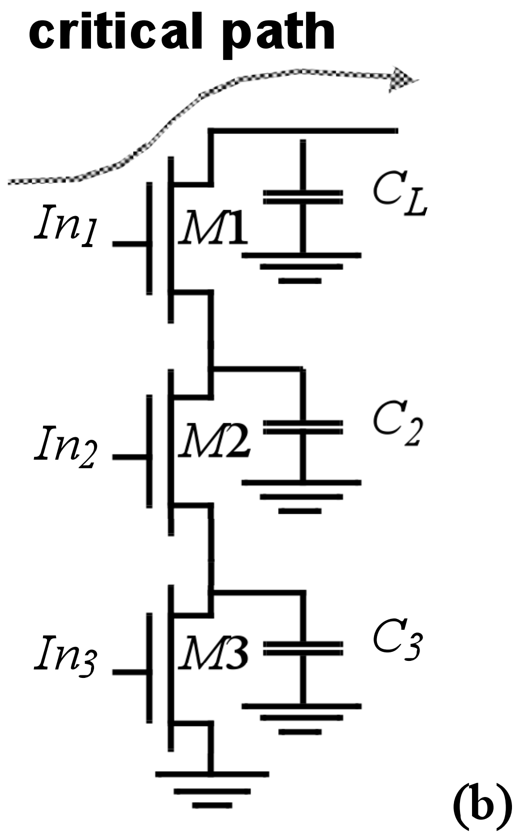
Distributed RC-line

Can Reduce Delay with more than 30%!

Fast Complex Gate - Design Techniques (2)

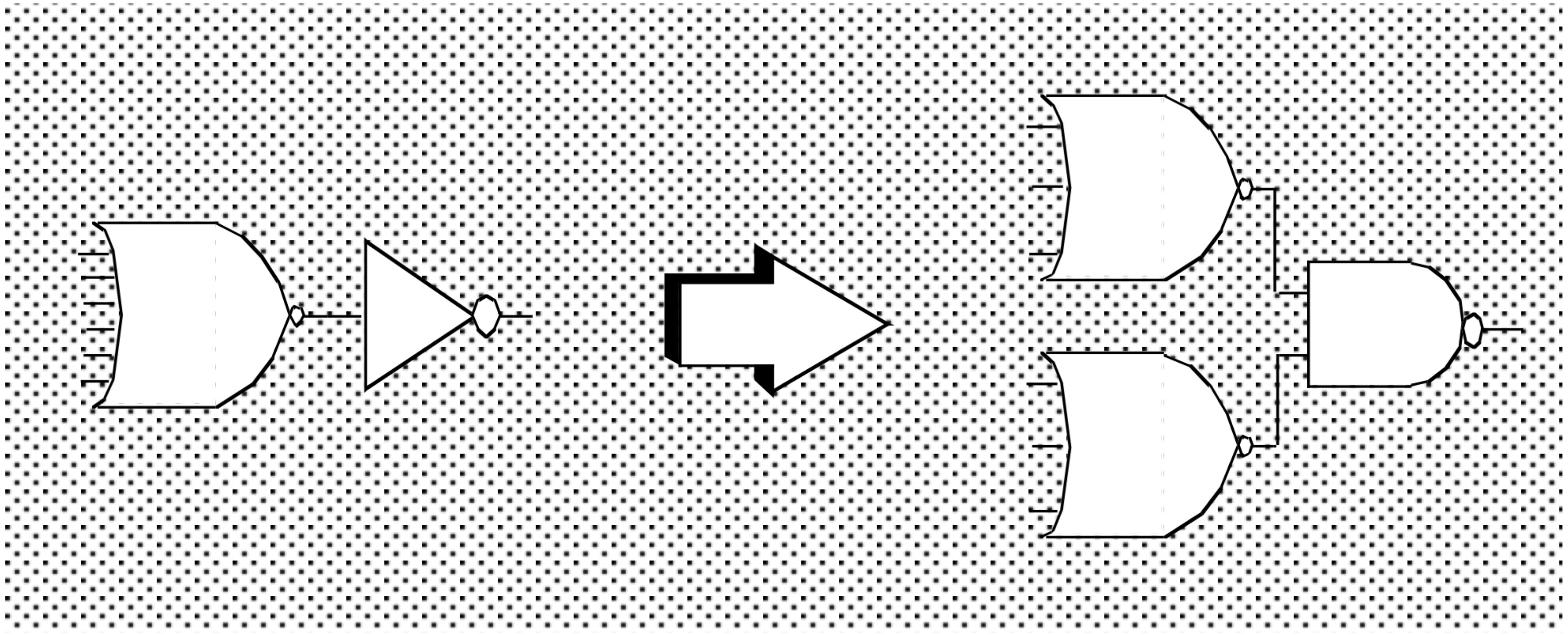
- **Transistor Ordering**





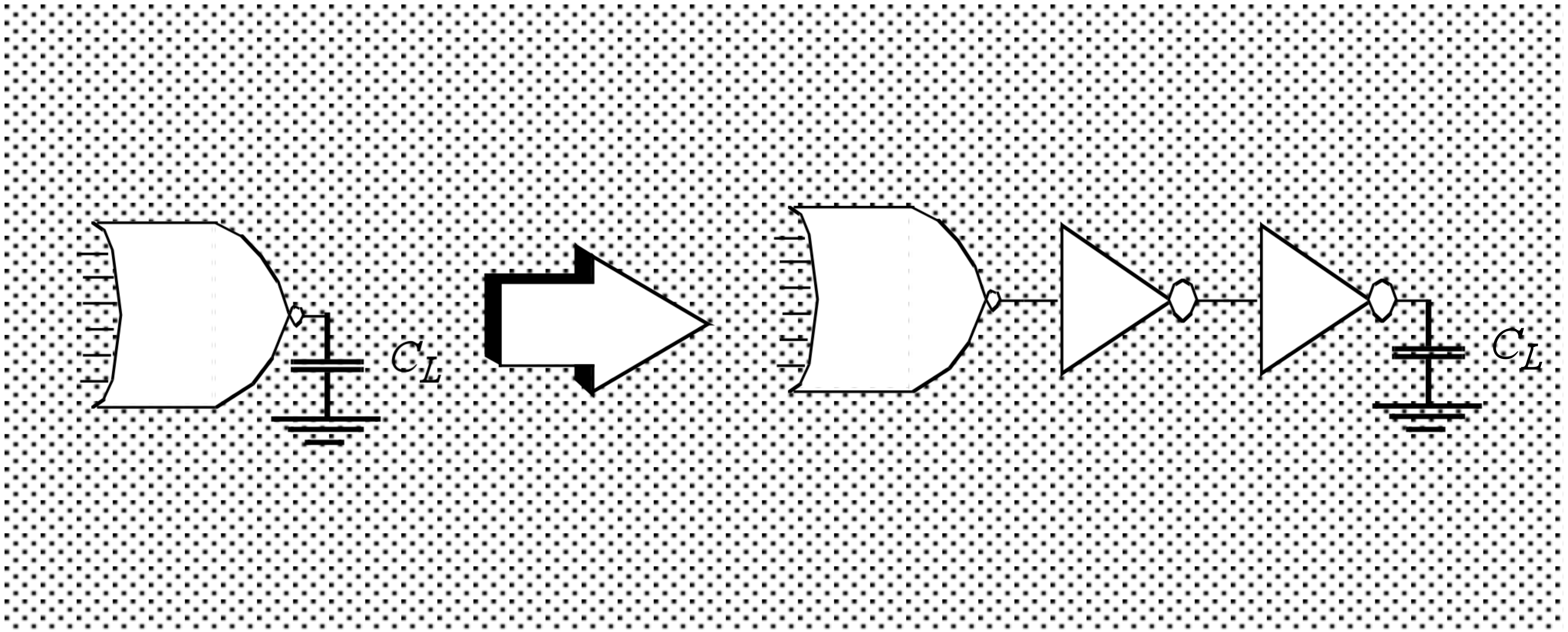
Fast Complex Gate - Design Techniques (3)

- **Improved Logic Design**

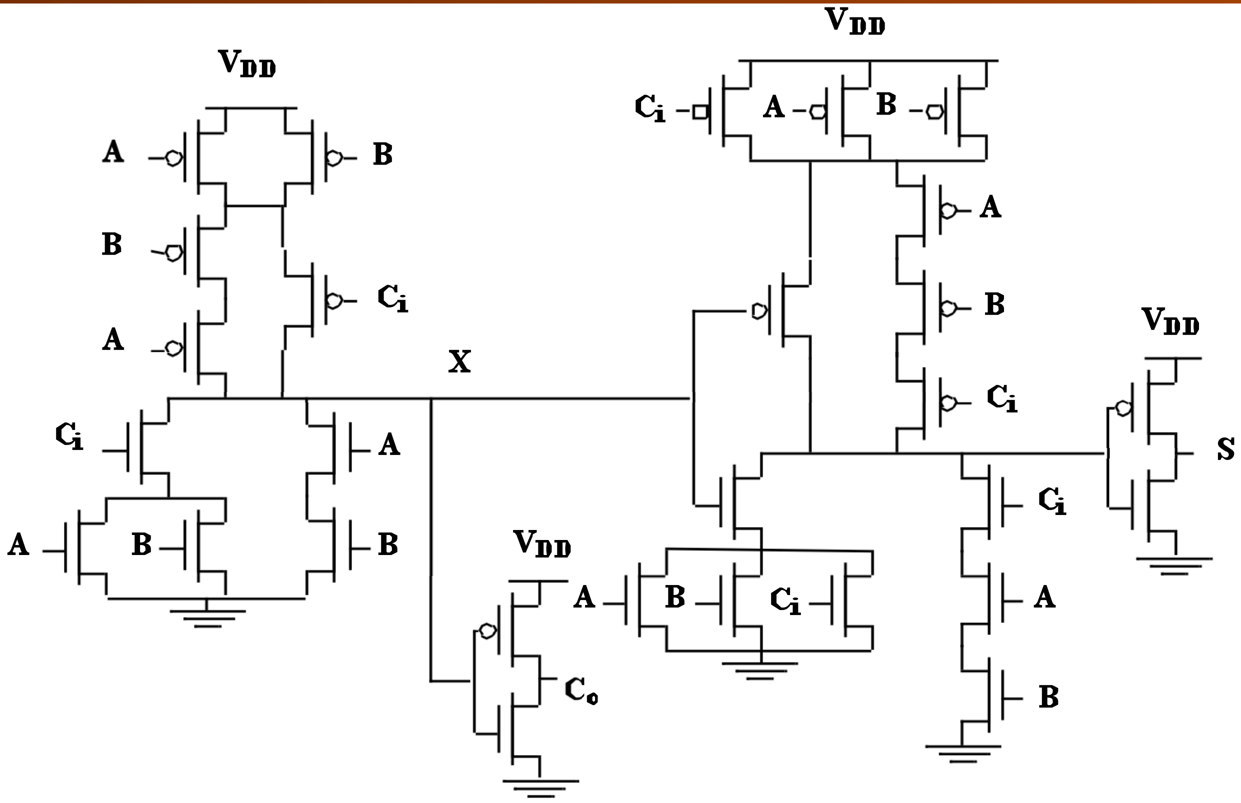


Fast Complex Gate - Design Techniques (4)

- **Buffering: Isolate Fan-in from Fan-out**



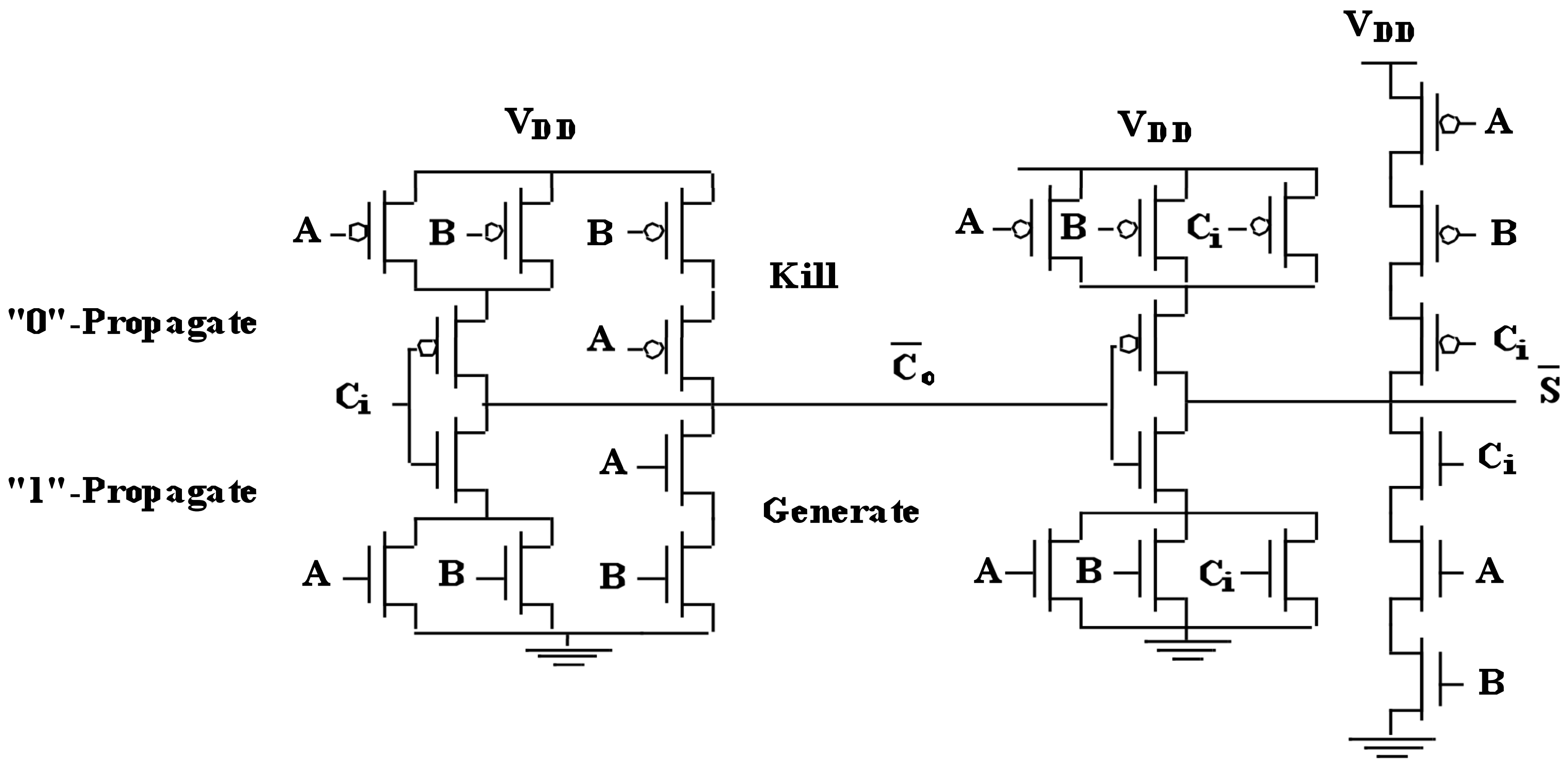
Example: Full Adder



$$C_o = AB + C_i(A+B)$$

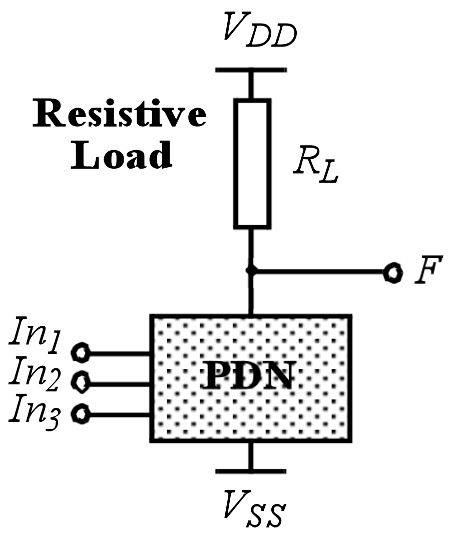
28 transistors

A Revised Adder Circuit

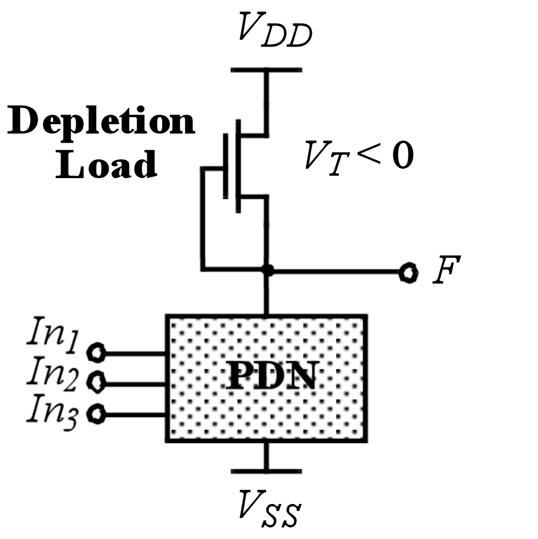


24 transistors

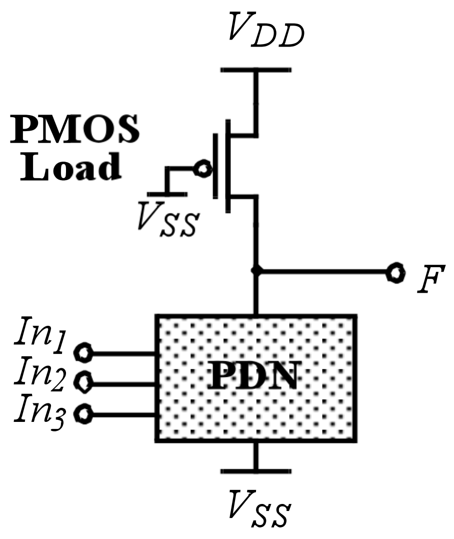
Ratioed Logic



(a) resistive load



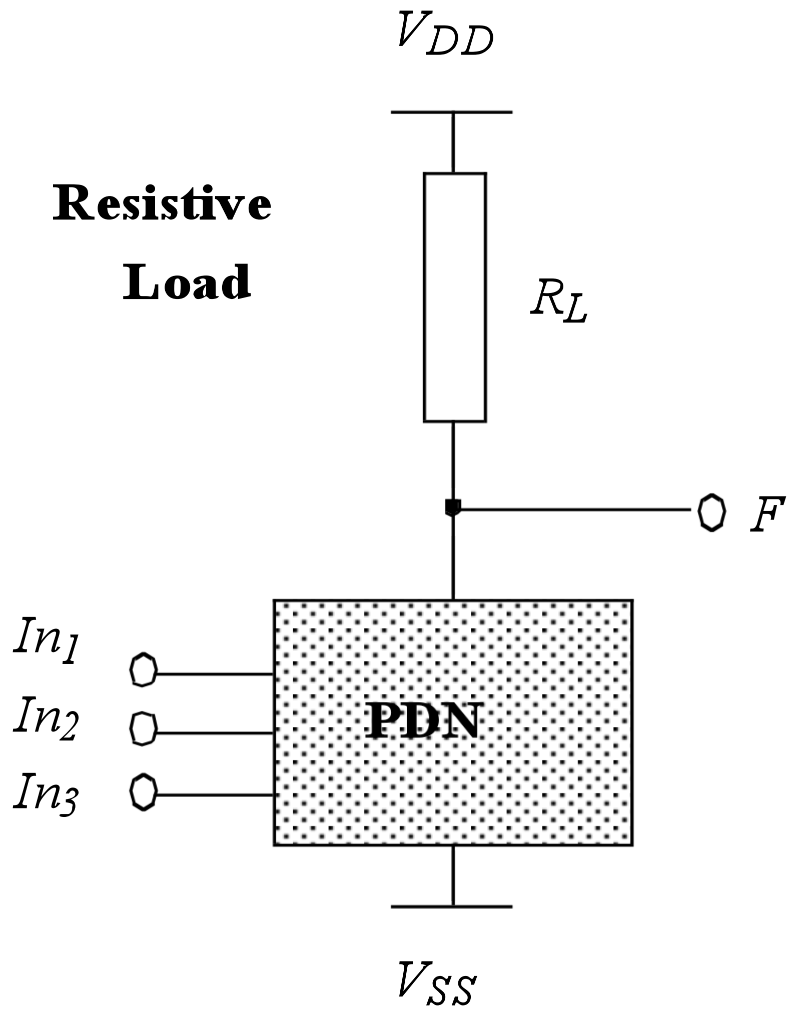
(b) depletion load NMOS



(c) pseudo-NMOS

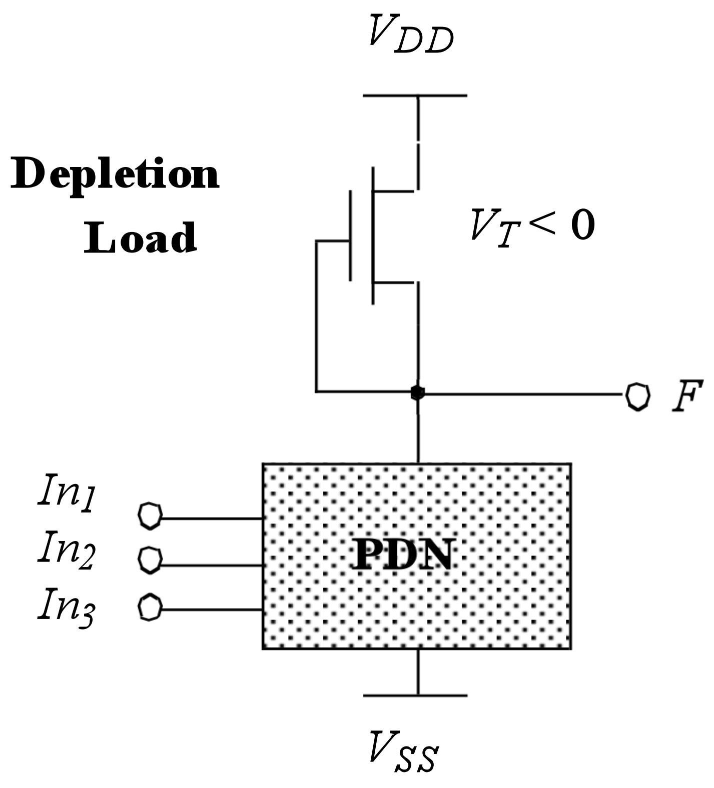
Goal: to reduce the number of devices over complementary CMOS

Ratioed Logic

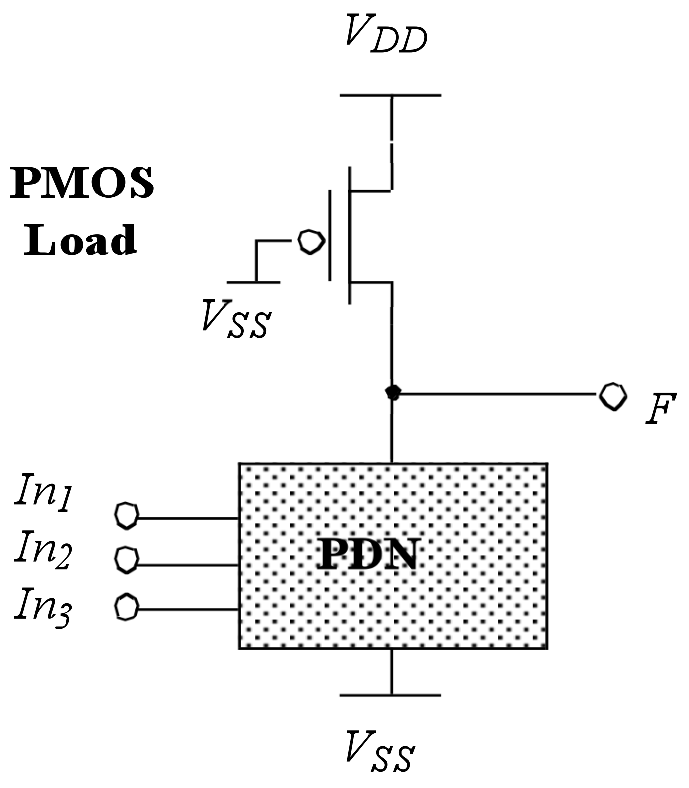


- N transistors + Load
- $V_{OH} = V_{DD}$
- $V_{OL} = \dfrac{R_{PN}}{R_{PN} + R_L}$
- Assymetrical response
- Static power consumption
- $t_{pL} = 0.69\ R_L C_L$

Active Loads



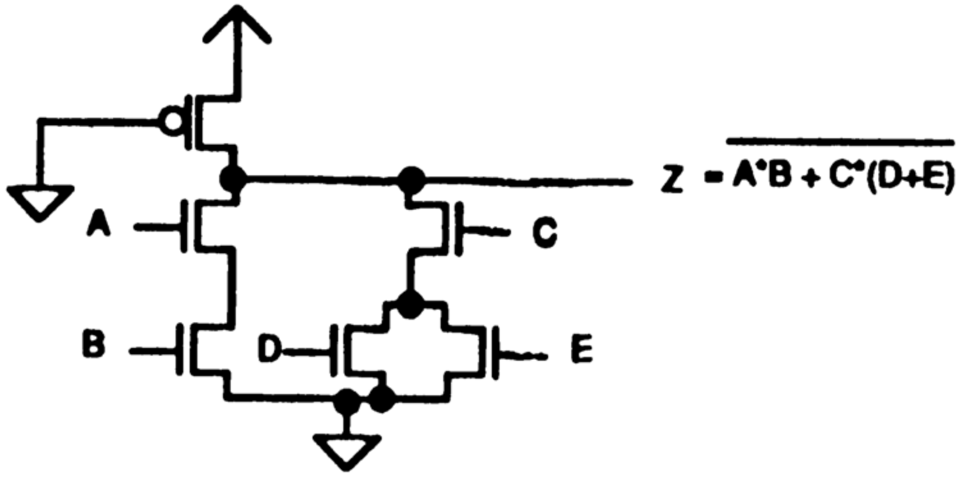
depletion load NMOS



pseudo-NMOS

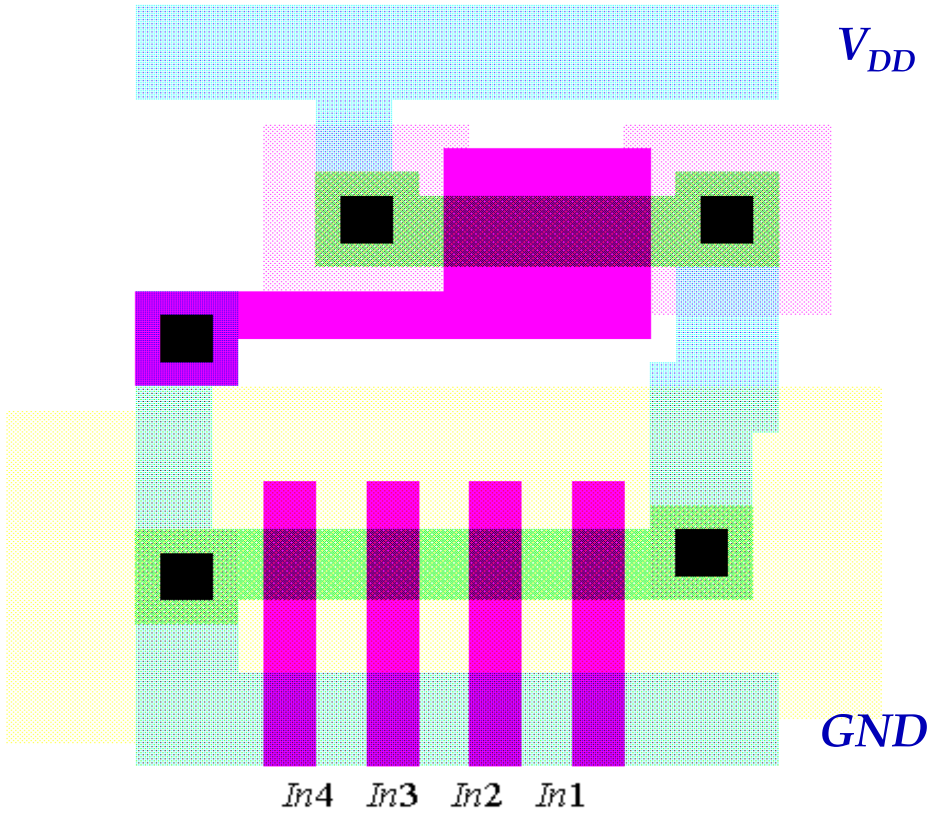
Psuedo NMOS

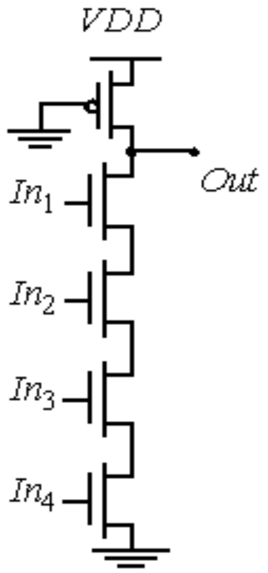
- *Disadvantages of previous circuit* :
  - Almost twice as many transistors as equivalent NMOS implementation.
  - If there are too many series transistors in the tree, switching speed is reduced.
- Try a pseudo NMOS circuit:-



- The pull-up p-channel transistor is always conducting.
  - *Disadvantages*: high d.c. dissipation & slow rise time.

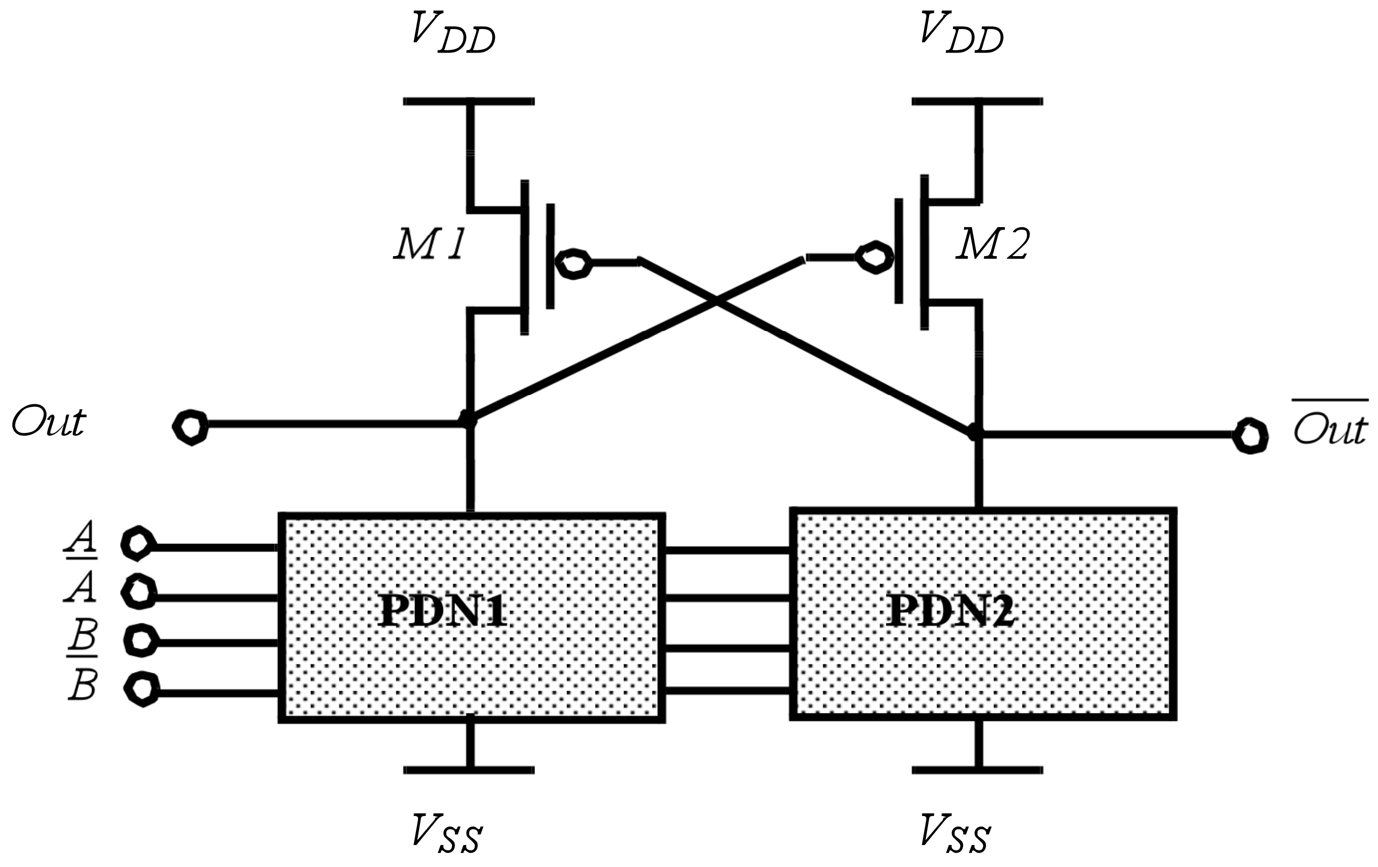
Pseudo-NMOS NAND Gate





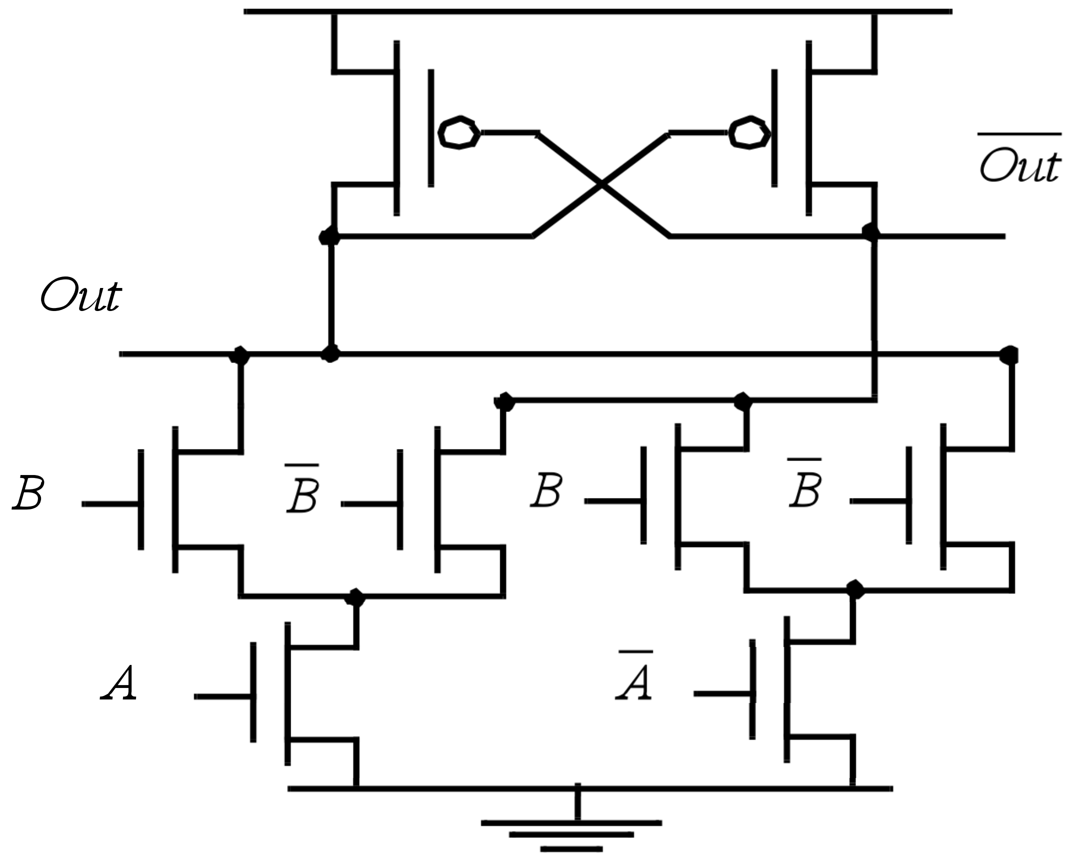
$C_{L,pseudo} = 0.5\ C_{L,CMOS}$ **(Fan-out of 1)**

Improved Loads (1)



Dual Cascode Voltage Switch Logic (DCVSL)

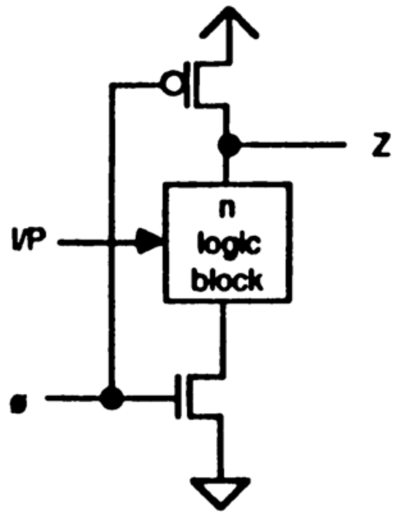
Example

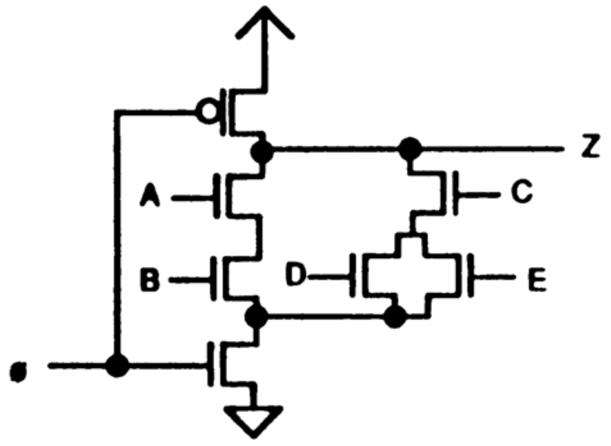


XOR-NXOR gate

Dynamic Logic

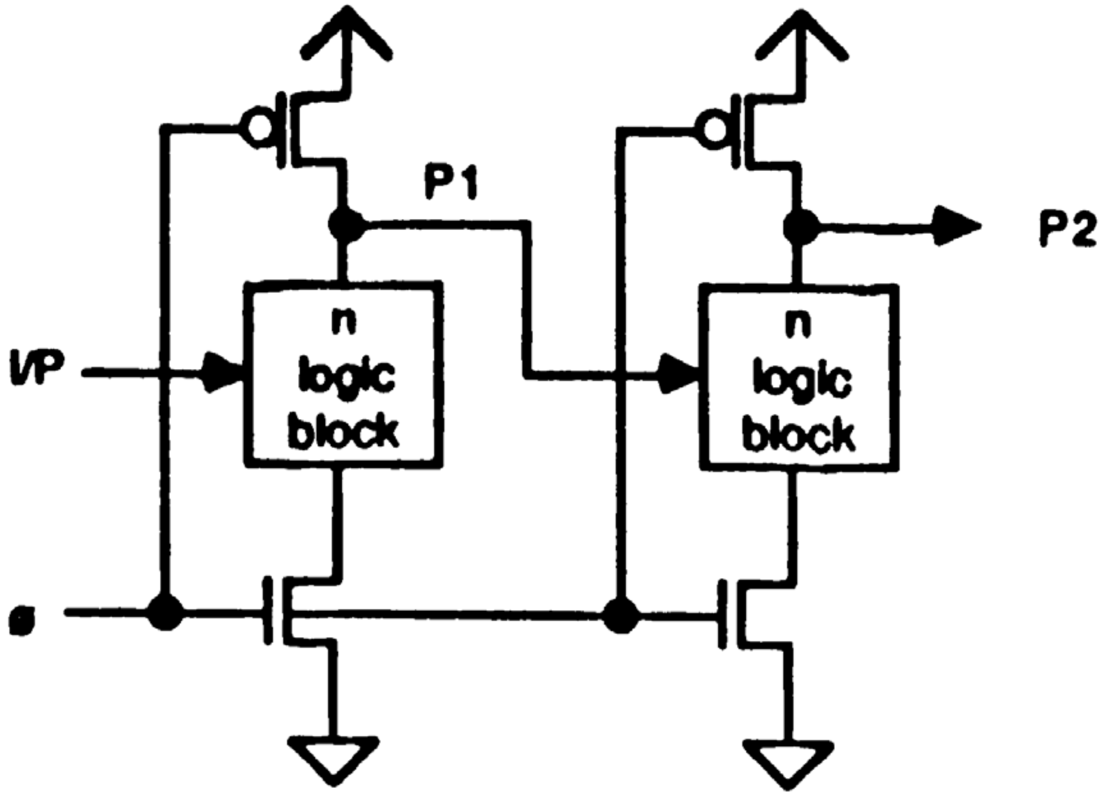
- There is another class of logic gates which relies on the use of a clock signal. This class of circuit is known as *dynamic circuits*. The clock signal is used to divide the gate operation into two halves. In the first half, the output node is ***pre-charged*** to a high or low logic state. In the second half of a clock cycle, the circuit ***evaluates*** the correct output state.
- When Ø is low, Z is charged to high. When Ø is high, n logic block evaluates input, and conditionally discharges Z. This circuit adds series resistance to the pull-down n-channel transistor, therefore the fall time is increased slightly.
- This circuit is *dynamic* because during evaluation, the output high level at Z is maintained by the stray capacitance at the output node. If Ø stays high (i.e. evaluation period) for a long time, Z may eventually discharge to a low logic level.





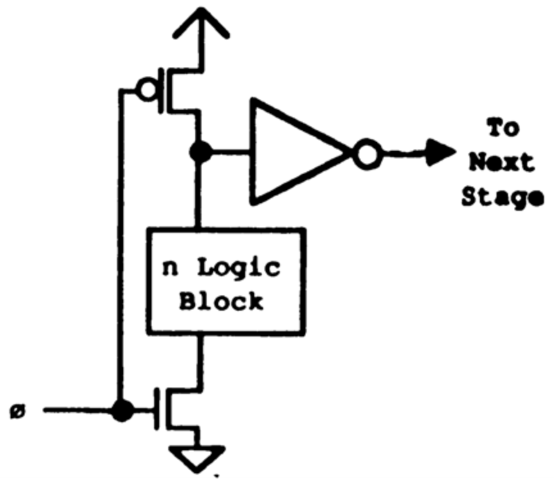
Problem with Cascading Dynamic Logic

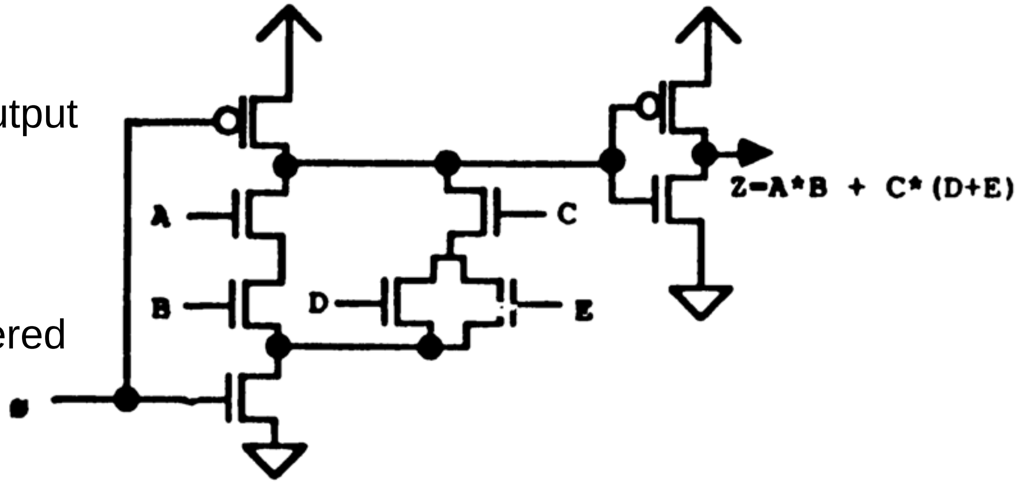
- Problem with cascading such as a circuit:-
  - Inputs can only be changed when Ø is low and must be stable when Ø is high.
  - When Ø is low, both P1 and P2 are precharged to a high voltage. However when Ø is high, delay through on the output P1 may erroneously discharge P2.



CMOS Domino Logic

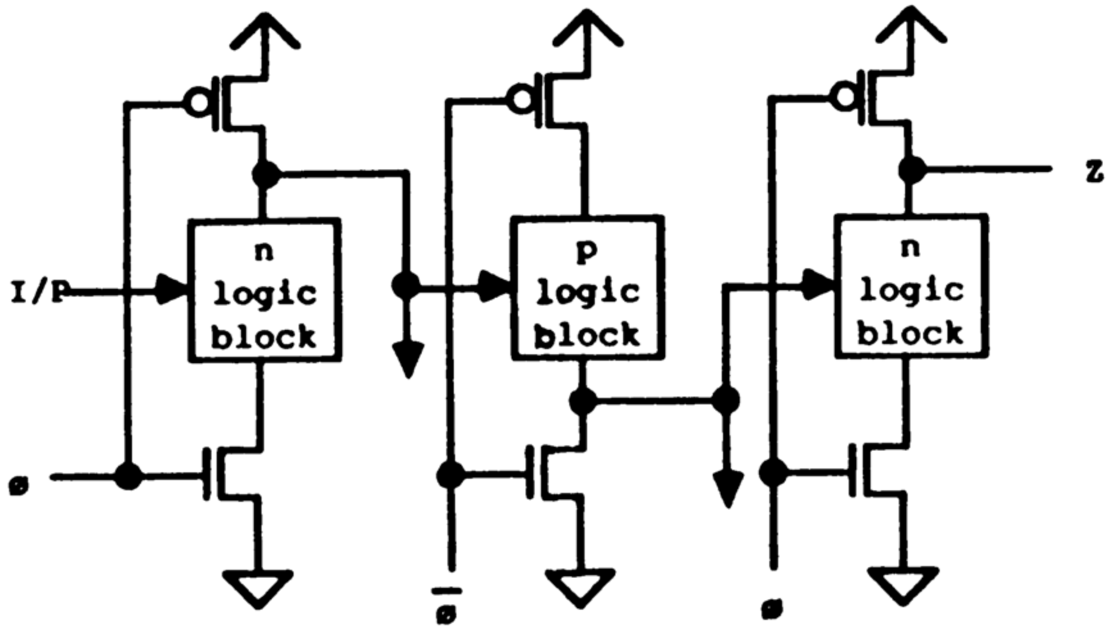
- Solution to the above problem:-
  - Add an inverter to ensure that the output is low during precharge, and prevent the next stage from evaluating, until the current stage has finished evaluation.
  - *This ensures that each stage* (at the output of the inverter) *will make at most a single transition from 0 -> 1.*
  - When many stages are cascaded, evaluation proceeds from one stage to the next - similar to dominos falling one after another.
- *Disadvantages* of domino logic:-
  - Only non-inverting logic is possible, i.e. output also high active
  - Each gate needs an inverter; hence more transistors
  - Suffer from charge sharing effect (considered later)





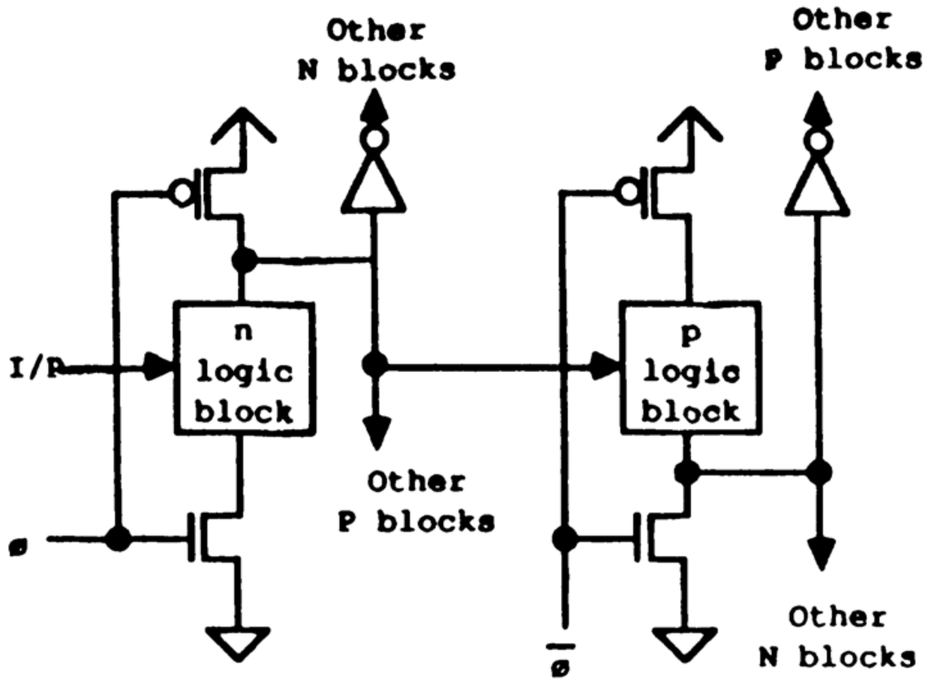
Alternating dynamic logic (1)

- Another possible scheme is to use alternate n and p logic blocks as shown below.
- In this scheme, each alternate stage is pre-charged high and low. Each stage uses alternate n and p transistors to implement the gate function. Stage 1 makes at most one high to low transition, while stage 2 makes at most one low to high transition for each evaluation. Since the p logic block will only change state if input is a low, this circuit behaves like the domino logic.



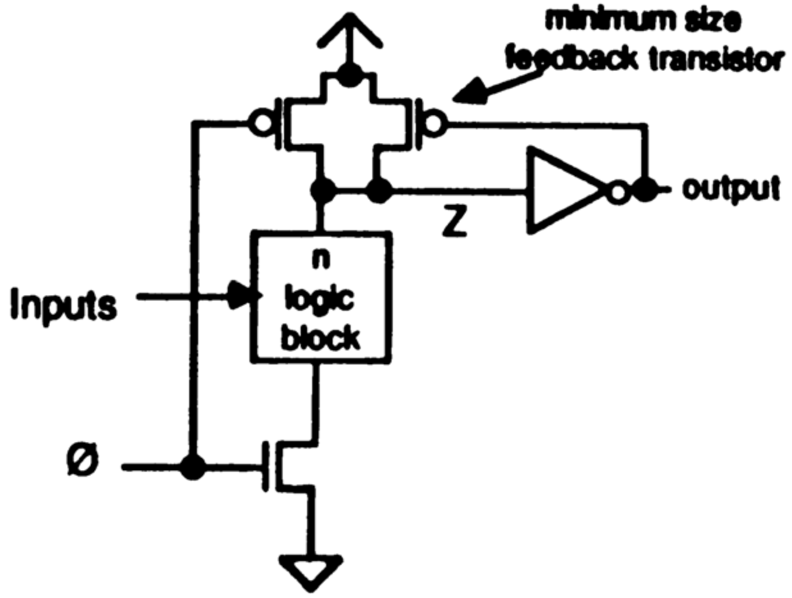
Alternating dynamic logic (2)

- A slight variation of this circuit is show below, where an inverter is added per stage to increase flexibility. Here each stage can drive either n or p blocks and both low active and high active logic can be implemented.



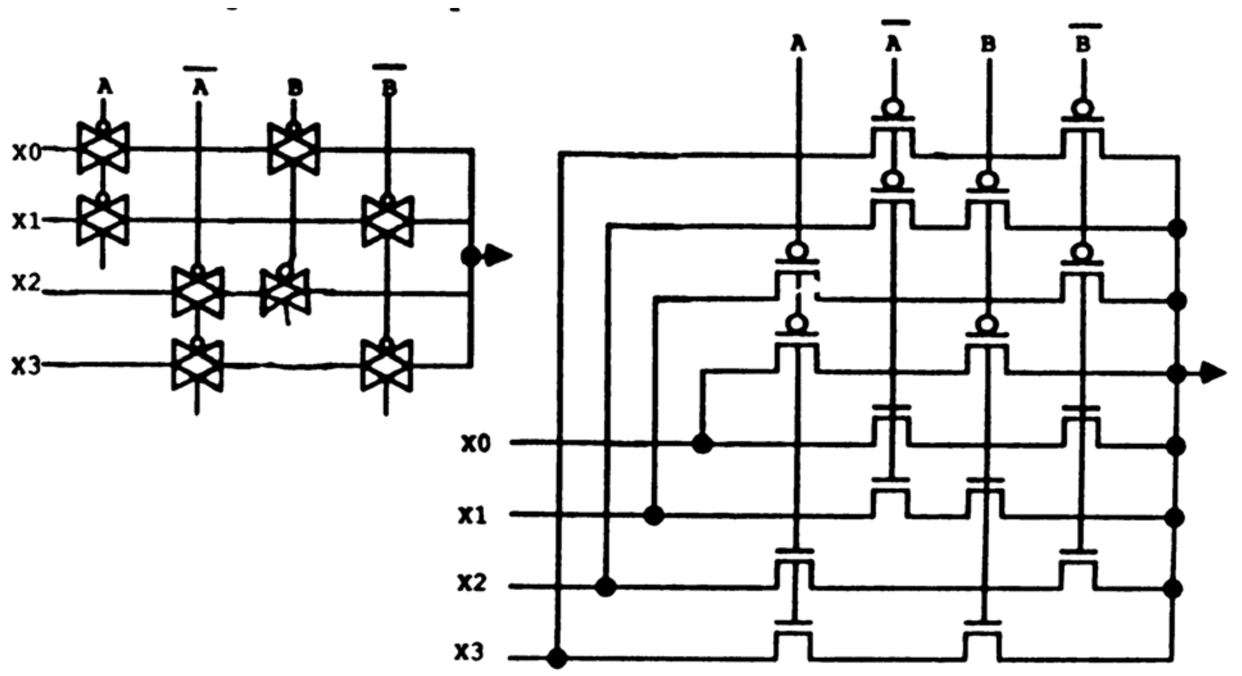
Making a Dynamic Gate static

- Finally, by adding a feedback pullup, we can make the circuit static.
- This circuit turns the originally ***dynamic*** gate into a ***static*** gate because the feedback transistor can maintain a logic high level at the node Z for an indefinite length of time. Without this feedback transistor, the charge stored at the node Z will eventually leak away.



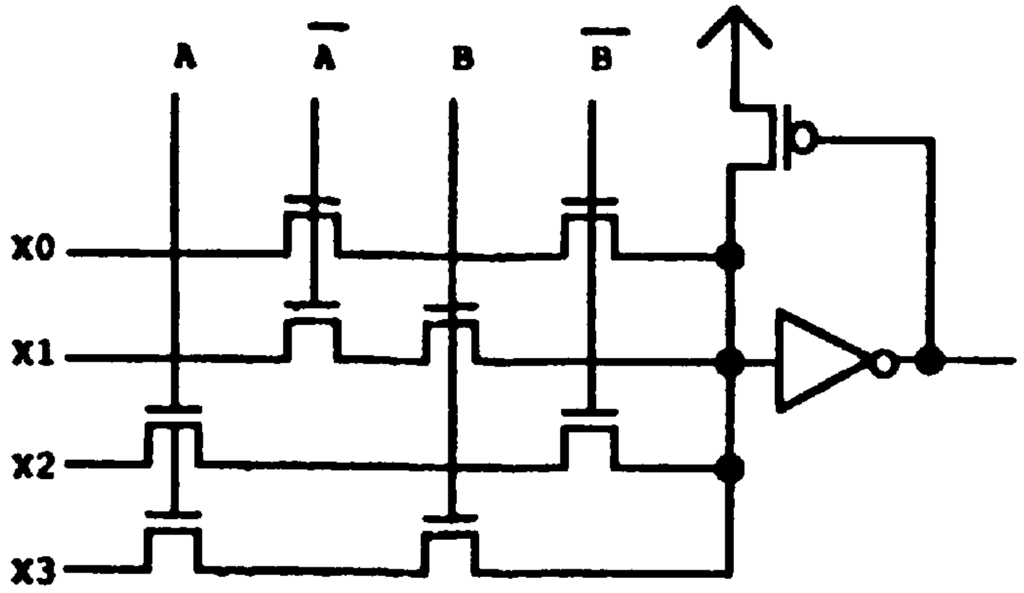
Pass Transistor Logic

- An alternative design style is to use pass transistors. The following is an example of a multiplexer.
- Complementary transmission gates are used here because n-channel pass transistors will pass 0 logic level well but, 1 logic level poorly. This is because in order for the n-transistor to be **ON**, V_{gs} must be greater than V_{th}. Therefore each series n transistor will degrade the 1 logic level by V_{th}. The opposite is true with p-channel pass transistors: 0 logic level is passed poorly.



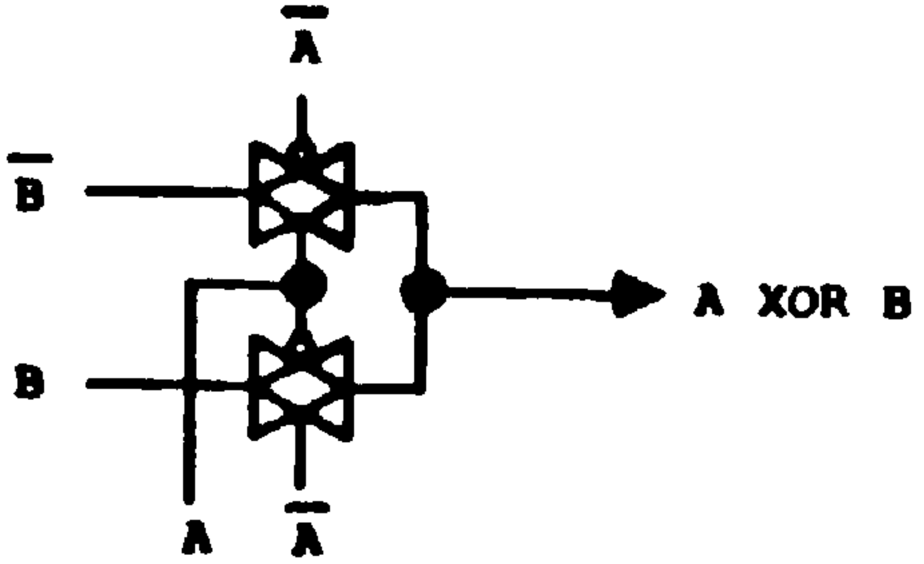
Pass Transistor Logic with feedback

- This circuit uses only n transistors, therefore it is economical on transistor count. In order to ensure that the 1 logic level is passed properly, a p pull-up transistor is added. This restores the 1 logic level at the input of the inverter.



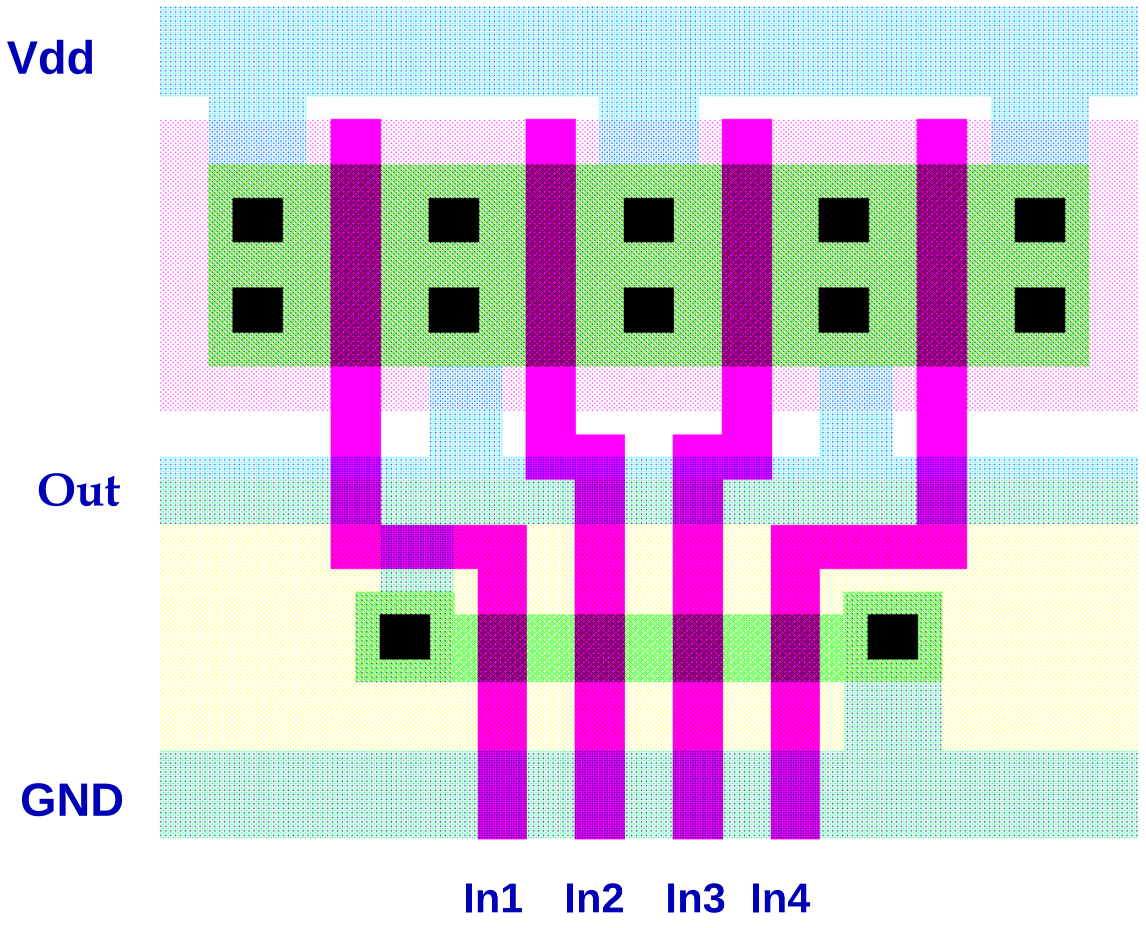
Pass Transistor XOR gate

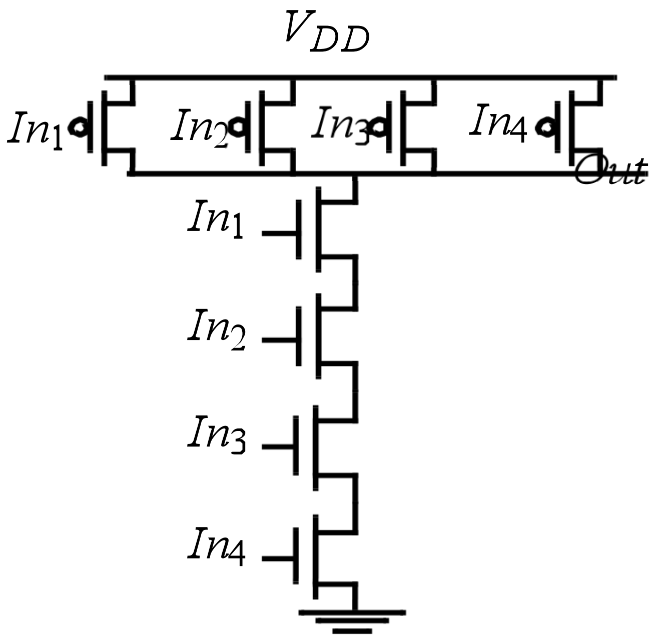
- Pass transistor logic can sometimes be very economical in implementing logic functions. For example, an **XOR** gate can be implemented with just two transmission gates:-



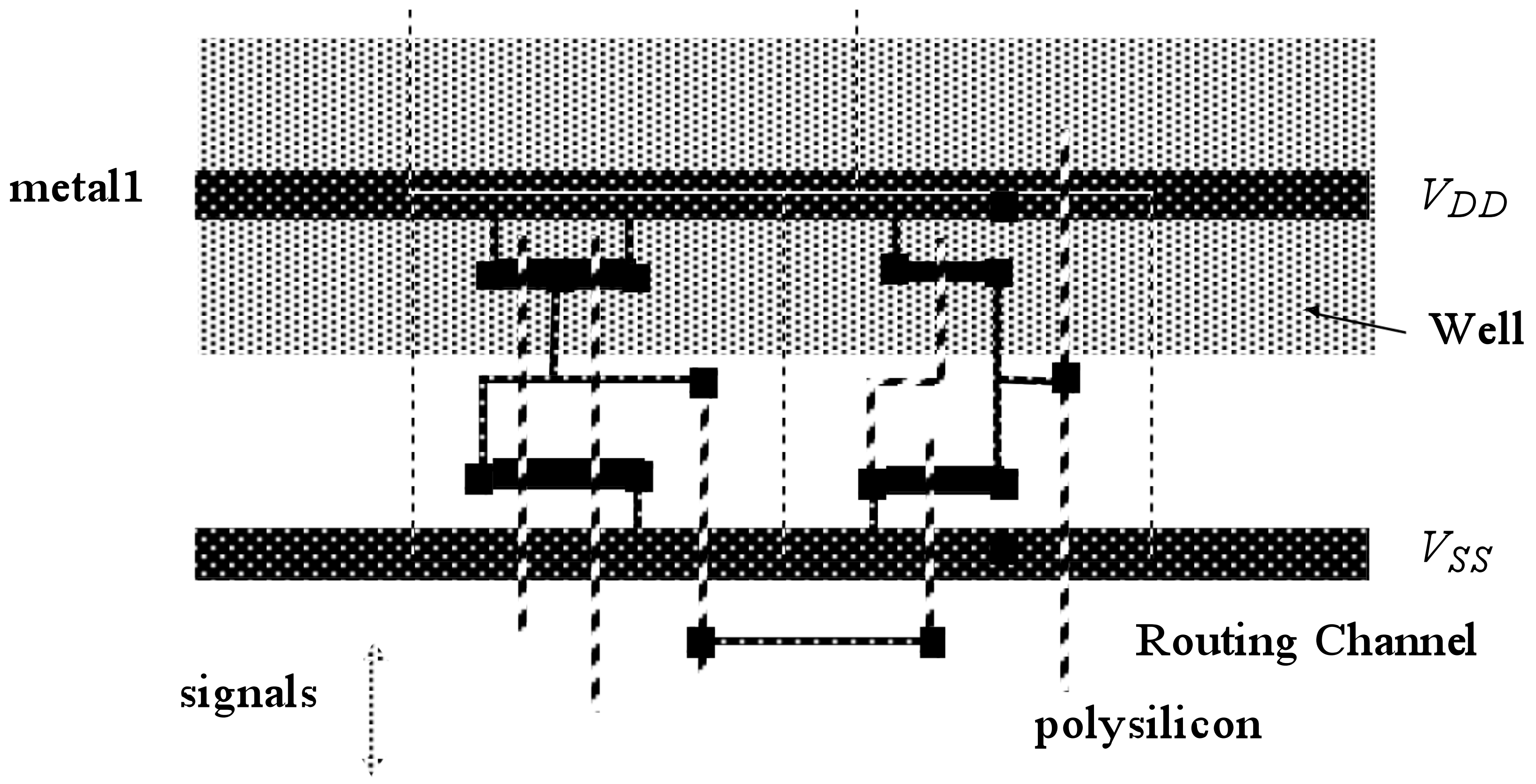
A	B	XOR
0	0	0
1	0	1
0	1	1
1	1	0

4-input NAND Gate

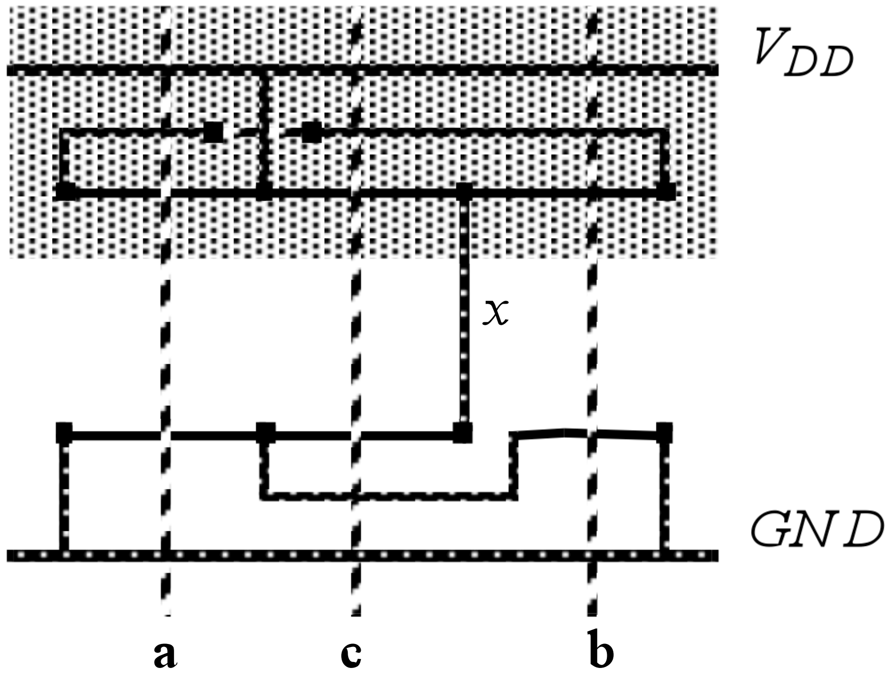




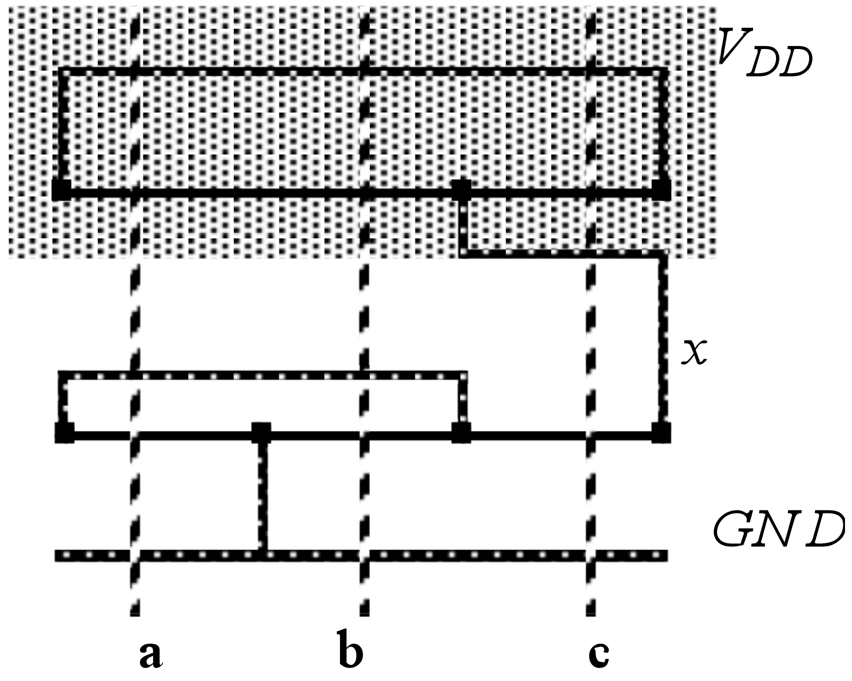
Standard Cell Layout Methodology



Two Versions of (a+b).c



(a) Input order $\{a\ c\ b\}$



(b) Input order $\{a\ b\ c\}$