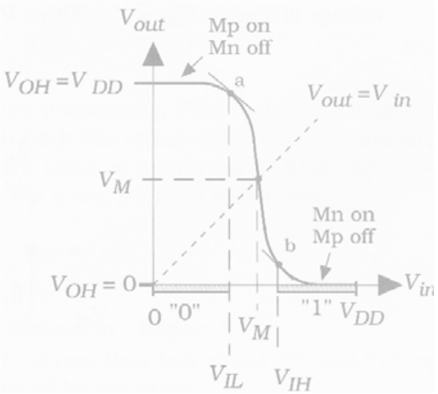
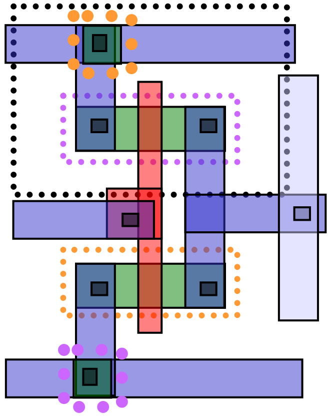
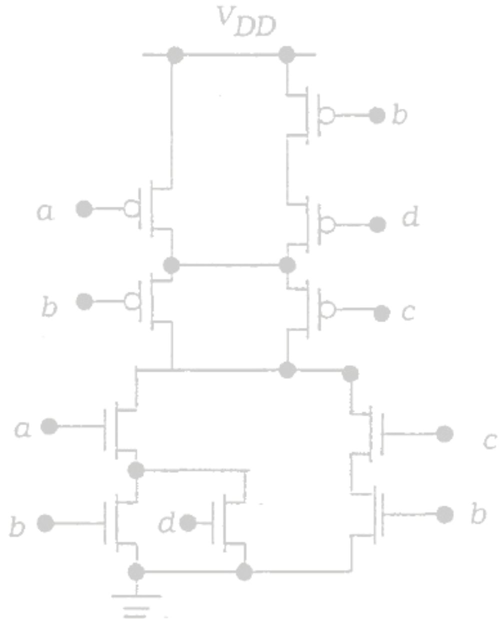
$$
\begin{aligned}
(a+b)\cdot(a+c) &= a + a\cdot b + a\cdot c + b\cdot c \\
&= a\cdot(1+b) + a\cdot c + b\cdot c \\
&= a\cdot(1+c) + b\cdot c \\
&= a + b\cdot c
\end{aligned}
$$





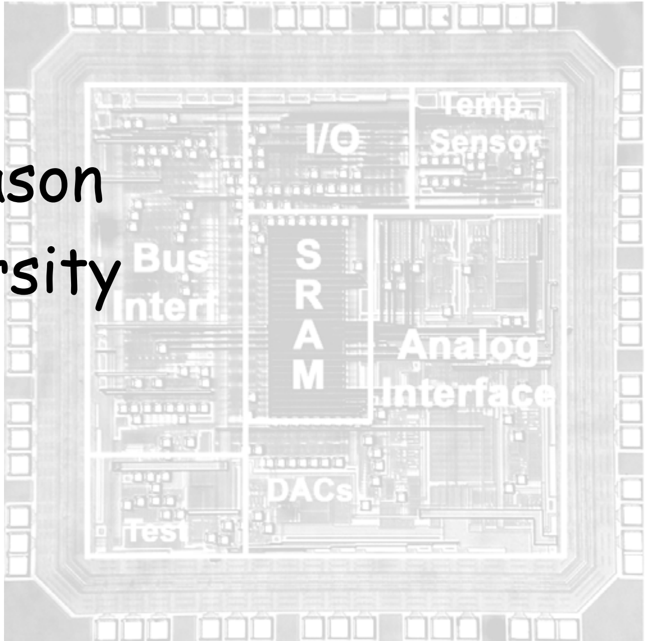
ECE 410: VLSI Design Course Lecture Notes

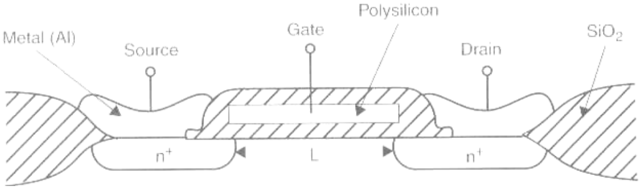
(Uyemura textbook)



Professor Andrew Mason

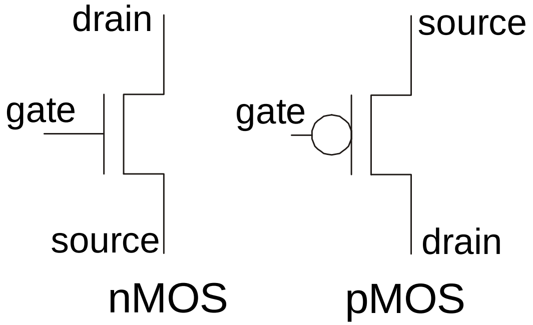
Michigan State University

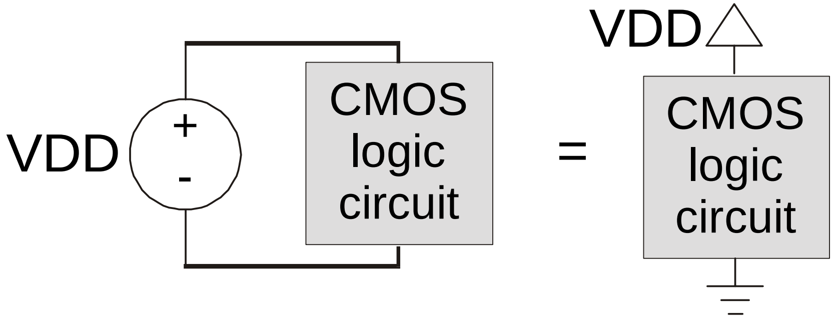


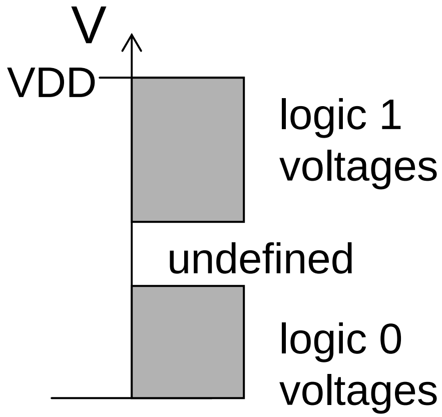


CMOS Circuit Basics

- **CMOS** = complementary MOS
  - uses 2 types of MOSFETs to create logic functions
    - nMOS
    - pMOS
- CMOS Power Supply
  - typically single power supply
  - **VDD**, with Ground reference
    - typically uses single power supply
    - VDD varies from 5V to 1V
- Logic Levels
  - all voltages between 0V and VDD
  - Logic '1' = VDD
  - Logic '0' = ground = 0V

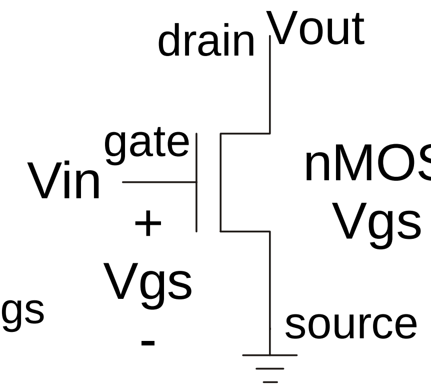






Transistor Switching Characteristics

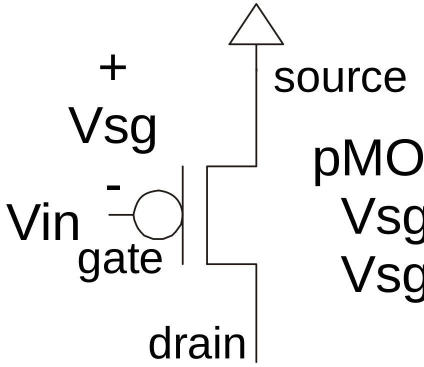
- nMOS
  - switching behavior
    - on = closed, when Vin > Vtn
      - Vtn = nMOS "threshold voltage"
      - Vin is referenced to ground, Vin = Vgs
    - off = open, when Vin < Vtn



nMOS

nMOS
Vgs > Vtn = on

- pMOS
  - switching behavior
    - on = closed, when Vin < VDD - |Vtp|
      - |Vtp| = pMOS "threshold voltage" magnitude
      - Vin is referenced to ground, Vin = VDD-Vsg
    - off = open, when Vin > VDD - |Vtp|



pMOS

pMOS
Vsg > |Vtp| = on
Vsg = VDD - Vin

Rule to Remember: 'source' is at
- lowest potential for nMOS
- highest potential for pMOS

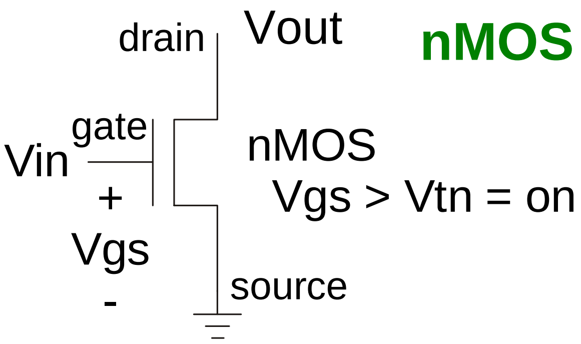
Transistor Digital Behavior

- nMOS

Vin	Vout (drain)	
1	Vs=0	device is ON
0	?	device is OFF

- pMOS

Vin	Vout (drain)	
1	?	device is OFF
0	Vs=VDD=1	device is ON



nMOS

drain

Vout

gate

Vin

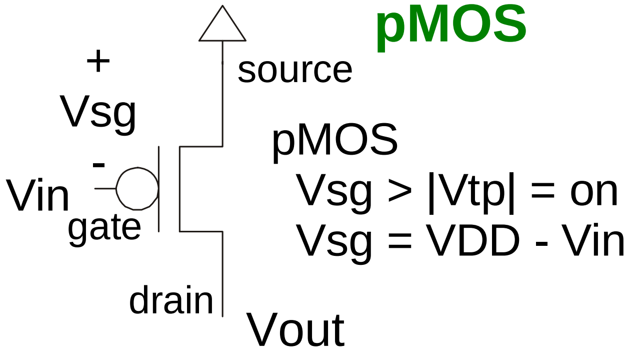
+

Vgs

-

source

nMOS
Vgs > Vtn = on



pMOS

source

+

Vsg

-

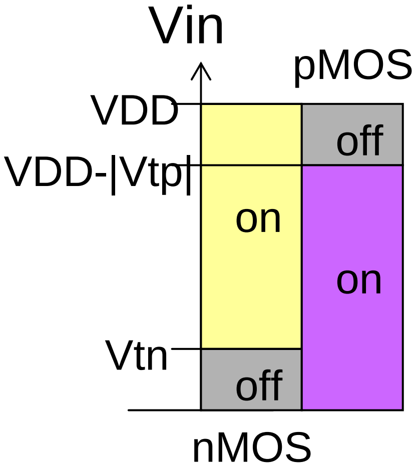
Vin

gate

drain

Vout

pMOS
Vsg > |Vtp| = on
Vsg = VDD - Vin



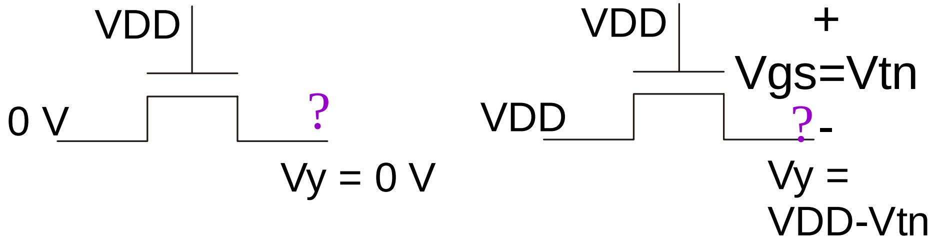
Notice:
When Vin = low, nMOS is off, pMOS is on
When Vin = high, nMOS is on, pMOS is off
→ Only one transistor is on for each digital voltage

MOSFET Pass Characteristics

- Pass characteristics: passing of voltage from drain (or source) to source (or drain) when device is ON (via gate voltage)
- Each type of transistor is better than the other at passing (to output) one digital voltage
  - nMOS passes a good low (0) but not a good high (1)
  - pMOS passes a good high (1) but not a good low (0)

nMOS

ON when gate is 'high'

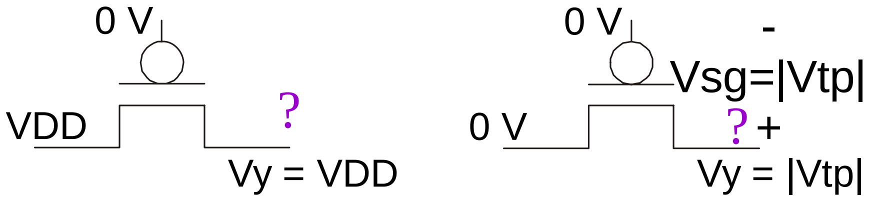


Passes a good low

Max high is VDD-Vtn

pMOS

ON when gate is 'low'



Passes a good high

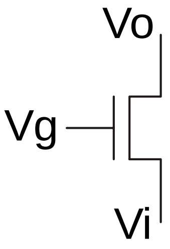
Min low is |Vtp|

Rule to Remember

'source' is at lowest potential for nMOS and at highest potential for pMOS

MOSFET Terminal Voltages

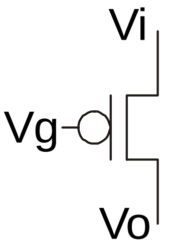
- How do you find one terminal voltage if the other 2 are known?
  - nMOS



    - case 1) if Vg > Vi + Vtn, then Vo = Vi (Vg-Vi > Vtn)
      - here Vi is the "source" so the nMOS will pass Vi to Vo
    - case 2) if Vg < Vi + Vtn, then Vo = Vg-Vtn (Vg-Vi < Vtn)
      - here Vo is the "source" so the nMOS output is limited

    For nMOS, max(Vo) = Vg-Vtn

  - pMOS



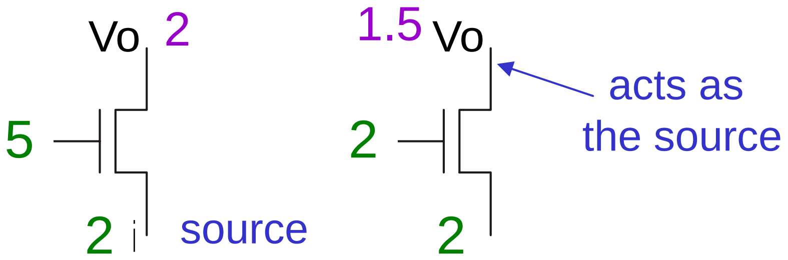
    - case 1) if Vg < Vi - |Vtp|, then Vo = Vi (Vi-Vg > |Vtp|)
      - here Vi is the "source" so the pMOS will pass Vi to Vo
    - case 2) if Vg > Vi - |Vtp|, then Vo = Vg+|Vtp| (Vi-Vg < |Vtp|)
      - here Vo is the "source" so the pMOS output is limited

    For pMOS, min(Vo) = Vg+|Vtp|

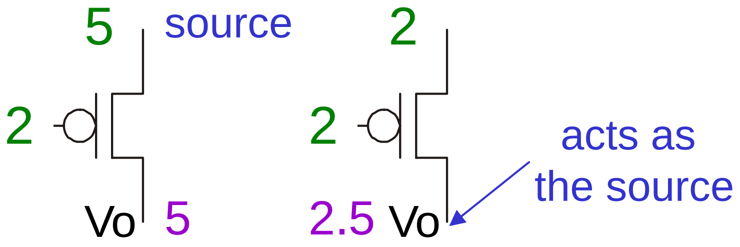
IMPORTANT:
Rules only apply if the devices is ON (e.g., Vg > Vtn for nMOS)

MOSFET Terminal Voltages: Examples

- nMOS rules max(Vo) = Vg-Vtn
  - case 1) if Vg > Vi + Vtn, then Vo = Vi (Vg-Vi > Vtn)
  - case 2) if Vg < Vi + Vtn, then Vo = Vg-Vtn (Vg-Vi < Vtn)
- nMOS examples (Vtn=0.5V)
  - 1: Vg=5V, Vi=2V
    - Vg=5 > Vi +Vtn = 2.5 ⇒ Vo = 2V
  - 2: Vg=2V, Vi=2V
    - Vg=2 < Vi+Vtn = 2.5 ⇒ Vo = 1.5V



- pMOS rules min(Vo) = Vg+|Vtp|
  - case 1) if Vg < Vi - |Vtp|, then Vo = Vi (Vi-Vg > |Vtp|)
  - case 2) if Vg > Vi - |Vtp|, then Vo = Vg+|Vtp| (Vi-Vg < |Vtp|)
- pMOS examples (Vtp=-0.5V)
  - 1: Vg=2V, Vi=5V
    - Vg=2 < Vi-|Vtp|=4.5 ⇒ Vo = 5V
  - 2: Vg=2V, Vi=2V
    - Vg=2 > Vi-|Vtp|=1.5 ⇒ Vo = 2.5V



Switch-Level Boolean Logic

- Logic gate are created by using sets of controlled switches
- Characteristics of an **assert-high** switch

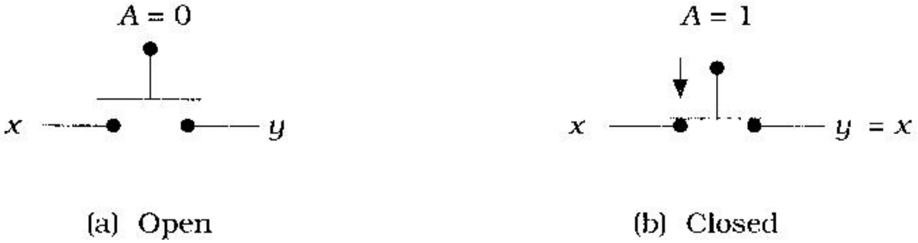


Figure 2.1 Behavior of an assert-high switch

nMOS acts like an **assert-high** switch

- $y = x \bullet A$, i.e. $y = x$ if $A = 1$

AND, or multiply function

Series switches ⇒ AND function

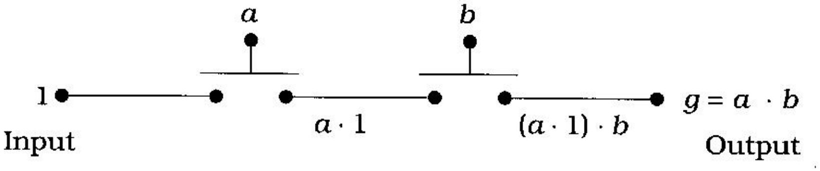


Figure 2.2 Series-connected switches

a AND b

Parallel switches ⇒ OR function

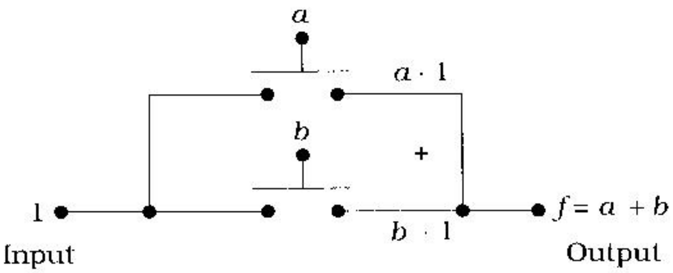
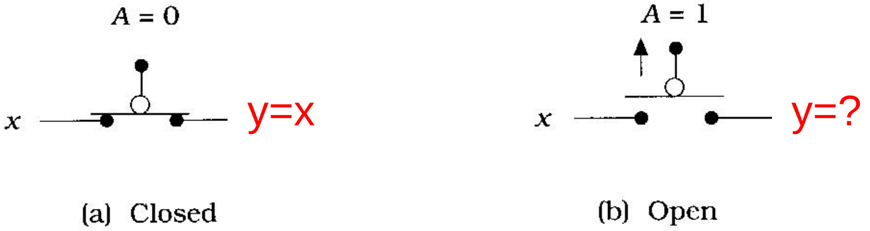


Figure 2.4 Parallel-connected switches

a OR b

Switch-Level Boolean Logic

- Characteristics of an **assert-low** switch



pMOS acts like an **assert-low** switch

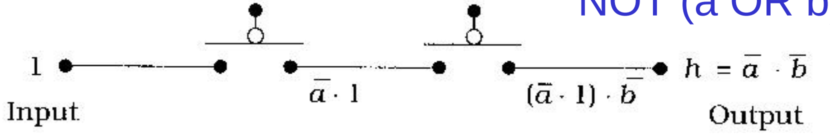
- $y = x \bullet \overline{A}$, i.e. y = x if A = 0

error in figure 2.5

Series assert-low switches ⇒ ?

a b

NOT (a OR b)



1
Input
$\bar{a} \cdot 1$
$(\bar{a} \cdot 1) \cdot \bar{b}$
$h = \bar{a} \cdot \bar{b}$
Output

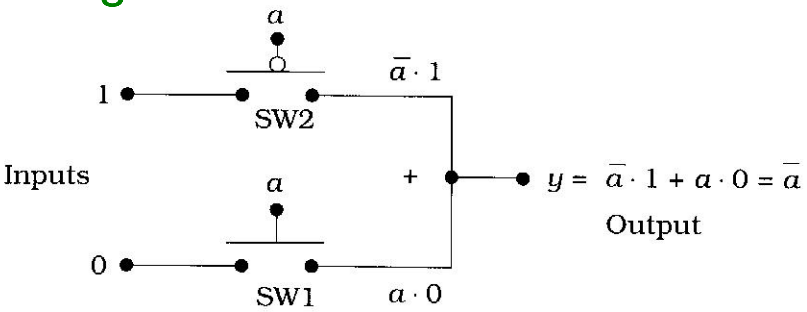
NOR

Remember This??

$$\overline{a} \bullet \overline{b} = \overline{a + b}, \quad \overline{a} + \overline{b} = \overline{a \bullet b}$$

DeMorgan relations

NOT function, combining assert-high and assert-low switches

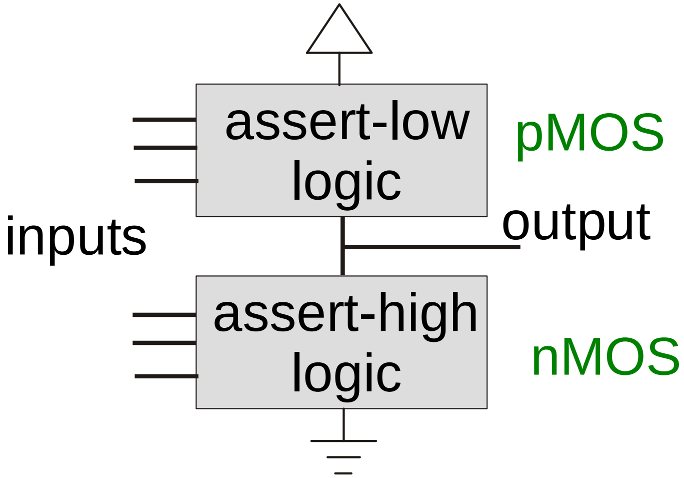


a=1 ⇒ SW1 closed, SW2 open ⇒ y=0 = $\overline{a}$

a=0 ⇒ SW1 open, SW2 closed ⇒ y=1 = $\overline{a}$

CMOS "Push-Pull" Logic

- CMOS Push-Pull Networks
  - pMOS
    - "on" when input is low
    - pushes output high
  - nMOS
    - "on" when input is high
    - pulls output low

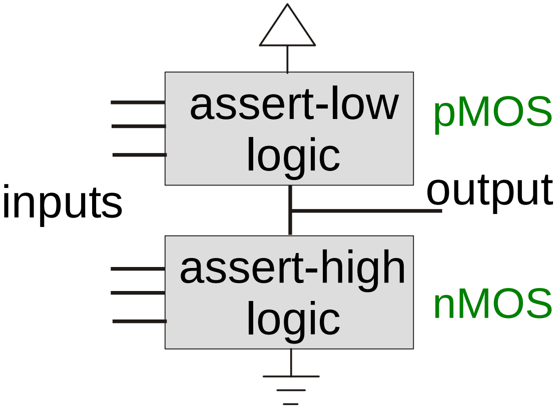


- Operation: for a given logic function
  - one logic network (p or n) produces the logic function and pushes or pulls the output
  - the other network acts as a "load" to complete the circuit, but is turned off by the logic inputs
  - since only one network it active, there is no static current (between VDD and ground)
    - zero **static** power dissipation

Creating Logic Gates in CMOS

- All standard Boolean logic functions (INV, NAND, OR, etc.) can be produced in CMOS push-pull circuits.

- Rules for constructing logic gates using CMOS
  - use a complementary nMOS/pMOS pair for each input
  - connect the output to VDD through pMOS txs
  - connect the output to ground through nMOS txs
  - insure the output is always either high or low



- CMOS produces "inverting" logic
  - CMOS gates are based on the inverter
  - outputs are always inverted logic functions

    e.g., NOR, NAND rather than OR, AND

- Logic Properties

DeMorgan's Rules

$(a \cdot b)' = a' + b'$

$(a + b)' = a' \cdot b'$

Useful Logic Properties

$1 + x = 1 \quad 0 + x = x$

$1 \cdot x = x \quad 0 \cdot x = 0$

$x + x' = 1 \quad x \cdot x' = 0$

$a \cdot a = a \quad a + a = a$

$ab + ac = a(b+c)$

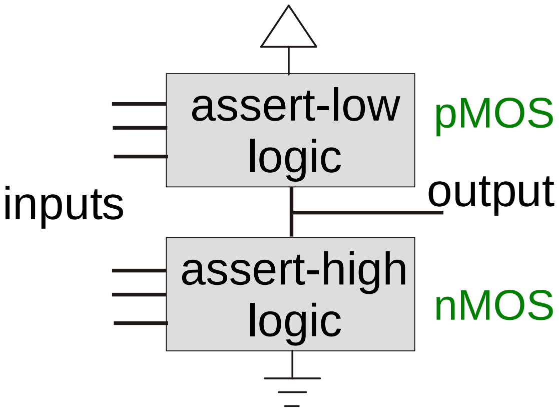
Properties which can be proven

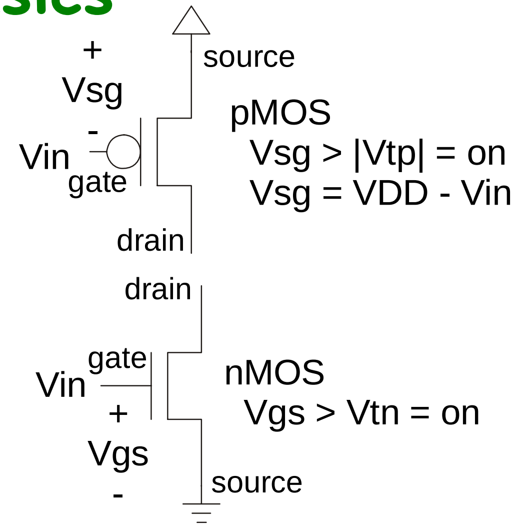
$(a+b)(a+c) = a+bc$

$a + a'b = a + b$

Review: Basic Transistor Operation

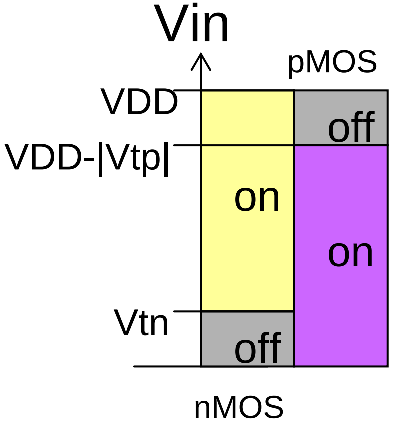
CMOS Circuit Basics





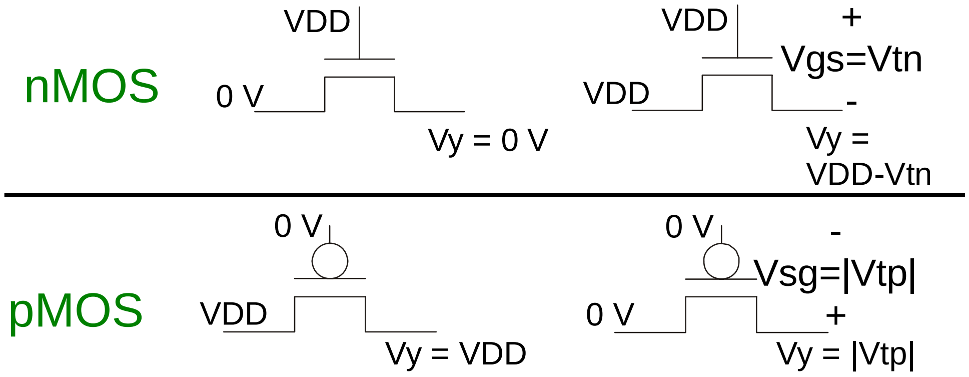
Vg= Vin	Vout
0	1 on = closed
1	? off = open

Vg= Vin	Vout
0	? off = open
1	0 on = closed



CMOS Pass Characteristics

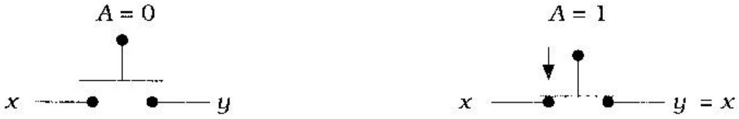
'source' is at lowest potential (nMOS) and highest potential (pMOS)



- nMOS
  - 0 in → 0 out
  - VDD in → VDD-Vtn out
  - strong '0', weak '1'
- pMOS
  - VDD in → VDD out
  - 0 in → |Vtp| out
  - strong '1', weak '0'

Review: Switch-Level Boolean Logic

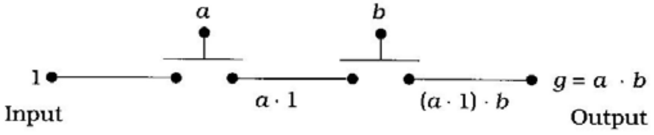
- **assert-high** switch
  - y = x • A, i.e. y = x if A = 1
  - series = AND
  - parallel = OR



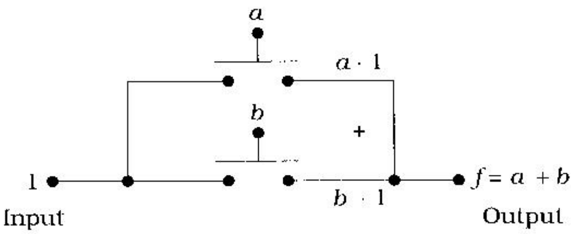
(a) Open

(b) Closed

ı assert-high switch

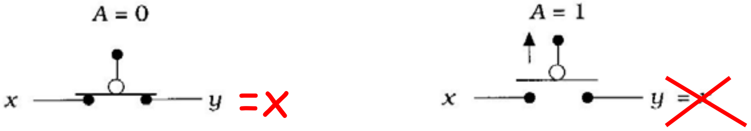


a AND b



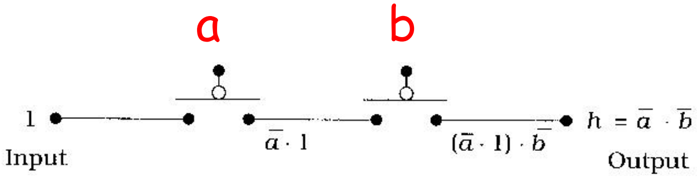
a OR b

- **assert-low** switch
  - y = x • $\overline{A}$, i.e. y = x if A = 0
  - series = NOR
  - parallel = NAND



(a) Closed

(b) Open

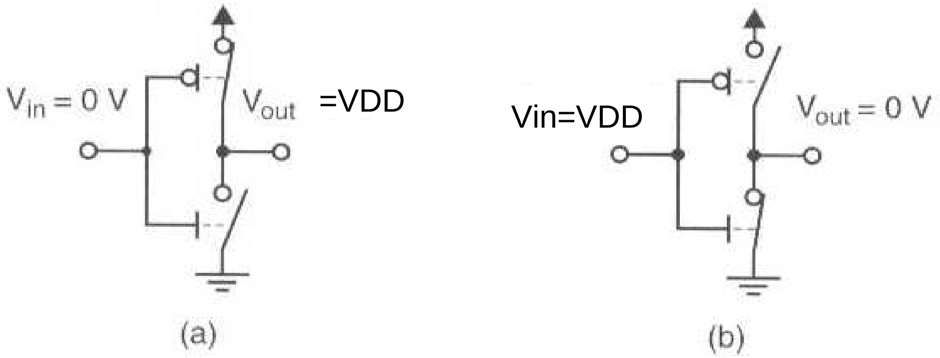

a
b
1
Input
$\bar{a} \cdot 1$
$(\bar{a} \cdot 1) \cdot \bar{b}$
$h = \bar{a} \cdot \bar{b}$
Output

NOT (a OR b)

CMOS Inverter

- Inverter Function
  - toggle binary logic of a signal

- Inverter Switch Operation



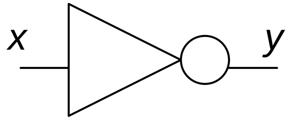
input low → output high
nMOS off/open
pMOS on/closed

pMOS "on"
→ output high (1)

input high → output low
nMOS on/closed
pMOS off/open

nMOS "on"
→ output low (0)

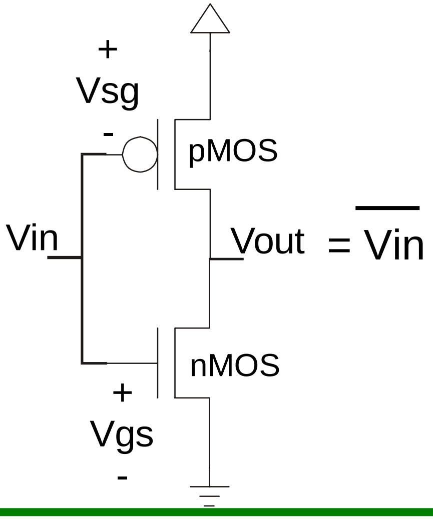
- Inverter Symbol



- Inverter Truth Table

x	$y = \overline{x}$
0	1
1	0

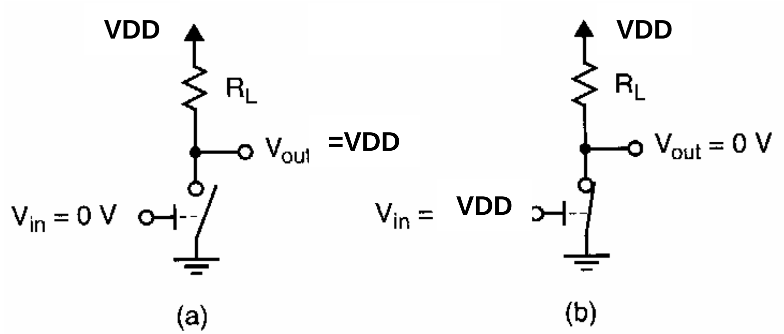
- CMOS Inverter Schematic



nMOS Logic Gates

- We will look at nMOS logic first, more simple than CMOS
- nMOS Logic (no pMOS transistors)
  - assume a resistive load to VDD
  - nMOS switches pull output low based on inputs

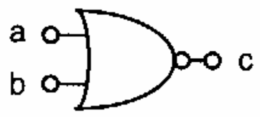
nMOS Inverter



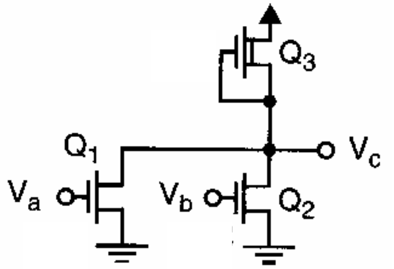
(a) nMOS is off
→ output is high (1)

(b) nMOS is on
→ output is low (0)

nMOS NOR

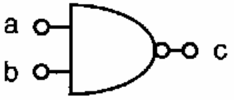


$c = \overline{a+b}$

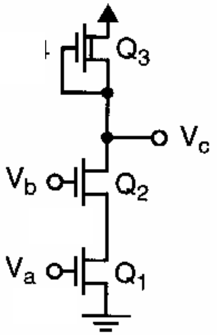


- parallel switches = OR function
- nMOS pulls low (NOTs the output)

nMOS NAND



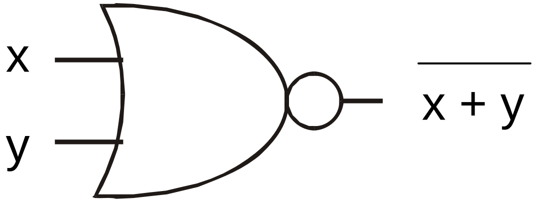
$c = \overline{ab}$



- series switches = AND function
- nMOS pulls low (NOTs the output)

CMOS NOR Gate

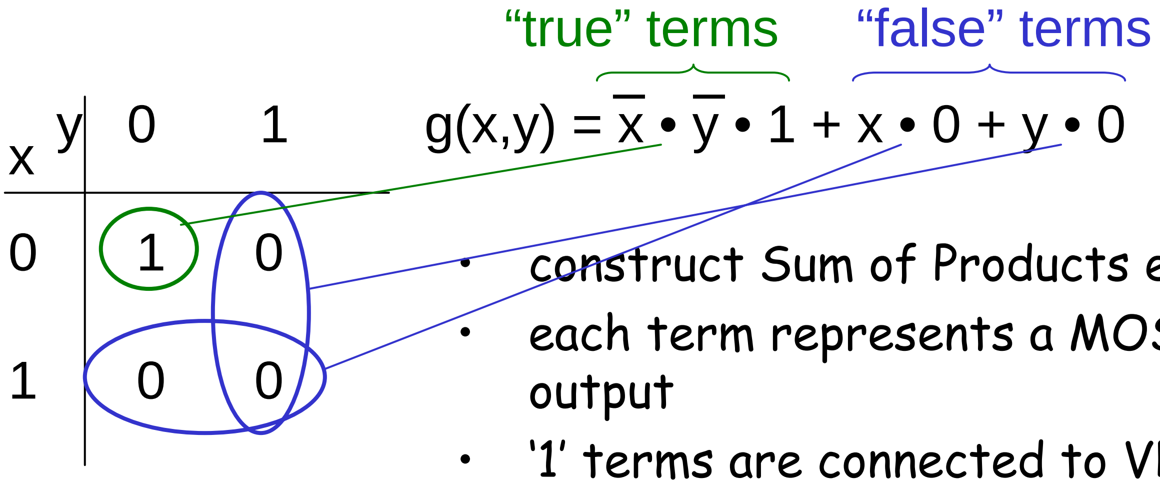
- NOR Symbol



- NOR Truth Table

x	y	$\overline{x+y}$
0	0	1
0	1	0
1	0	0
1	1	0

- Karnaugh map



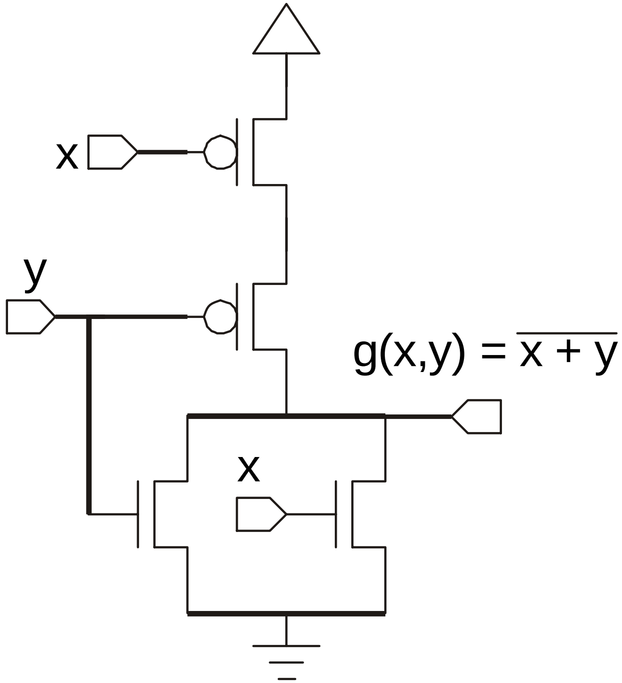
$$g(x,y) = \overbrace{\overline{x} \bullet \overline{y} \bullet 1}^{\text{“true” terms}} + \overbrace{x \bullet 0 + y \bullet 0}^{\text{“false” terms}}$$

x \ y	0	1
0	1	0
1	0	0

- construct Sum of Products equation with all terms
- each term represents a MOSFET path to the output
- '1' terms are connected to VDD via pMOS
- '0' terms are connected to ground via nMOS

CMOS NOR Gate

- CMOS NOR Schematic



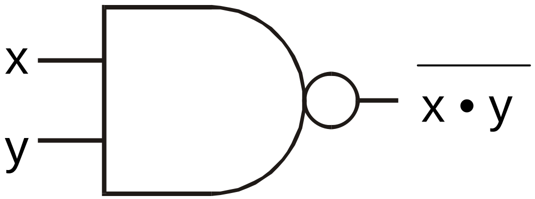
$g(x,y) = \overline{x} \bullet \overline{y} \bullet 1 + x \bullet 0 + y \bullet 0$

- **output is LOW if *x* OR *y* is true**
  - **parallel nMOS**
- **output is HIGH when *x* AND *y* are false**
  - **series pMOS**

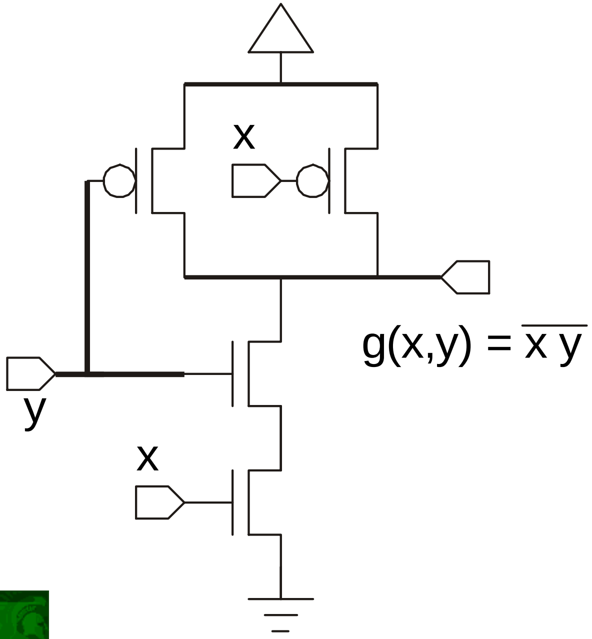
- Notice: series-parallel arrangement
  - when nMOS in series, pMOS in parallel, and visa versa
  - true for all *static CMOS* logic gates
  - allows us to construct more complex logic functions

CMOS NAND Gate

- NAND Symbol



- CMOS Schematic



- Truth Table

x	*y*	$\overline{x \bullet y}$
0	0	1
0	1	1
1	0	1
1	1	0

- K-map

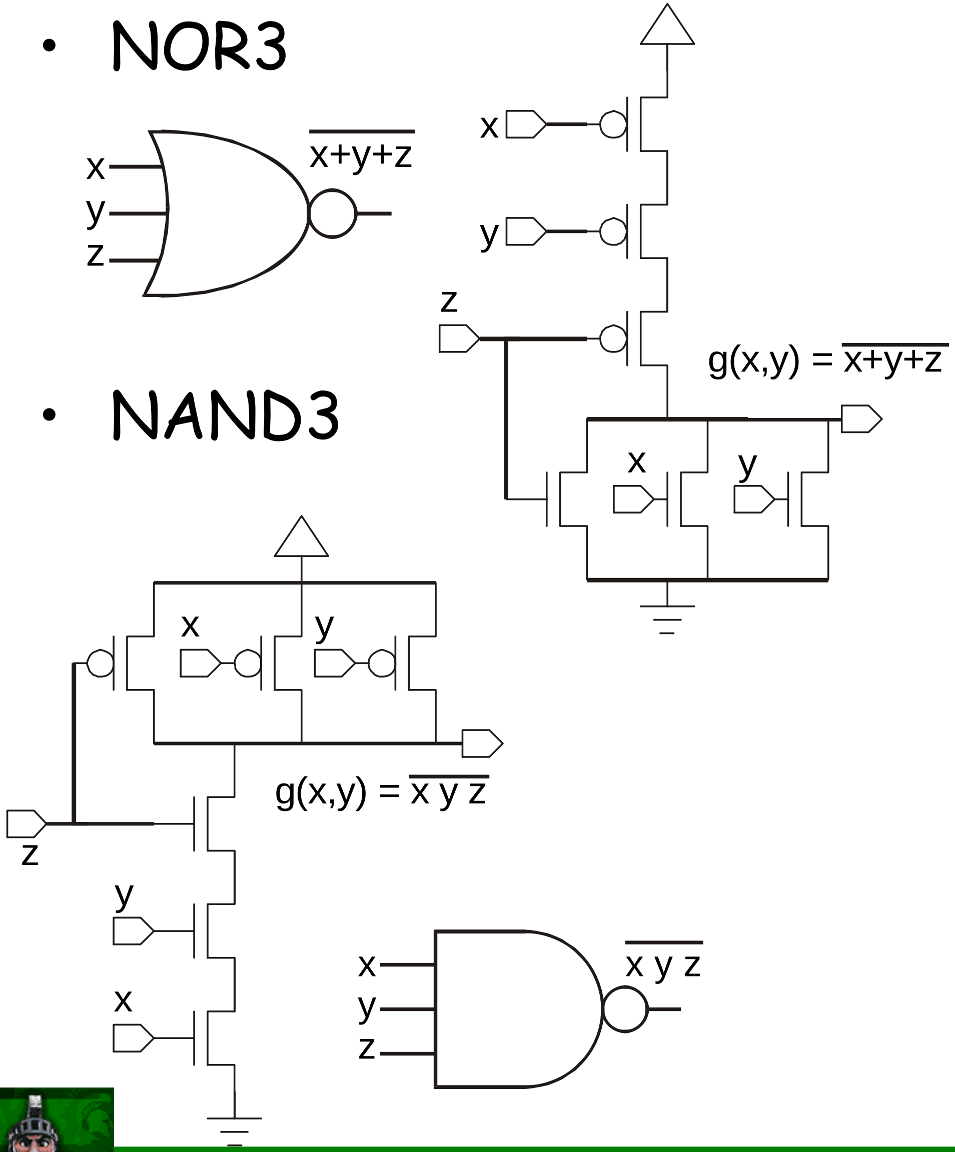
x \ y	0	1
0	1	1
1	1	0

$$g(x,y) = (\overline{y} \bullet 1) + (\overline{x} \bullet 1) + (x \bullet y \bullet 0)$$

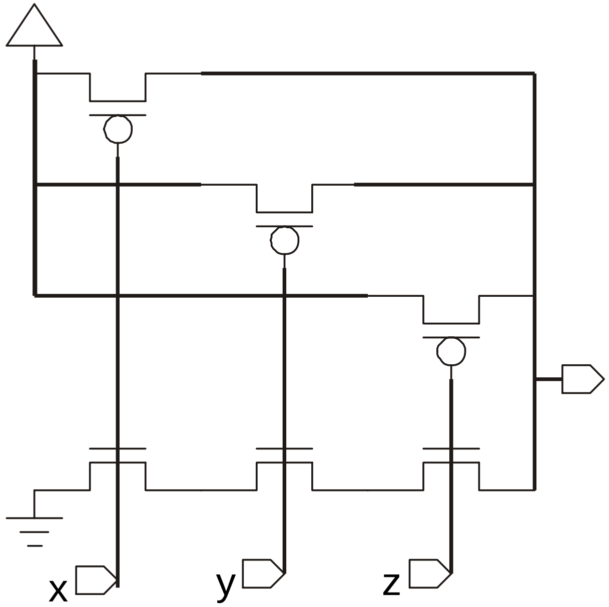
- **output is LOW if *x* AND *y* are true**
  - **series nMOS**
- **output is HIGH when *x* OR *y* is false**
  - **parallel pMOS**

3-Input Gates

- NOR3
- NAND3



- Alternate Schematic
  - what function?



- note shared gate inputs
  - is input order important?
  - in series, parallel, both?
- this schematic resembles how the circuit will look in *physical layout*

Complex Combinational Logic

- General logic functions
  - for example

$$f = \overline{a \cdot (b + c)}, \qquad f = \overline{(d \cdot e)} + a \cdot (\overline{b} + c)$$

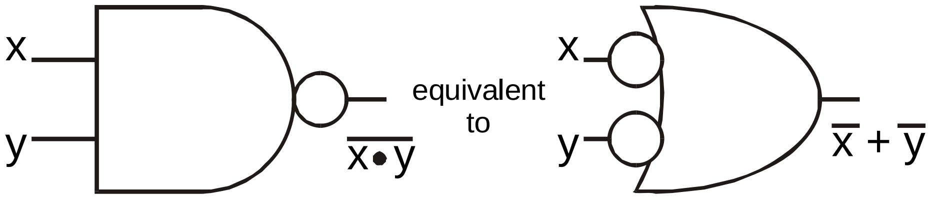
- How do we construct the CMOS gate?
  - use DeMorgan principles to modify expression
    - construct nMOS and pMOS networks

$$\boxed{\overline{a \cdot b} = \overline{a} + \overline{b}} \qquad \boxed{\overline{a + b} = \overline{a} \cdot \overline{b}}$$

  - use Structured Logic (covered only briefly in ECE410)
    - AOI (AND OR INV)
    - OAI (OR AND INV)

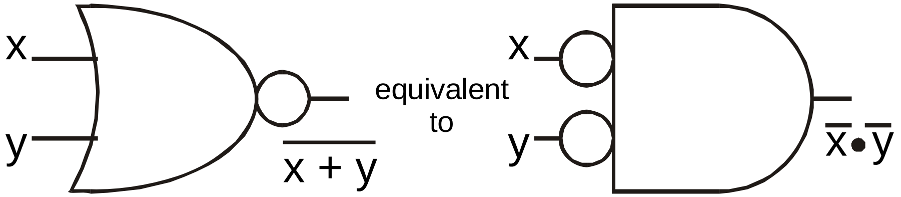
Using DeMorgan

- DeMorgan Relations
  - NAND-OR rule $\overline{a \bullet b} = \overline{a} + \overline{b}$
    - bubble pushing illustration



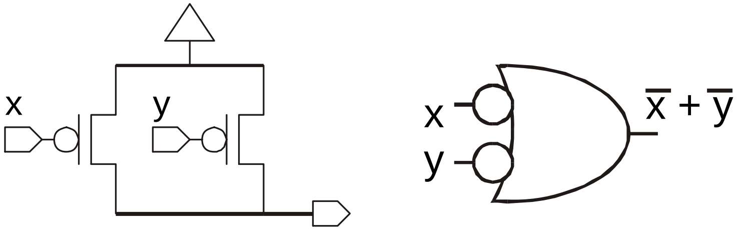
- bubbles = inversions

  - NOR-AND rule $\overline{a + b} = \overline{a} \bullet \overline{b}$



to implement pMOS this way, **must** push all bubbles to the inputs and remove all NAND/NOR output bubbles

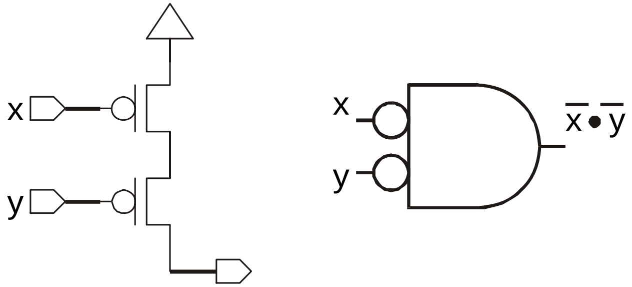
- pMOS and bubble pushing
  - Parallel-connected pMOS



$g(x,y) = \overline{x} + \overline{y} = \overline{x\,y}$

- assert-low OR
- creates NAND function

  - Series-connected pMOS

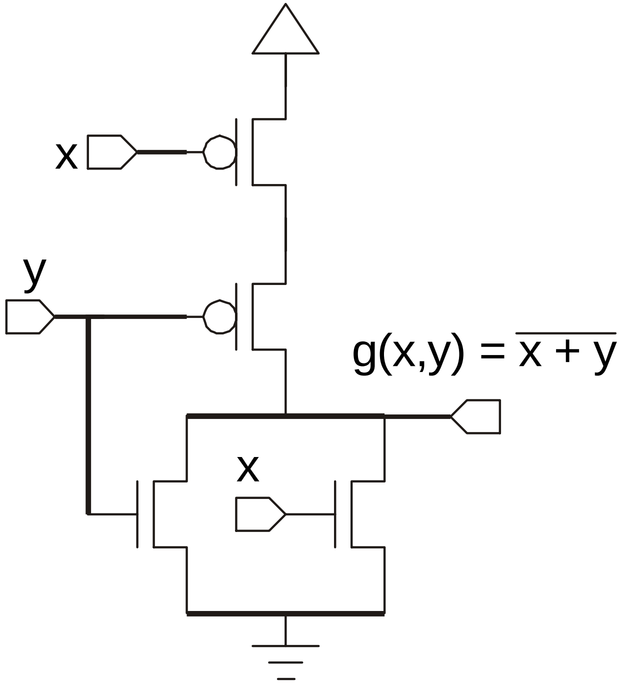


$g(x,y) = \overline{x}\ \overline{y} = \overline{x + y}$

- assert-low AND
- creates NOR function

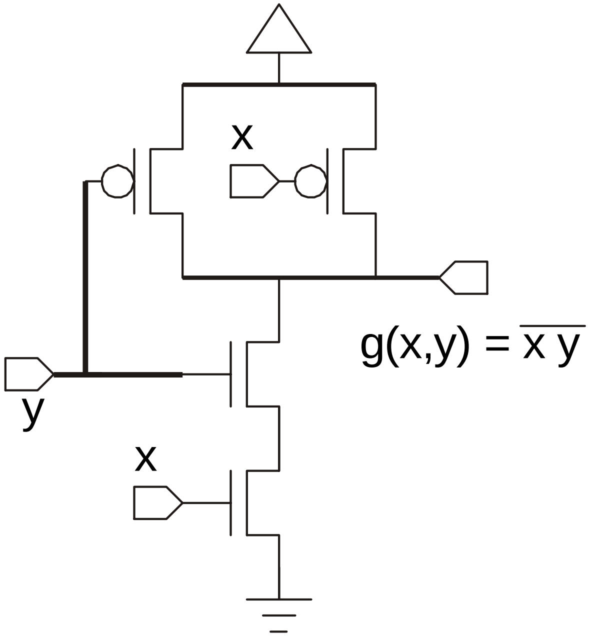
Review: CMOS NAND/NOR Gates

- NOR Schematic



- output is LOW if *x* OR *y* is true
  - parallel nMOS
- output is HIGH when *x* AND *y* are false
  - series pMOS

- NAND Schematic



- output is LOW if *x* AND *y* are true
  - series nMOS
- output is HIGH when *x* OR *y* is false
  - parallel pMOS

Rules for Constructing CMOS Gates

The Mathematical Method

- Given a logic function

  $F = f(a, b, c)$

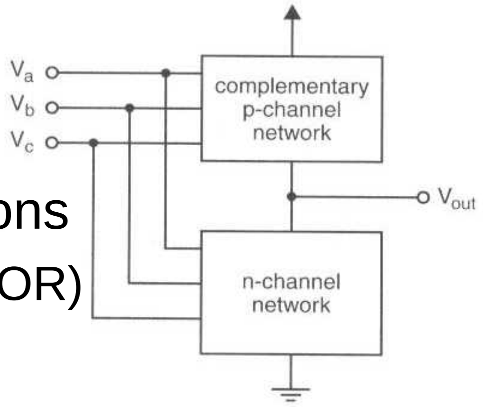
- Reduce (using DeMorgan) to eliminate inverted operations
  - inverted variables are OK, but not operations (NAND, NOR)
- Form pMOS network by complementing the **inputs**

  $Fp = f(\overline{a}, \overline{b}, \overline{c})$

- Form the nMOS network by complementing the **output**

  $Fn = \overline{f(a, b, c)} = \overline{F}$

- Construct Fn and Fp using AND/OR series/parallel MOSFET structures
  - series = AND, parallel = OR

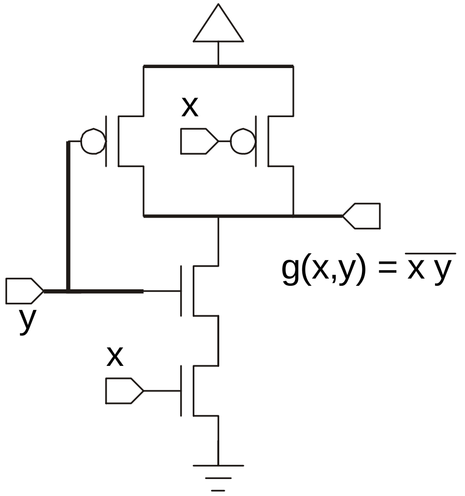


EXAMPLE:

$F = \overline{ab} \Rightarrow$

$Fp = \overline{\overline{a}\ \overline{b}} = a+b;$ OR/parallel

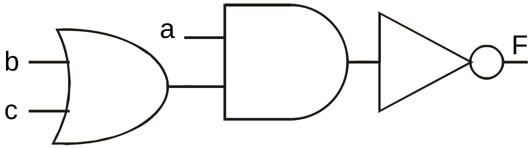
$Fn = \overline{\overline{ab}} = ab;$ AND/series



CMOS Combinational Logic Example

- Construct a CMOS logic gate to implement the function:

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



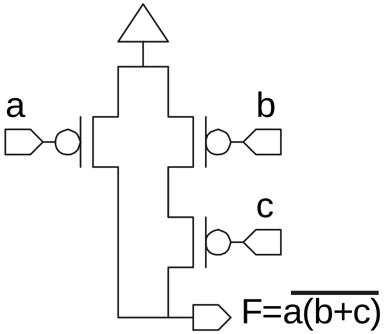
14 transistors (cascaded gates)

- pMOS
  - Apply DeMorgan expansions

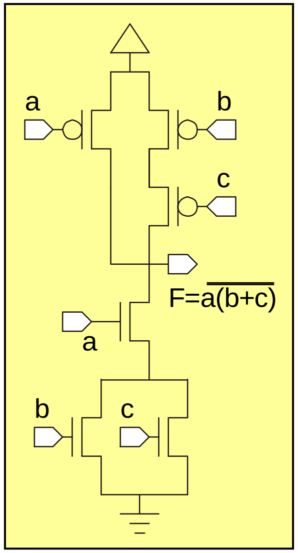
    $F = \overline{a} + \overline{(b + c)}$

    $F = \overline{a} + (\overline{b} \cdot \overline{c})$
  - Invert inputs for pMOS

    $Fp = a + (b \cdot c)$
  - Resulting Schematic



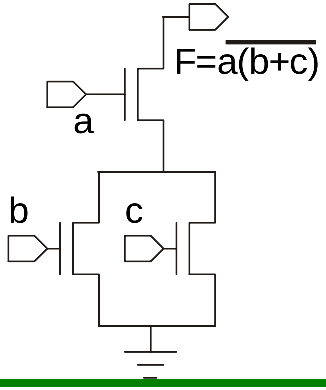
6 transistors (CMOS)



- nMOS
  - Invert output for nMOS

    $Fn = a \cdot (b + c)$
  - Apply DeMorgan

    none needed
  - Resulting Schematic



Structured Logic

- Recall CMOS is inherently Inverting logic
- Can used structured circuits to implement general logic functions
- AOI: implements logic function in the order

  AND, OR, NOT (Invert)

  - Example: $F = \overline{a \cdot b + c \cdot d}$
    - operation order: i) a AND b, c AND d, ii) (ab) OR (cd), iii) NOT
  - Inverted Sum-of-Products (SOP) form
- OAI: implements logic function in the order

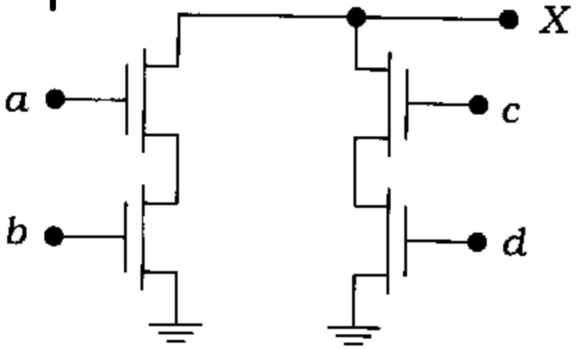
  OR, AND, NOT (Invert)

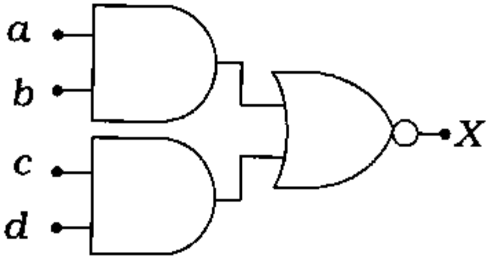
  - Example: $G = \overline{(x+y) \cdot (z+w)}$
    - operation order: i) x OR y, z OR w, ii) (x+y) AND (z+w), iii) NOT
  - Inverted Product-of-Sums (POS) form
- Use a ***structured CMOS array*** to realize such functions

AOI/OAI nMOS Circuits

- nMOS AOI structure
  - series txs in parallel

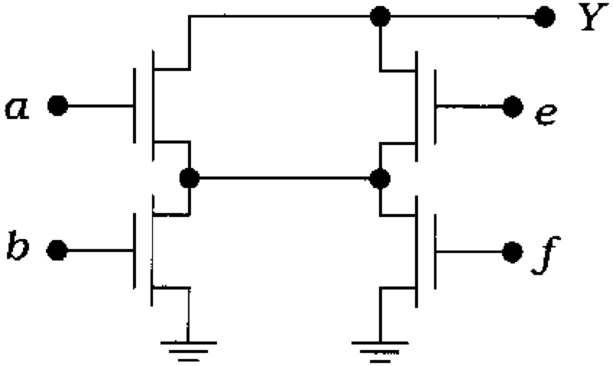
$X = \overline{a \cdot b + c \cdot d}$

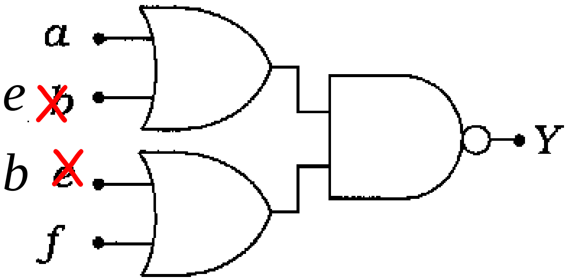




- nMOS OAI structure
  - series of parallel txs

$Y = \overline{a+e \cdot b+f}$

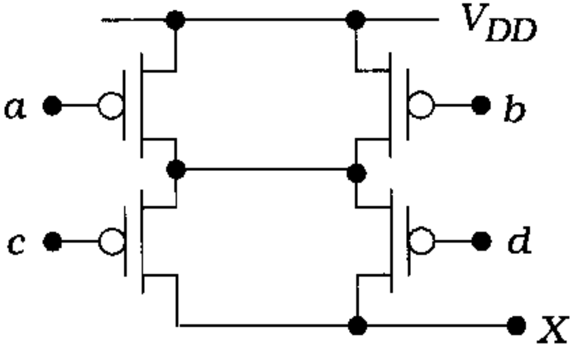




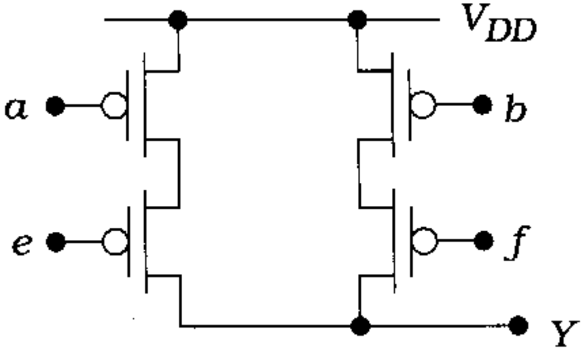
error in textbook Figure 2.45

AOI/OAI pMOS Circuits

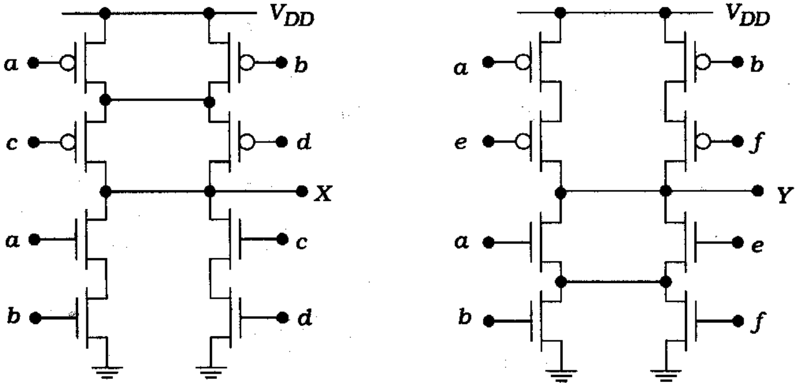
- pMOS AOI structure
  - series of parallel txs
  - opposite of nMOS (series/parallel)



- pMOS OAI structure
  - series txs in parallel
  - opposite of nMOS (series/parallel)



Complete CMOS AOI/OAI circuits



(a) AOI circuit (b) OAI circuit

Implementing Logic in CMOS

- Reducing Logic Functions
  - fewest operations ⇒ fewest txs
  - minimized function to eliminate txs
  - Example: x y + x z + x v = x (y + z + v)

5 operations:
3 AND, 2 OR

3 operations:
1 AND, 2 OR

# txs = ___?

# txs = ___?

- Suggested approach to implement a CMOS logic function
  - create nMOS network
    - invert output
    - reduce function, use DeMorgan to eliminate NANDs/NORs
    - implement using **series for AND** and **parallel for OR**
  - create pMOS network
    - complement each operation in nMOS network
      - i.e. make parallel into series and visa versa

CMOS Logic Example

- Construct the function below in CMOS

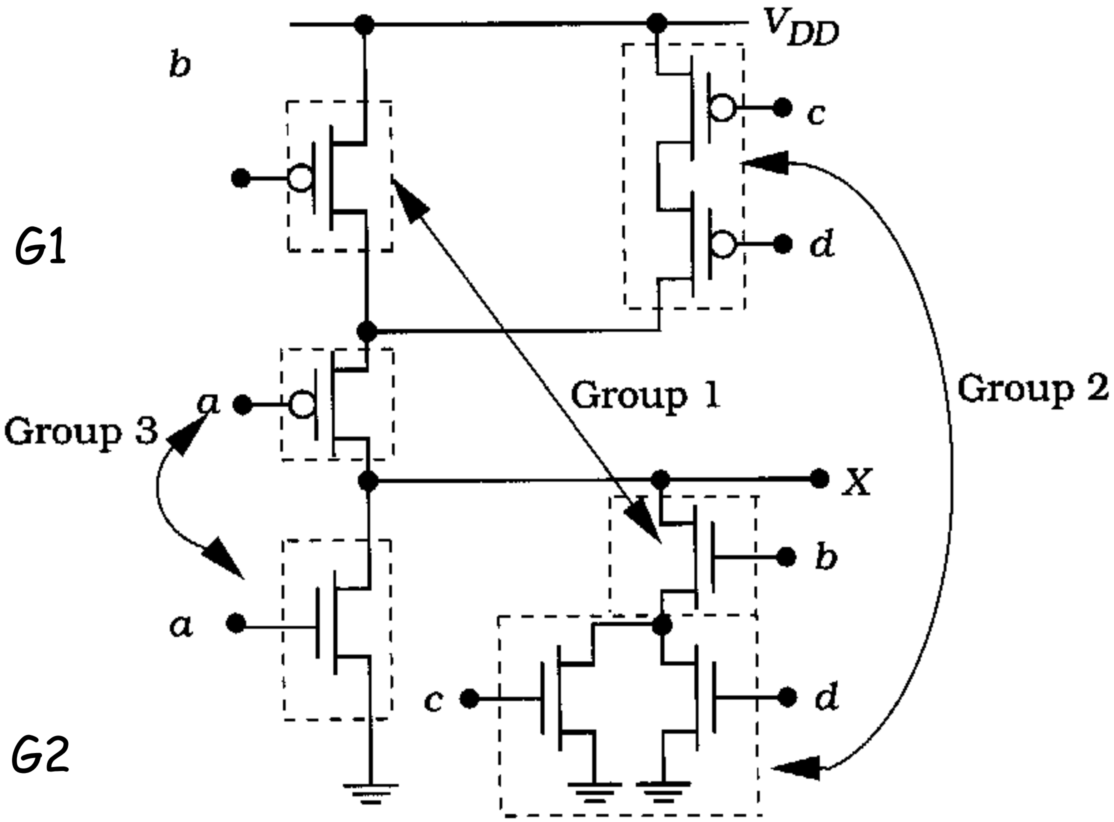
  $F = \overline{a + b \cdot (c + d)}$; remember AND operations occur before OR

- nMOS
  - Group 1: c & d in parallel
  - Group 2: b in series with G1
  - Group 3: a parallel to G2

follow same order in pMOS

don't compliment inputs

- pMOS
  - Group 1: c & d in series
  - Group 2: b parallel to G1
  - Group 3: a in series with G2



- Circuit has an OAOI organization (AOI with extra OR)

Another Combinational Logic Example

- Construct a CMOS logic gate which implements the function:

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

- pMOS
  - Apply DeMorgan expansions

    none needed
  - Invert inputs for pMOS

    $Fp = a \cdot (\overline{b} + c)$
  - Resulting Schematic ?

- nMOS
  - Invert output for nMOS

    $Fn = \overline{\overline{a} \cdot (b + \overline{c})}$
  - Apply DeMorgan

    $Fn = a + \overline{(b+\overline{c}\,)}$

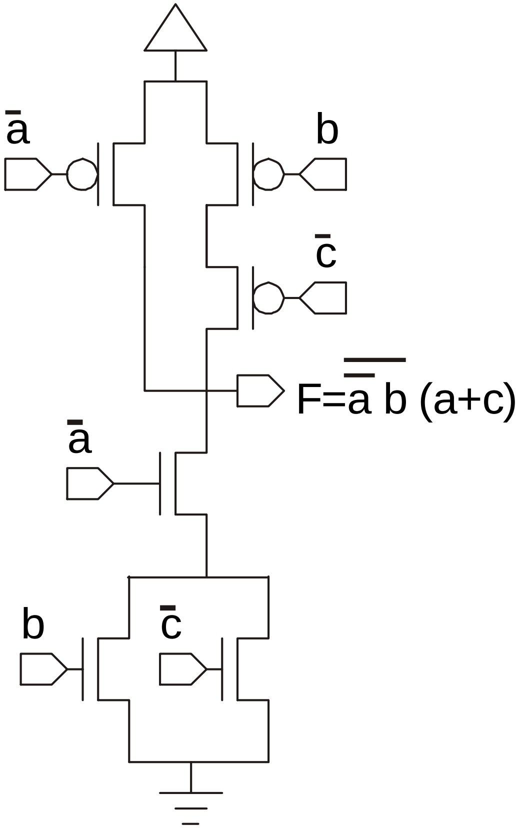
    $Fn = a + (\overline{b} \cdot c)$
  - Resulting Schematic ?

Yet Another Combinational Logic Example

- Implement the function below by constructing the nMOS network and complementing operations for the pMOS:

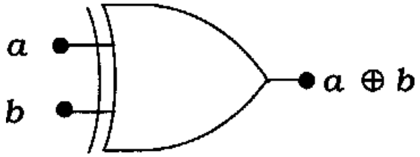
  $F = \overline{\overline{a} \cdot b} \cdot (a + c)$

- nMOS
  - Invert Output
    - $Fn = \overline{\overline{\overline{a} \cdot b} \cdot (a + c)} = \overline{a} \cdot b + \overline{(a + c)}$
  - Eliminate NANDs and NORs
    - $Fn = \overline{a} \cdot b + (\overline{a} \cdot \overline{c})$
  - Reduce Function
    - $Fn = \overline{a} \cdot (b + \overline{c})$
  - Resulting Schematic ?
  - Complement operations for pMOS
    - $Fp = \overline{a} + (b \cdot \overline{c})$



XOR and XNOR

- Exclusive-OR (XOR)
  - $a \oplus b = \overline{a} \cdot b + a \cdot \overline{b}$
  - not AOI form (no "I")



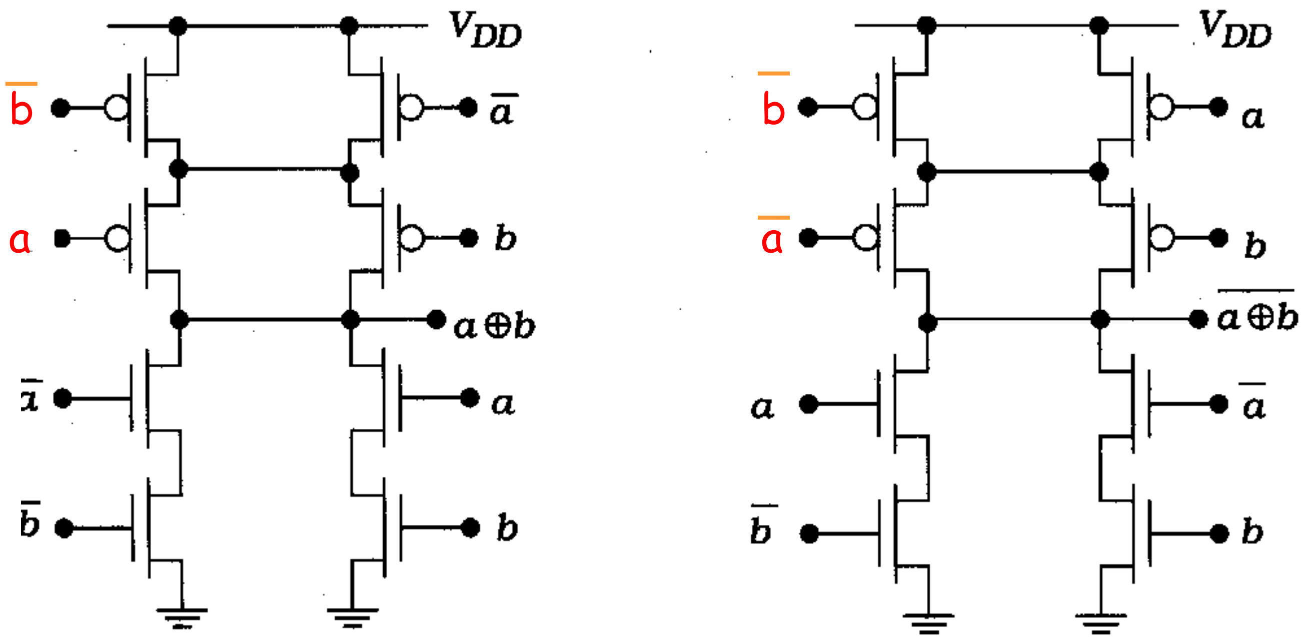
a	b	a ⊕ b
0	0	0
0	1	1
1	0	1
1	1	0

- Exclusive-NOR
  - $\overline{a \oplus b} = a \cdot b + \overline{a} \cdot \overline{b}$
  - inverse of XOR

- XOR/XNOR in AOI form
  - XOR: $\overline{\overline{a \oplus b}} = \overline{a \cdot b + \overline{a} \cdot \overline{b}}$, formed by complementing XNOR above
  - XNOR: $\overline{a \oplus b} = \overline{\overline{a} \cdot b + a \cdot \overline{b}}$, formed by complementing XOR

thus, interchanging a and $\overline{a}$ (or b and $\overline{b}$) converts from XOR to XNOR

XOR and XNOR AOI Schematic



(a) Exclusive-OR

(b) Exclusive-NOR

note: errors in textbook figure

–XOR: $a \oplus b = \overline{a \cdot b + \bar{a} \cdot \bar{b}}$

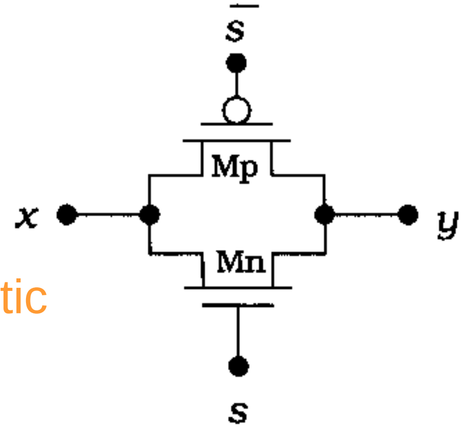
–XNOR: $\overline{a \oplus b} = \overline{\bar{a} \cdot b + a \cdot \bar{b}}$

uses exact same structure as generic AOI

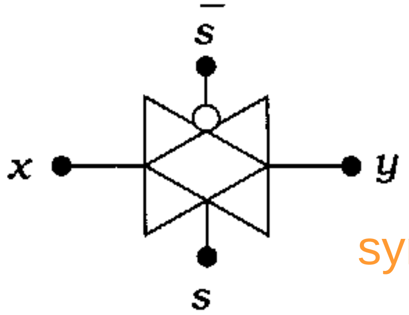
CMOS Transmission Gates

recall: pMOS passes a good ‘1’ and nMOS passes a good ‘0’

- Function
  - gated switch, capable of passing both ‘1’ and ‘0’
- Formed by a parallel nMOS and pMOS tx



schematic



symbol

- Controlled by gate select signals, s and $\bar{s}$
  - if s = 1, y = x, switch is **closed**, txs are **on**
  - if s = 0, y = unknown (high impedance), switch **open**, txs **off**

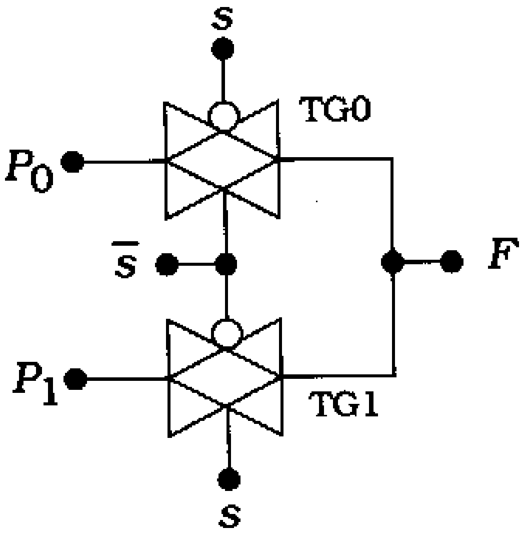
y = x s, for s=1

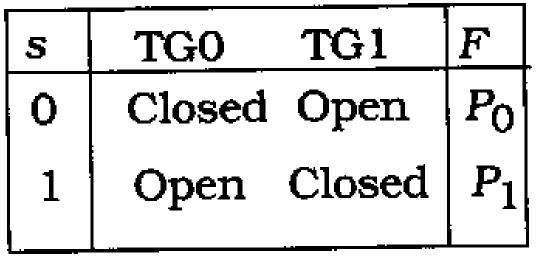
Transmission Gate Logic Functions

- TG circuits used extensively in CMOS
  - good switch, can pass full range of voltage (VDD-ground)

- **2-to-1 MUX** using TGs

$F = Po \bullet \overline{s} + P1 \bullet s$



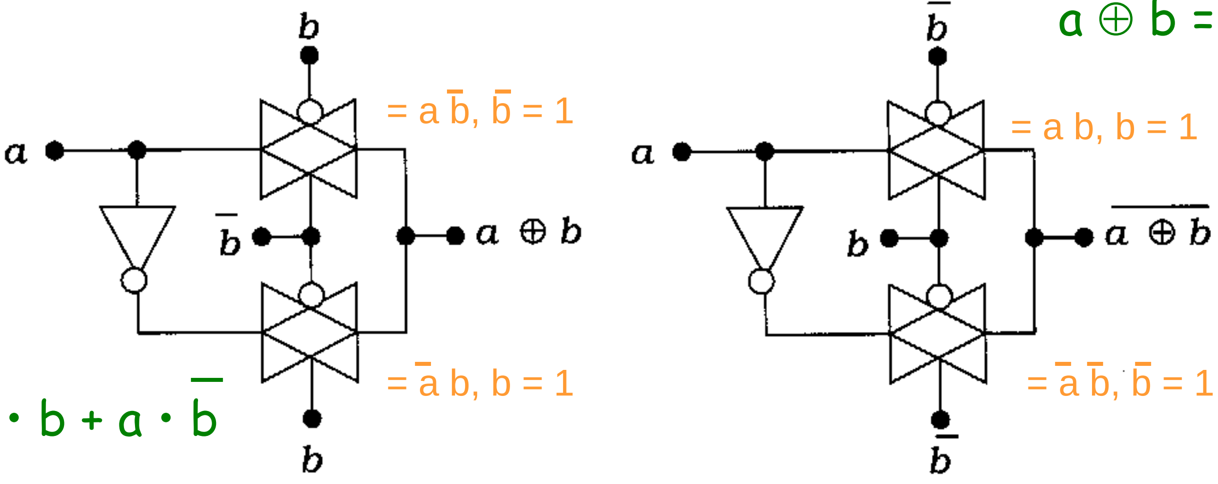


s	TG0	TG1	F
0	Closed	Open	P_0
1	Open	Closed	P_1

More TG Functions

- TG XOR and XNOR Gates

$\overline{a \oplus b} = a \cdot b + \bar{a} \cdot \bar{b}$



$= a\,\bar{b},\ \bar{b} = 1$

$= \bar{a}\,b,\ b = 1$

$= a\,b,\ b = 1$

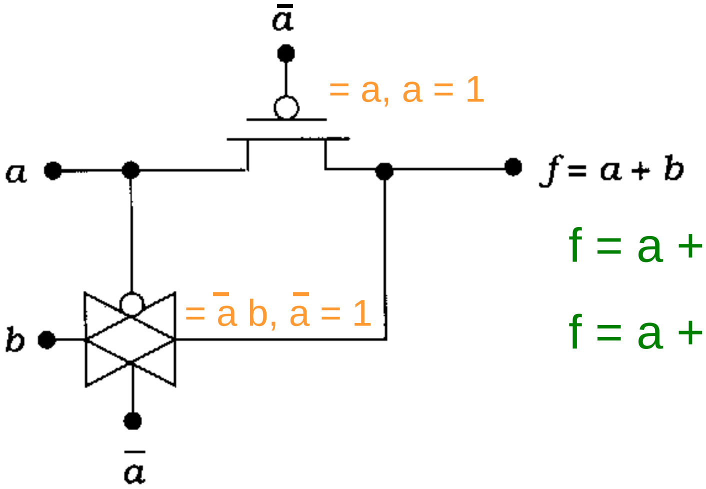
$= \bar{a}\,\bar{b},\ \bar{b} = 1$

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

(a) XOR circuit

(b) XNOR circuit

- Using TGs instead of "static CMOS"
  - TG OR gate



$= a,\ a = 1$

$= \bar{a}\,b,\ \bar{a} = 1$

$f = a + \bar{a}\,b$

$f = a + b$