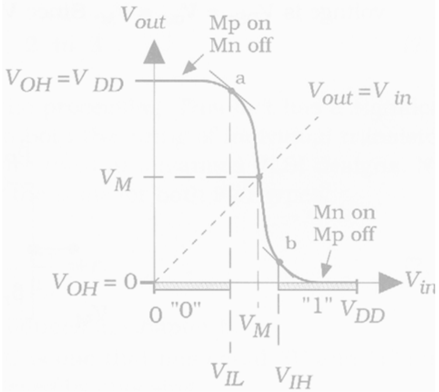
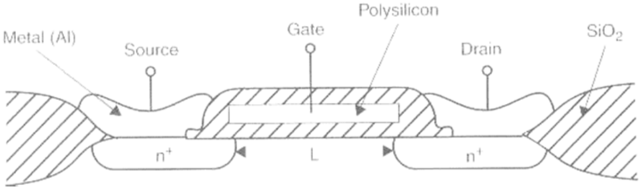
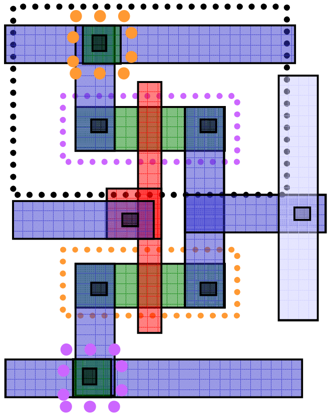


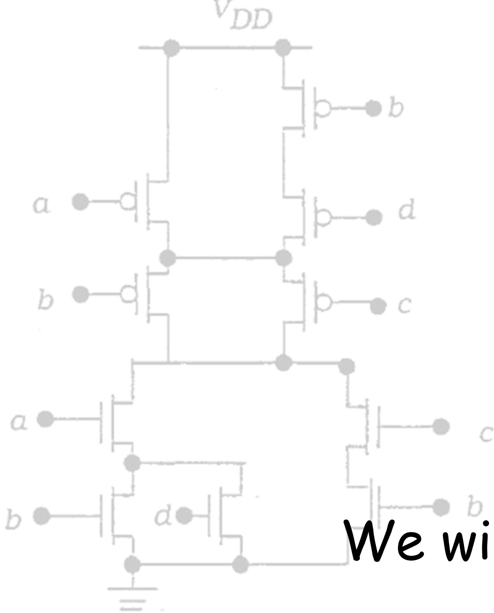
$$(a+b)\cdot(a+c) = a + a\cdot b + a\cdot$$
$$= a\cdot(1+b) + c$$
$$= a\cdot(1+c) + b$$
$$= a + b\cdot c$$

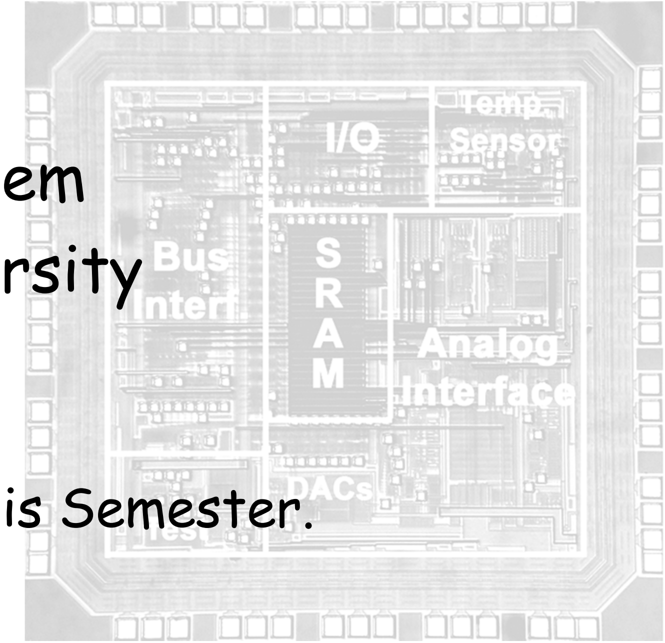




ECE 410: VLSI Design Course Lecture Notes

(Uyemura textbook)





Professor Fathi Salem

Michigan State University

We will be updating the notes this Semester.

Electronics Revolution

- Age of electronics
  - microcontrollers, DSPs, and other VLSI chips are everywhere
- Electronics of today and tomorrow
  - higher performance (speed) circuits
  - low power circuits for portable applications
  - more mixed signal emphasis
    - wireless hardware
    - high performance signal processing
    - Sensors, actuators, and microsystems

(Digital Camera), Camcorder, PDAs

MP3/CD Player Laptop Cell phone

Games: Nintendo; xbox, etc.

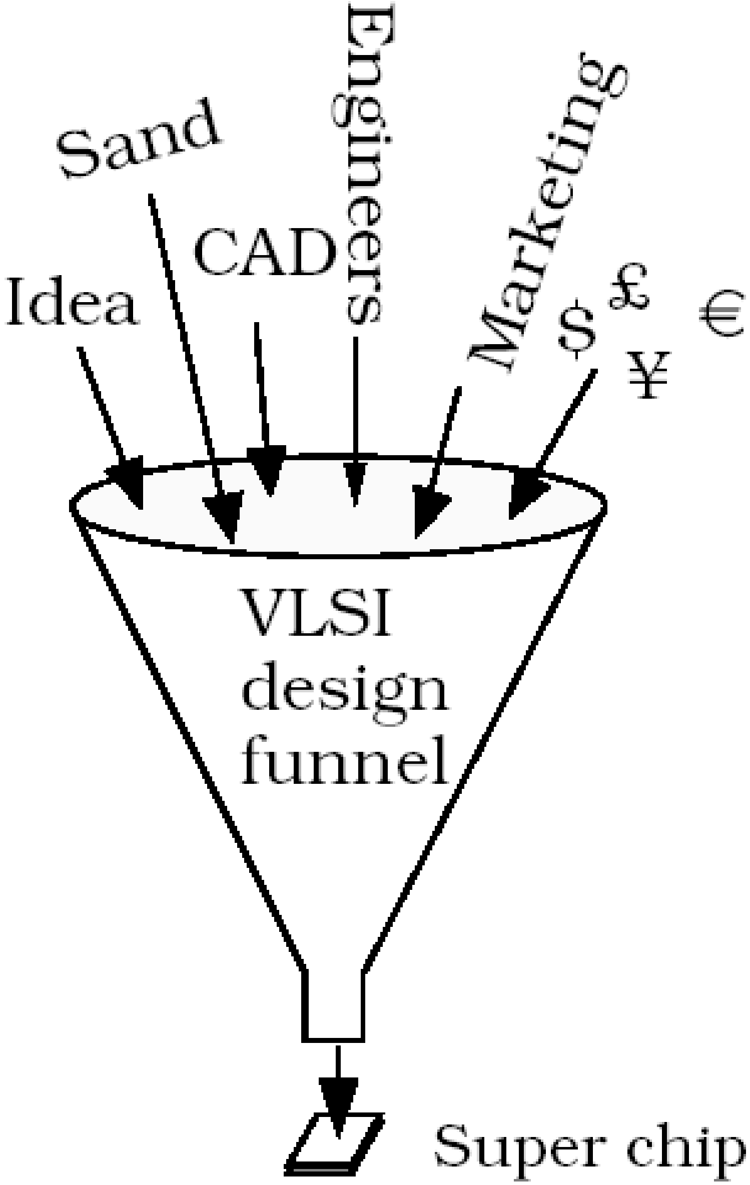
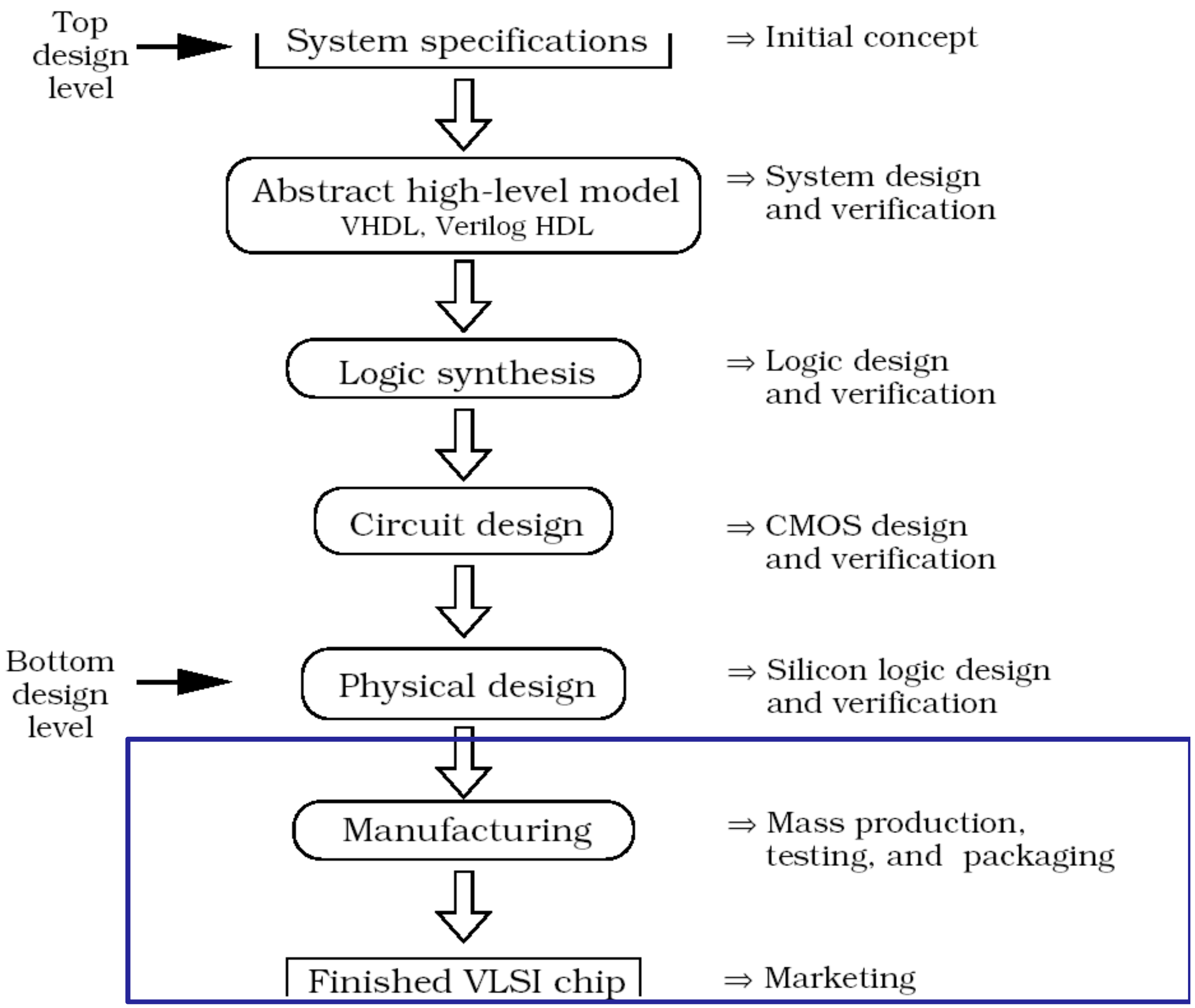


Figure 1.1 (p. 2)

The VLSI design funnel.

Figure 1.2 (p.4)

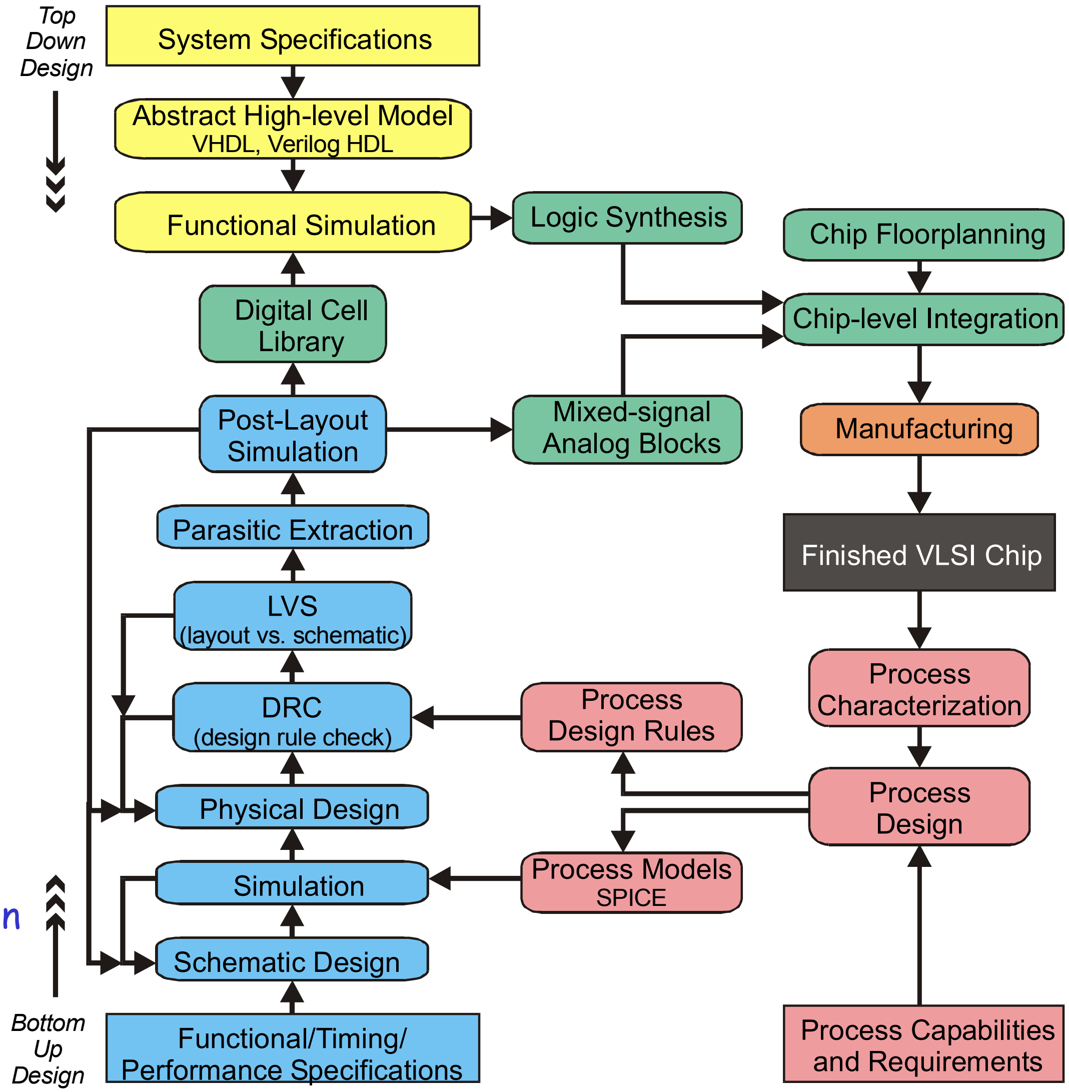
General overview of the design heirarchy.



Top design level
System specifications
⇒ Initial concept
Abstract high-level model
VHDL, Verilog HDL
⇒ System design and verification
Logic synthesis
⇒ Logic design and verification
Circuit design
⇒ CMOS design and verification
Bottom design level
Physical design
⇒ Silicon logic design and verification
Manufacturing
⇒ Mass production, testing, and packaging
Finished VLSI chip
⇒ Marketing

VLSI Design Flow

- VLSI
  - very large scale integration
  - lots of transistors integrated on a single chip
- Top Down Design
  - digital mainly
  - coded design
  - ECE 411
- Bottom Up Design
  - cell performance
  - Analog/mixed signal
  - ECE 410



VLSI Design Procedure

Integrated Circuit Technologies

- Why does CMOS dominate--Now?
  - other technologies
    - passive circuits
    - III-V devices
    - Silicon BJT

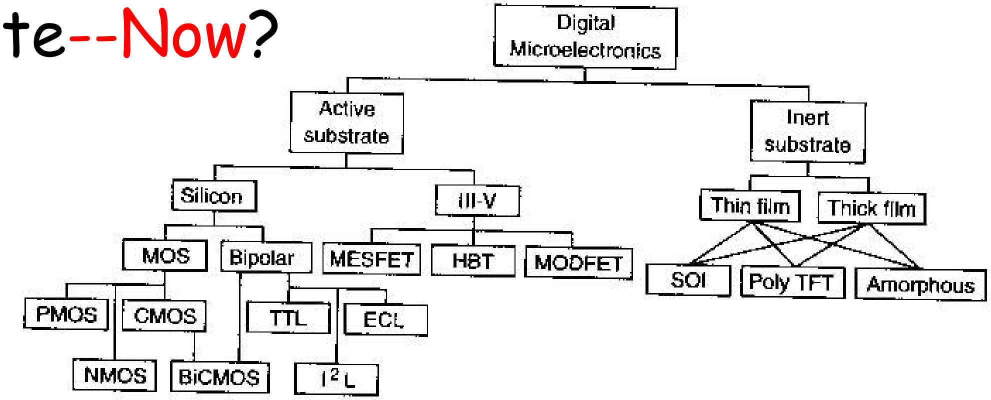


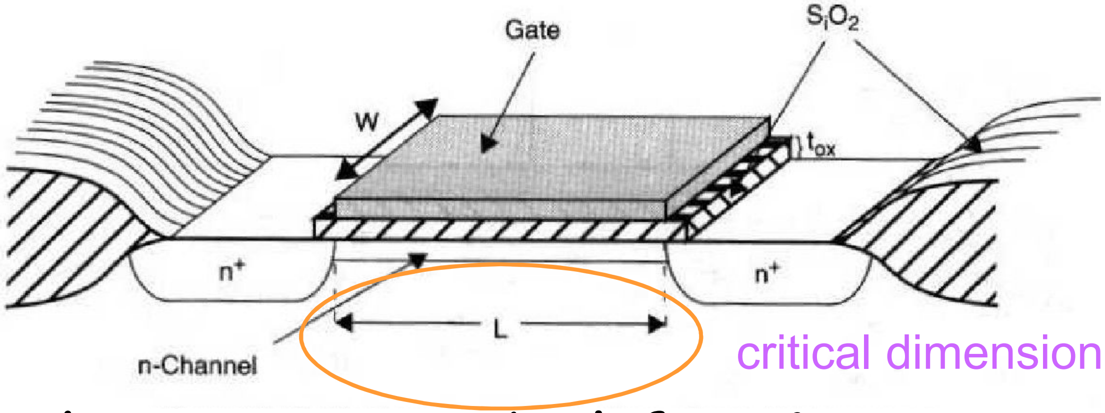
Fig. 1.1 Family of digital IC.

- CMOS dominates because:
  - **Silicon is cheaper** → preferred over other materials
  - **physics** of CMOS is **easier** to understand???
  - CMOS is **easier to implement/fabricate**
  - CMOS provides **lower power-delay product**
  - CMOS is **lowest power** ★
  - can get **more** CMOS transistors/functions **in same chip area**

- **BUT!** CMOS is <u>not</u> the fastest technology!
  - BJT and III-V devices are faster

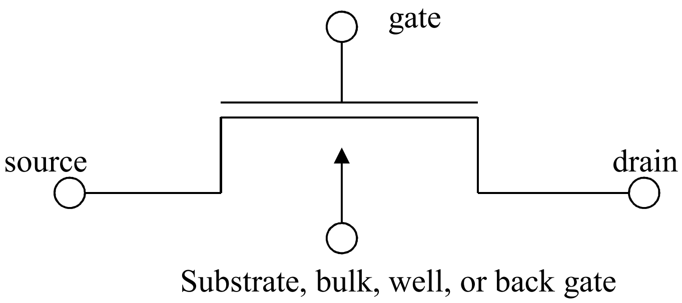
MOSFET Physical View

- Phvsical Structure of a MOSFET Device

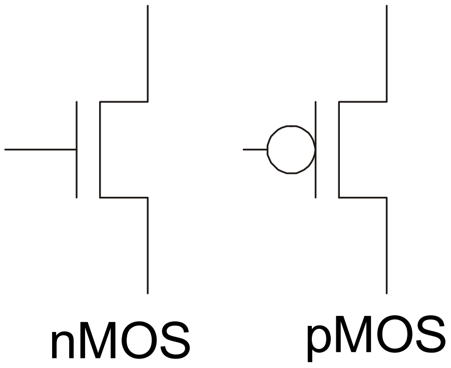


critical dimension = “feature size”

- Schematic Symbol for 4-terminal MOSFET

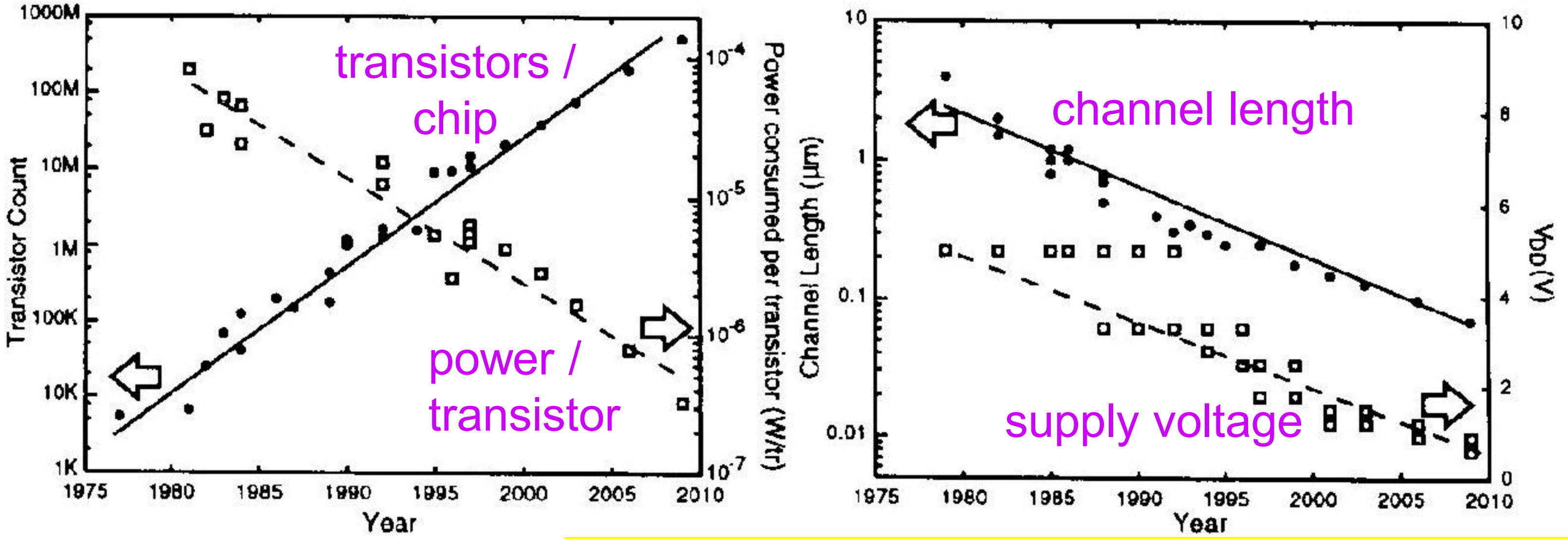


- Simplified Symbols



CMOS Technology Trends

- Variations over time
  - # transistors / chip: increasing with time
  - power / transistor: decreasing with time (constant power density)
  - device channel length: decreasing with time
  - power supply voltage: decreasing with time



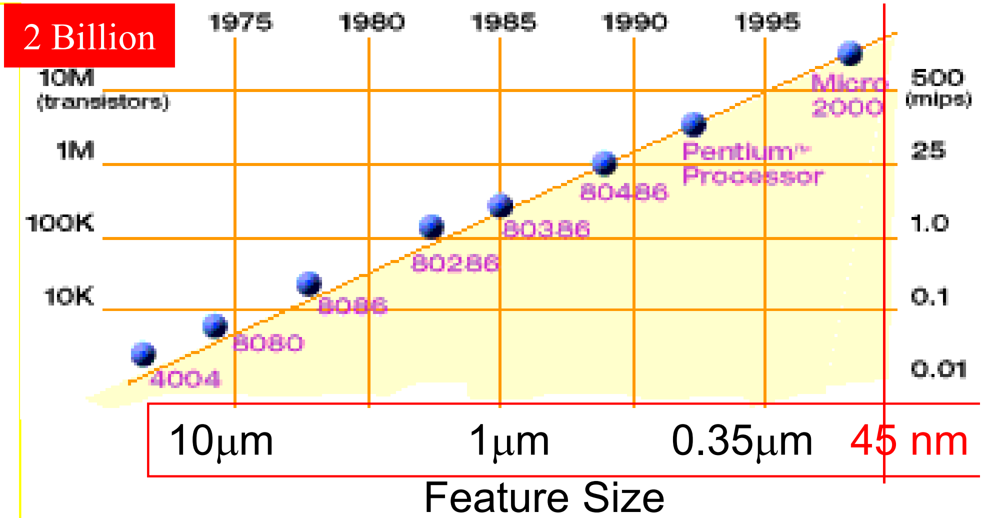
ref: Kuo and Lou, Low-Voltage CMOS VLSI Circuits, Fig. 1.3, p. 3

low power/transistor is critical for future ICs

Moore's Law

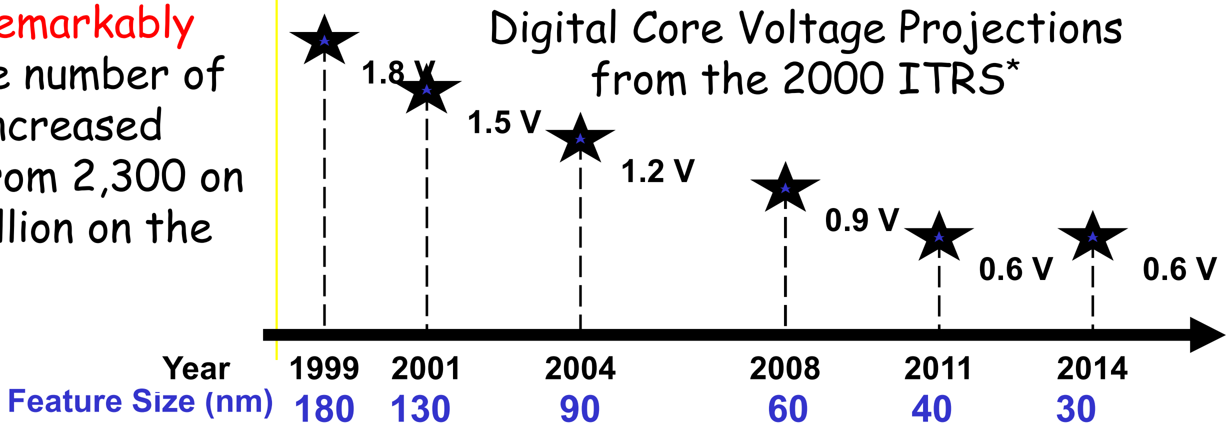
- In 1965, Gordon Moore realized there was a striking trend; each new generation of memory chip contained roughly ***twice as much capacity*** as its predecessor, and each chip was released within ***18-24 months*** of the previous chip. He reasoned, computing power would rise exponentially over relatively brief periods of time.

- Moore's observation, now known as ***Moore's Law***, described a trend that has continued and is still remarkably accurate. ***In 26 years*** the number of transistors on a chip has increased more than ***3,200 times***, from 2,300 on the 4004 in 1971 to 7.5 million on the Pentium¨ II processor.



(ref: http://www.intel.com/intel/museum/25anniv/hof/moore.htm)

Power Supply Tends



* http://public.itrs.net/Files/2000UpdateFinal/ORTC2000final.pdf

“Electronics” Building block(s)

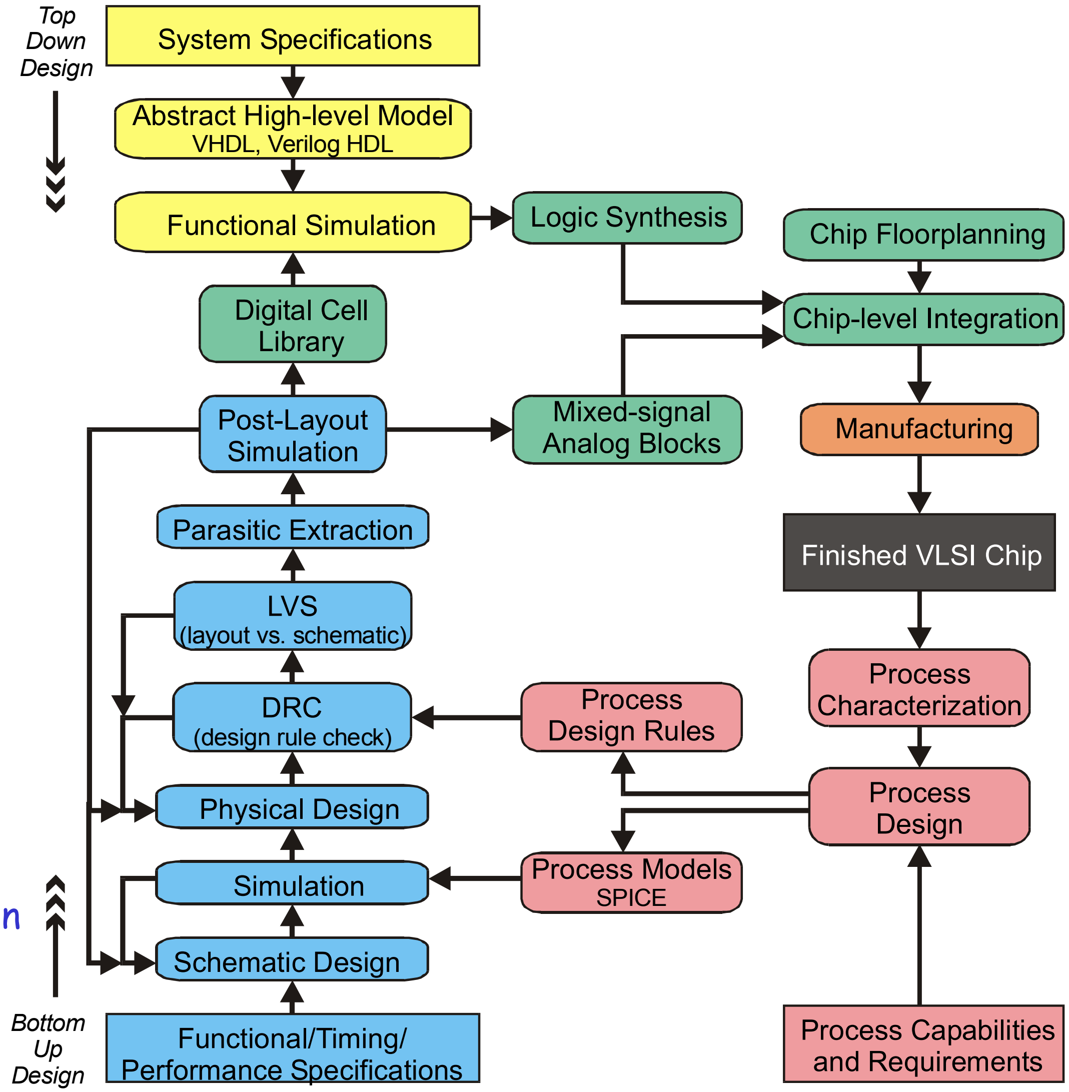
- MOSFET Device-- 1950+ to 2020

- New elements in nano technologies are emerging. These include:
  - Fin-Transistor
  - Memristor: memory resistor- see IEEE Spectrum
  - Nano-tubes
  - Molecular devices
  - Quantum dots
  - Etc.

VLSI Design Flow

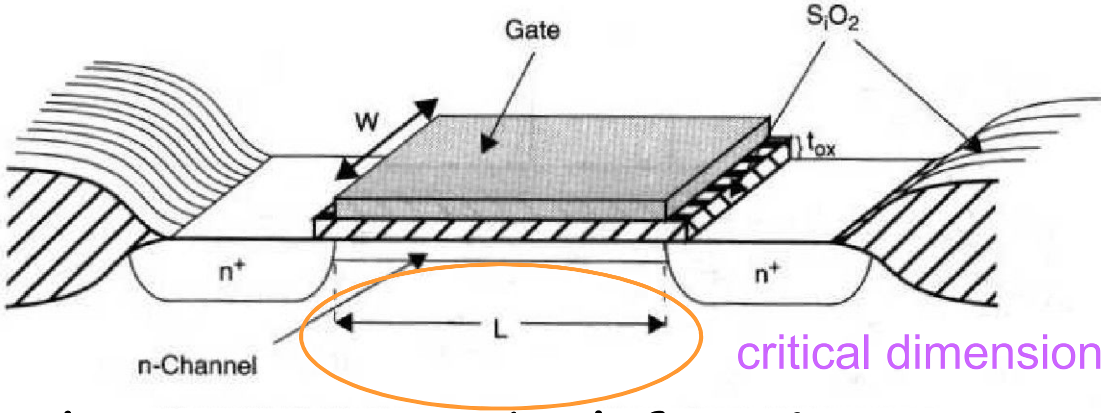
- VLSI
  - very large scale integration
  - lots of transistors integrated on a single chip
- Top Down Design
  - digital mainly
  - coded design
  - ECE 411
- Bottom Up Design
  - cell performance
  - Analog/mixed signal
  - ECE 410

VLSI Design Procedure



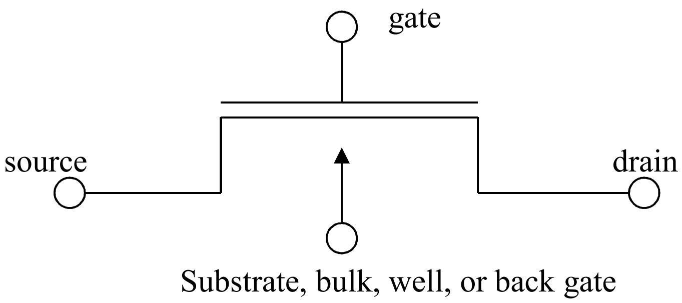
MOSFET Physical View

- Phvsical Structure of a MOSFET Device

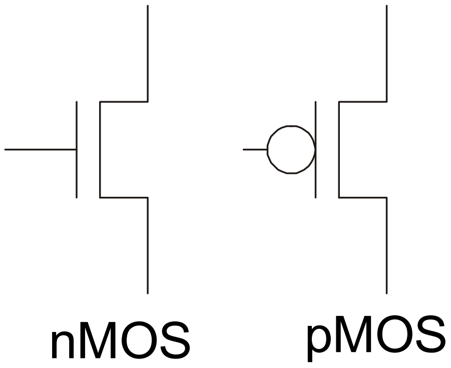


critical dimension = “feature size”

- Schematic Symbol for 4-terminal MOSFET

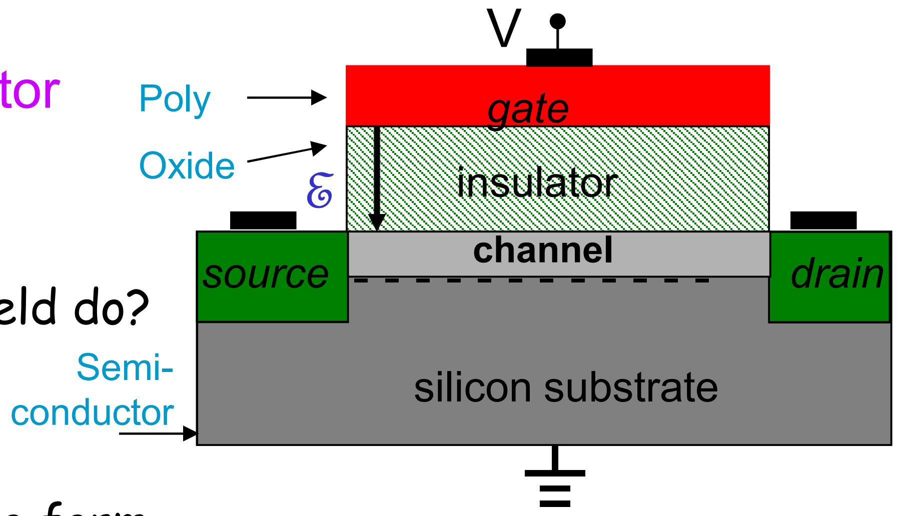


- Simplified Symbols



What is a MOSFET?

- Digital integrated circuits rely on transistor switches
  - most common device for digital and mixed signal: MOSFET
- Definitions
  - MOS = Metal Oxide Semiconductor
    - physical layers of the device
  - FET = Field Effect Transistor
    - What field? What does the field do?
    - Are other fields important?
  - CMOS = Complementary MOS
    - use of both nMOS and pMOS to form a circuit with lowest power consumption.

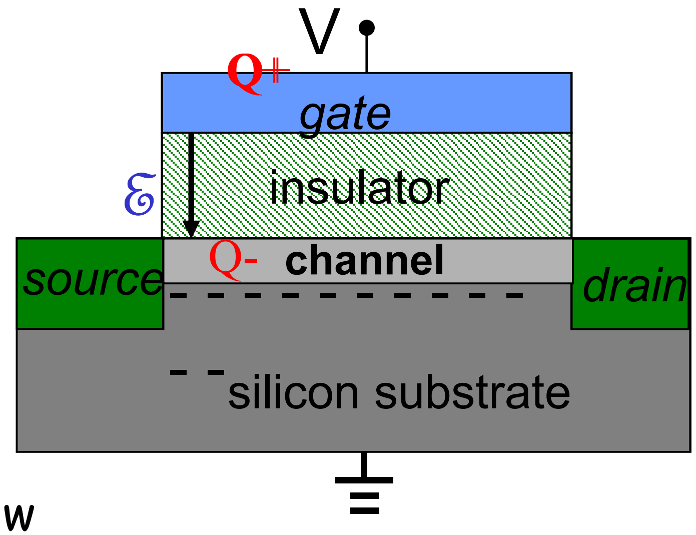


- Primary Features
  - gate; gate oxide (insulator)– very thin (~10^(-10))-- exaggerated in Fig.
  - source and drain
  - channel
  - bulk/substrate

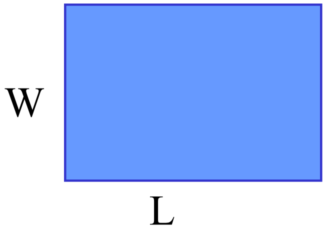
NOTE: “Poly” stands for polysilicon in modern MOSFETs

Fundamental Relations in MOSFET

- Electric Fields
  - fundamental equation
    - electric field: $E = V/d$
  - vertical field through gate oxide
    - determines charge induced in channel
  - horizontal field across channel
    - determines source-to-drain current flow



- Capacitance
  - fundamental equations
    - capacitor charge: $Q = CV$
    - capacitance: $C = \varepsilon A/d$
  - charge balance on capacitor, Q+ = Q-
    - charge on gate is balanced by charge in channel
    - what is the source of channel charge? where does it come from?



← Topview

CMOS Cross Section View

- Cross section of a 2 metal, 1 poly CMOS process

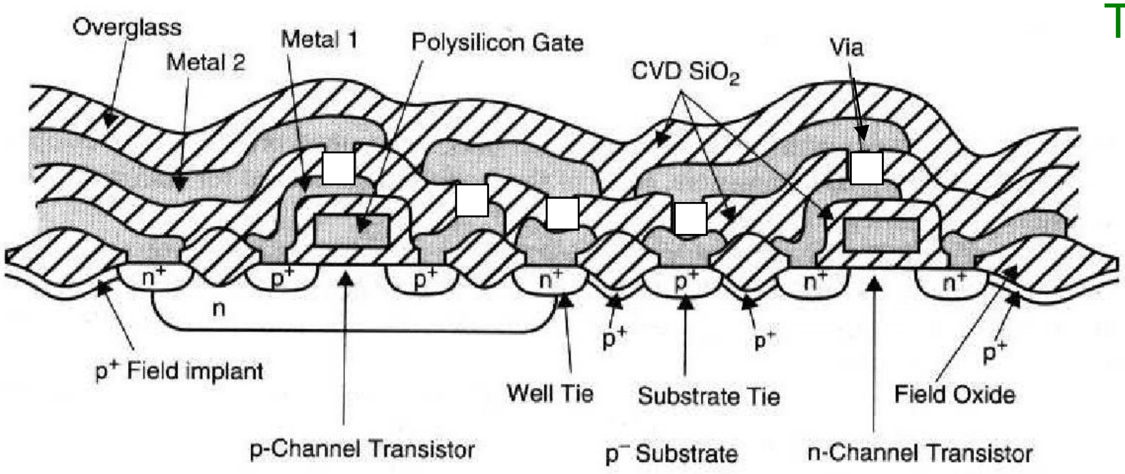
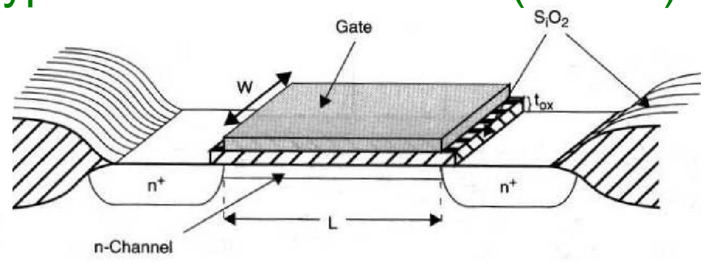
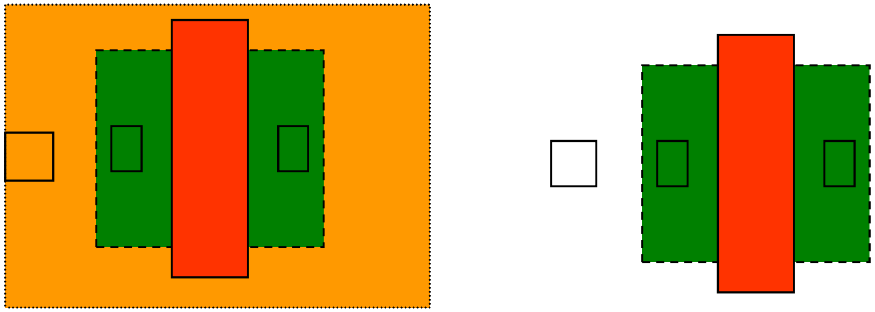


Figure 2.11 The final cross section of a CMOS microcircuit with two layers of metal.

Typical MOSFET Device (nMOS)

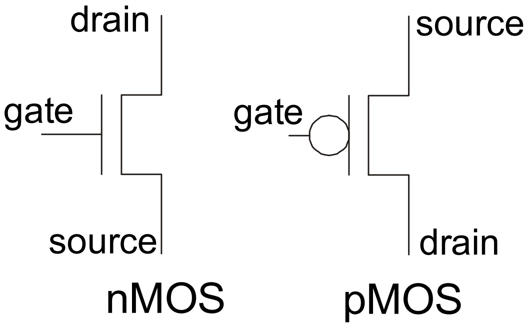


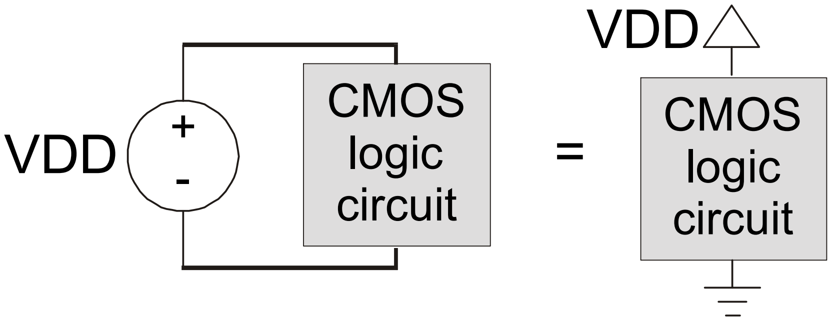
- Layout (top view) of the devices above (partial, simplified)

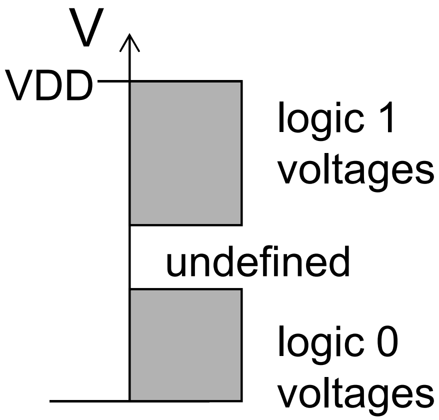


CMOS Circuit Basics

- **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 ranges from (0.6V) 1V to 5V
- Logic Levels (voltage-based)
  - all voltages between 0V and VDD
  - Logic '1' = VDD
  - Logic '0' = ground = 0V







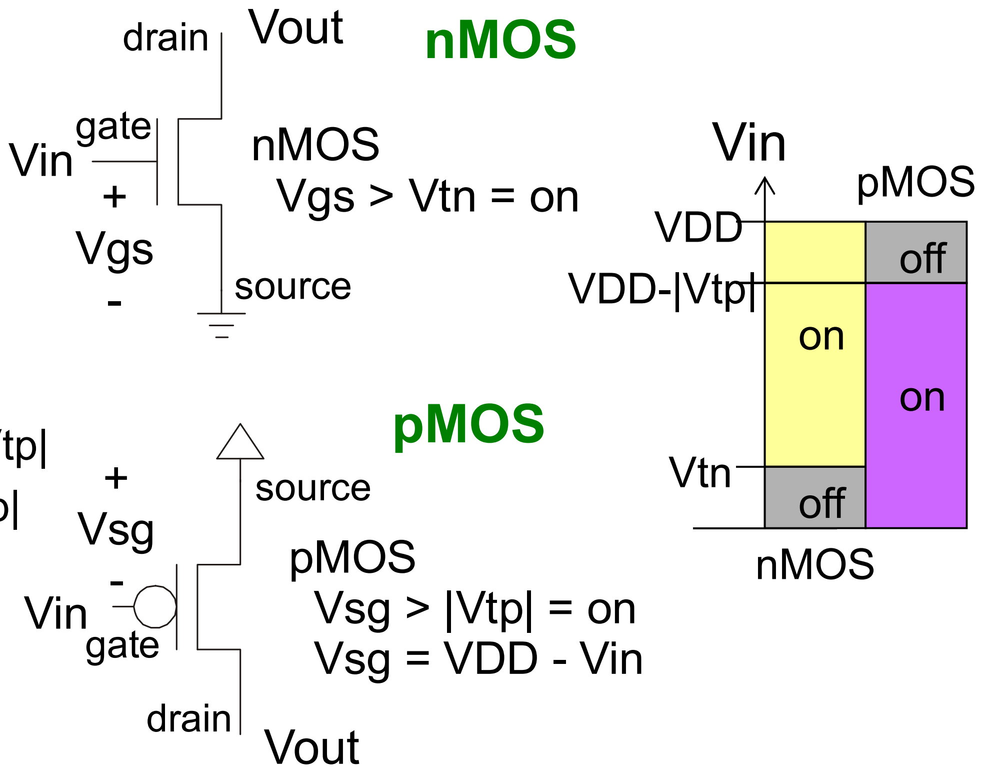
Transistor Switching Characteristics

- nMOS
  - switching behavior
    - on = closed, when Vin > Vtn
    - off = open, when Vin < Vtn
- pMOS
  - switching behavior
    - on = closed, when Vin < VDD - |Vtp|
    - off = open, when Vin > VDD - |Vtp|
- 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



Rule to Remember

'source' is at

- lowest potential for nMOS
- highest potential for pMOS

MOSFET Pass Characteristics

- Each type of transistor is better at passing (to output) one digital voltage than the other
  - nMOS passes a good low (0) but not a good high (1)
  - pMOS passes a good high (1) but not a good low (0)

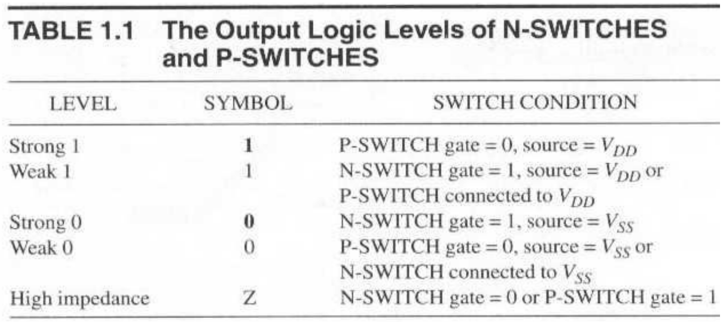
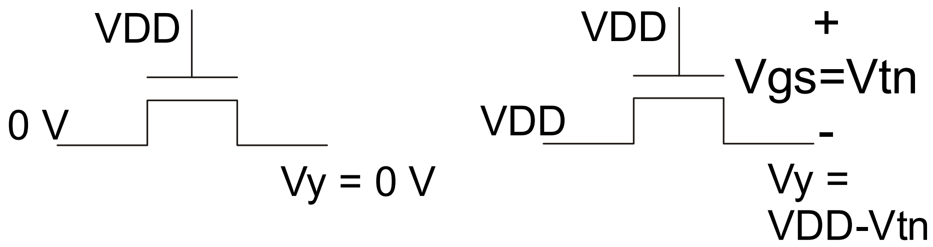


TABLE 1.1 The Output Logic Levels of N-SWITCHES and P-SWITCHES

LEVEL	SYMBOL	SWITCH CONDITION
Strong 1	**1**	P-SWITCH gate = 0, source = V_{DD}
Weak 1	1	N-SWITCH gate = 1, source = V_{DD} or P-SWITCH connected to V_{DD}
Strong 0	**0**	N-SWITCH gate = 1, source = V_{SS}
Weak 0	0	P-SWITCH gate = 0, source = V_{SS} or N-SWITCH connected to V_{SS}
High impedance	Z	N-SWITCH gate = 0 or P-SWITCH gate = 1

nMOS

on when gate is 'high'

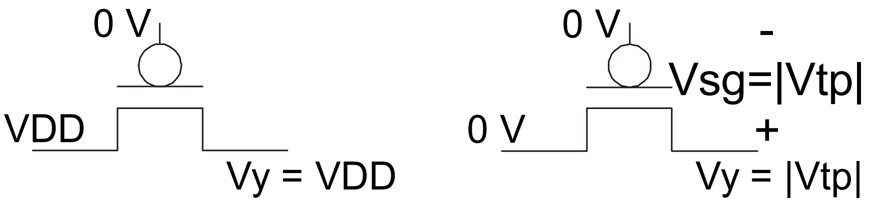


Passes a good low

Max high is VDD-Vtn

pMOS

on when gate is 'low'



Passes a good high

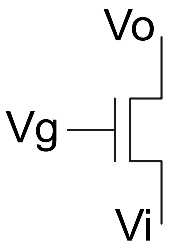
Min low is |Vtp|

Rule to Remember

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

MOSFET Terminal Voltages

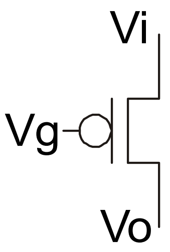
- How do you determine one terminal voltage if other 2 are known?
  - nMOS



    - 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
    - Example (Vtn=0.5V): Vg=5V, Vi=2V ⇒ Vo = 2V
      Vg=2V, Vi=2V ⇒ Vo = 1.5V

For nMOS,
max(Vo) = Vg-Vtn

  - pMOS

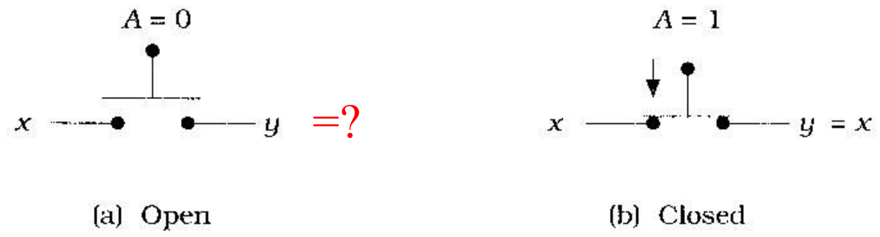


    - case 1) if Vg < Vi - |Vtp|, then Vo = Vi (Vi-Vg > |Vtp|)
      - here Vi is the "source" so the pMOS will pass Vi to Vo
    - case 2) if Vg > Vi - |Vtp|, then Vo = Vg+|Vtp| (Vi-Vg < |Vtp|)
      - here Vo is the "source" so the pMOS output is limited
    - Example (Vtp=-0.5V): Vg=2V, Vi=5V ⇒ Vo = 5V
      Vg=2V, Vi=2V ⇒ Vo = 2.5V

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

Switch-Level Boolean Logic

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



=?

(a) Open (b) Closed

Figure 2.1 Behavior of an assert-high switch

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

– y = x • A, i.e. y = x iff A = 1 (iff=if and only if)

Series switches ⇒ AND function

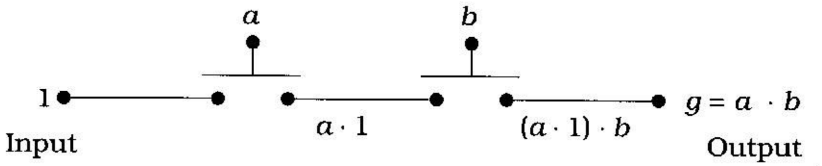


Figure 2.2 Series-connected switches

Parallel switches ⇒ OR function

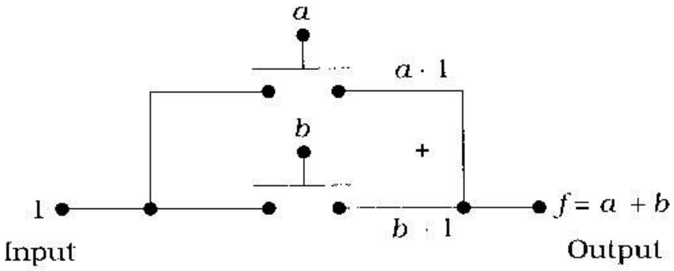
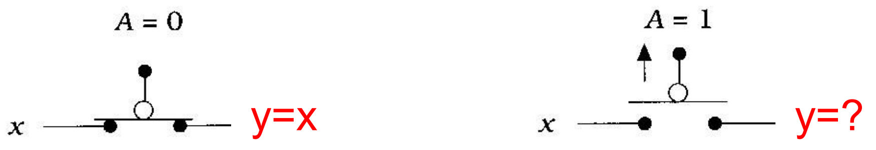


Figure 2.4 Parallel-connected switches

Switch-Level Boolean Logic

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



(a) Closed

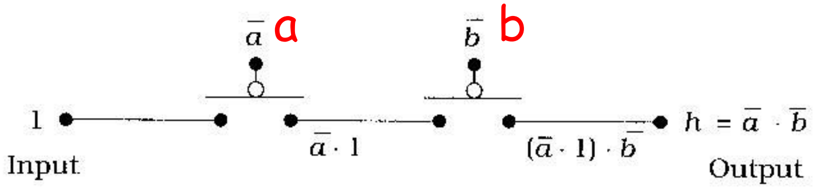
(b) Open

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 ⇒ ?


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

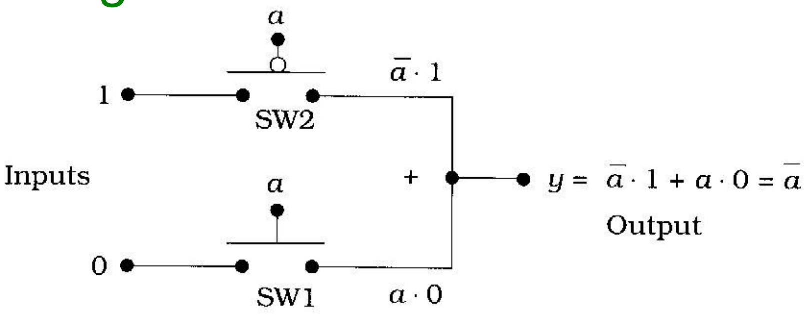
NOR

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

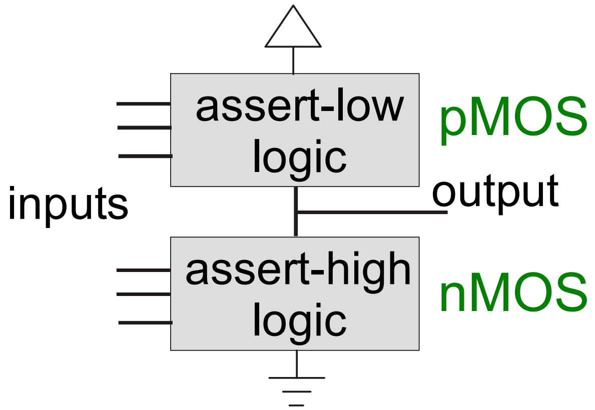


a=1 ⇒ SW1 closed, SW2 open ⇒ y=0 = $\overline{a}$

a=0 ⇒ SW1 open, SW2 closed ⇒ y=1 = a

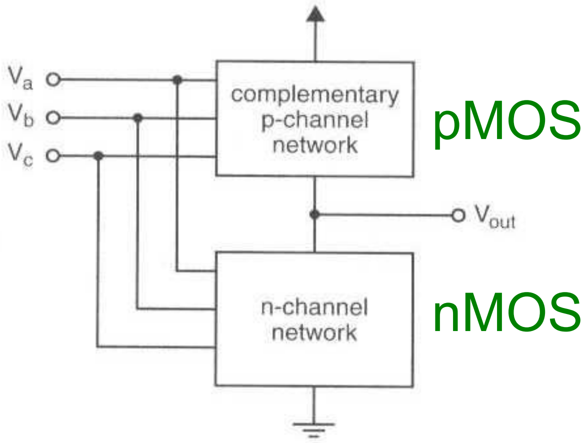
CMOS “Push-Pull” Logic

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



- only one logic network (p or n) is required to produce (1/2-) the logic function???

- but the complementary set allows the “load” to be turned off for zero **static** power dissipation



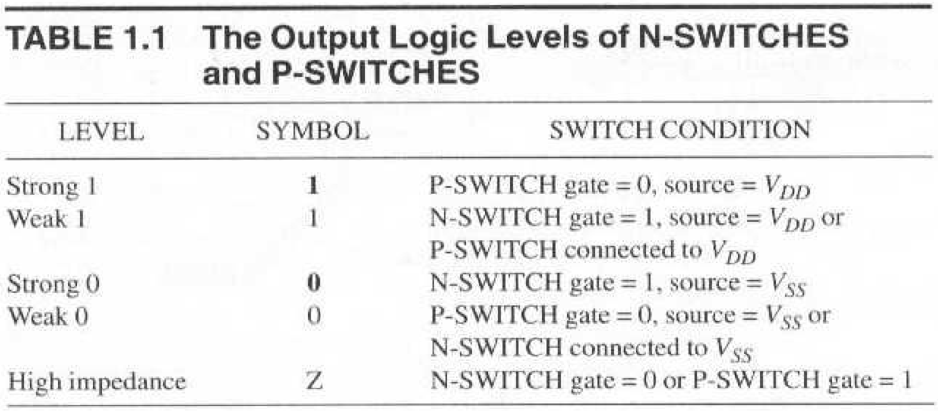


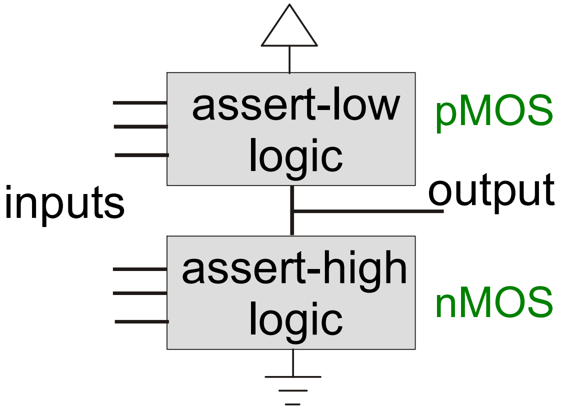
TABLE 1.1 The Output Logic Levels of N-SWITCHES and P-SWITCHES

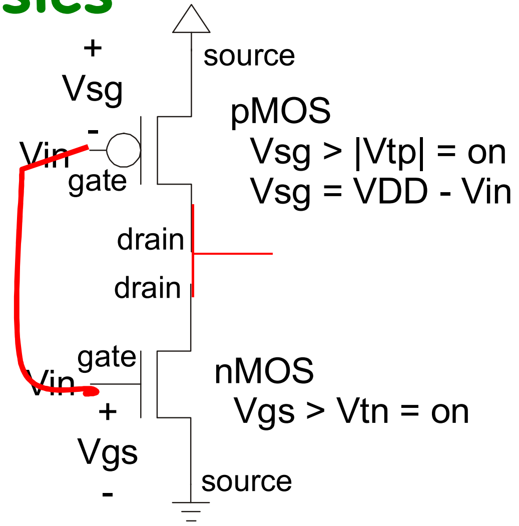
LEVEL	SYMBOL	SWITCH CONDITION
Strong 1	**1**	P-SWITCH gate = 0, source = V_{DD}
Weak 1	1	N-SWITCH gate = 1, source = V_{DD} or P-SWITCH connected to V_{DD}
Strong 0	**0**	N-SWITCH gate = 1, source = V_{SS}
Weak 0	0	P-SWITCH gate = 0, source = V_{SS} or N-SWITCH connected to V_{SS}
High impedance	Z	N-SWITCH gate = 0 or P-SWITCH gate = 1

VSS = ground

Review: Basic Transistor Operation

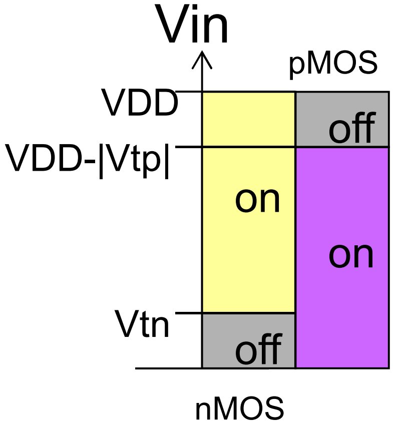
CMOS Circuit Basics





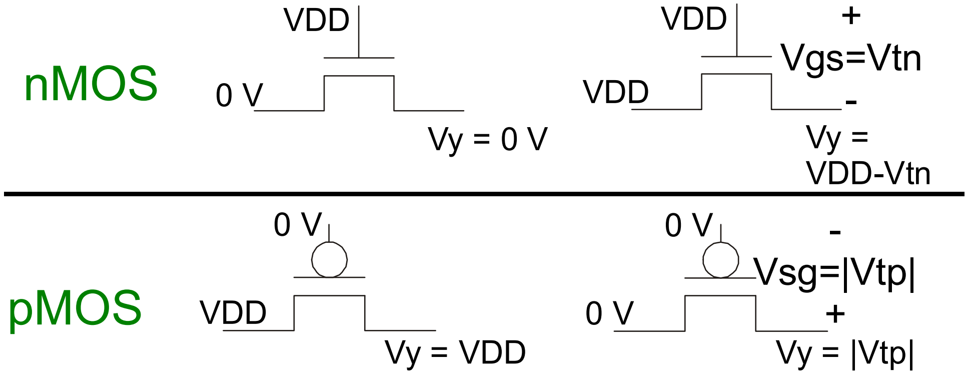
Vg= Vin	Vout	
0	1	on = closed
1	?	off = open

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



CMOS Pass Characteristics

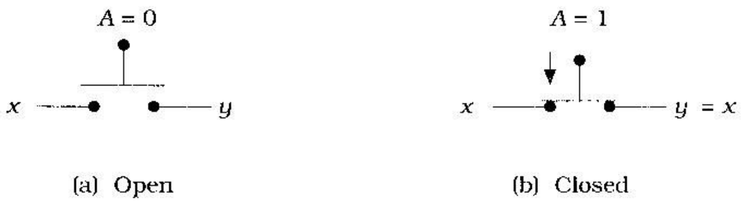
‘source’ is at lowest potential (nMOS) and highest potential (pMOS)



- 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

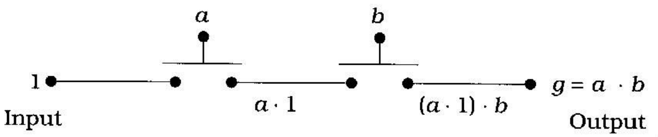
- **assert-high** switch
  - y = x • A, i.e. y = x iff A = 1



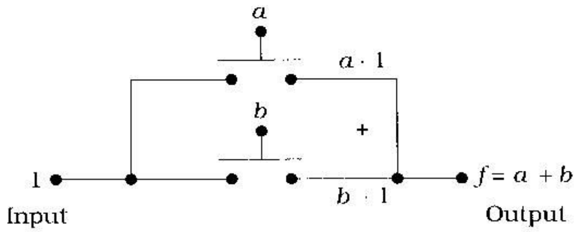
(a) Open (b) Closed

ı assert-high switch

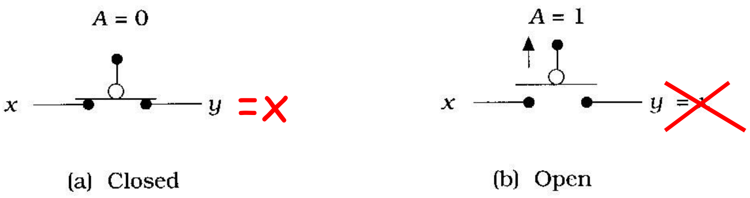
  - series = AND



  - parallel = OR

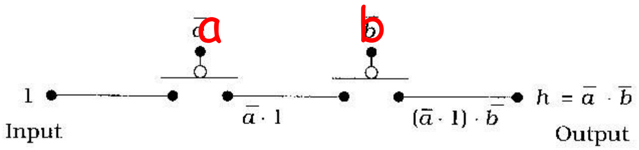


- **assert-low** switch
  - y = x • $\overline{A}$, i.e. y = x if A = 0



(a) Closed (b) Open

  - series = NOR
  - parallel = NAND

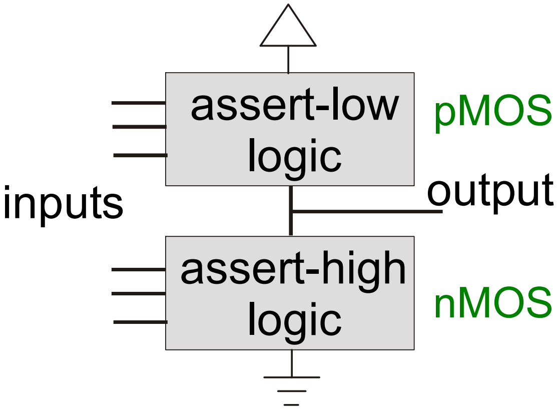


$\bar{a}$
a
$\bar{b}$
b
1
Input
$\bar{a} \cdot 1$
$(\bar{a} \cdot 1) \cdot \bar{b}$
$h = \bar{a} \cdot \bar{b}$
Output

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
  - ensure 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 · b)' = a' + b'

(a + b)' = a' · b'

Useful Logic Properties

1 + x = 1   0 + x = x

1 · x = x   0 · x = 0

x + x' = 1   x · x' = 0

a · a = a   a + a = a

ab + ac = a (b+c)

Properties which can be proven

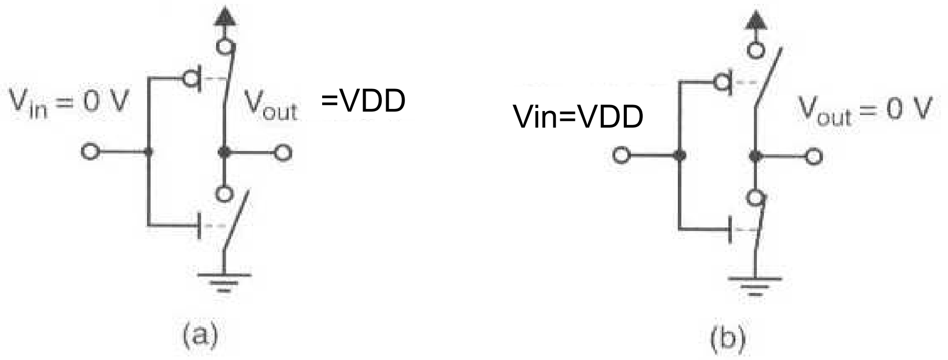
(a+b)(a+c) = a+bc

a + a'b = a + b

CMOS Inverter

- Inverter Function
  - toggle binary logic of a signal

- Inverter Switch Operation



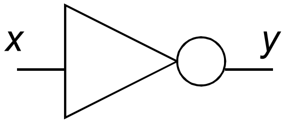
input low → output high
nMOS off/open
pMOS on/closed

pMOS “on”
→ output high (1)

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

nMOS “on”
→ output low (0)

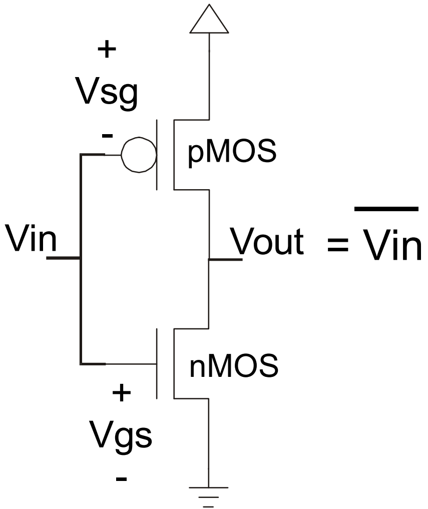
- Inverter Symbol



- Inverter Truth Table

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

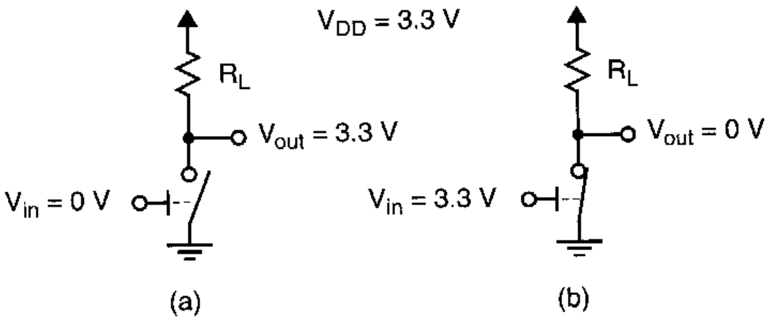
- CMOS Inverter Schematic



nMOS Logic Gates

- Study nMOS logic first, more simple than CMOS
- nMOS Logic
  - assume a resistive load to VDD
  - nMOS switches pull output low based on inputs

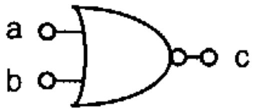
nMOS Inverter



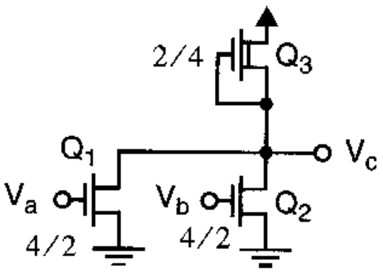
(a) nMOS is off
→ output is high (1)

(b) nMOS is on
→ output is low (0)

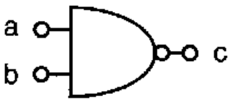
nMOS NOR



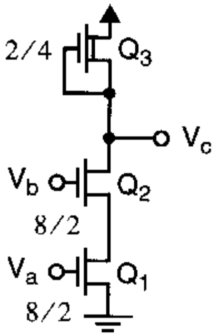
c = $\overline{a+b}$



nMOS NAND



c = $\overline{ab}$

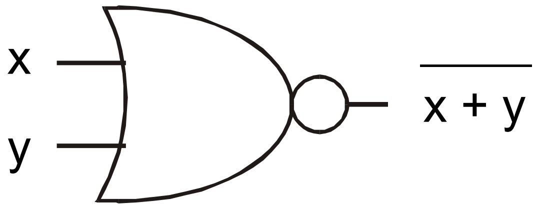


- parallel switches = OR function
- nMOS pulls low (NOTs the output)

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

CMOS NOR Gate

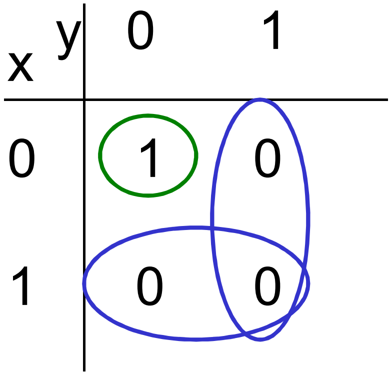
- NOR Symbol



- NOR Truth Table

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

- Karnaugh map



x \ y	0	1
0	1	0
1	0	0

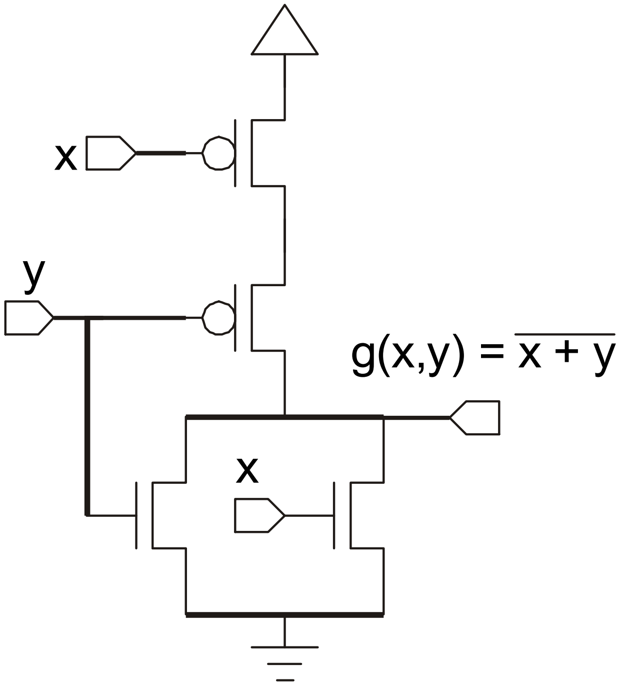
$$g(x,y) = \overline{x} \bullet \overline{y} \bullet 1 + x \bullet 0 + y \bullet 0$$

- 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

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

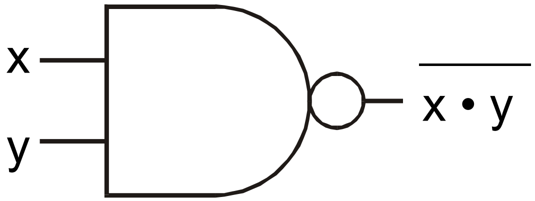


- **output is LOW if *x* OR *y* is true**
  - **parallel nMOS**
- **output is HIGH when *x* AND *y* are false**
  - **series pMOS**

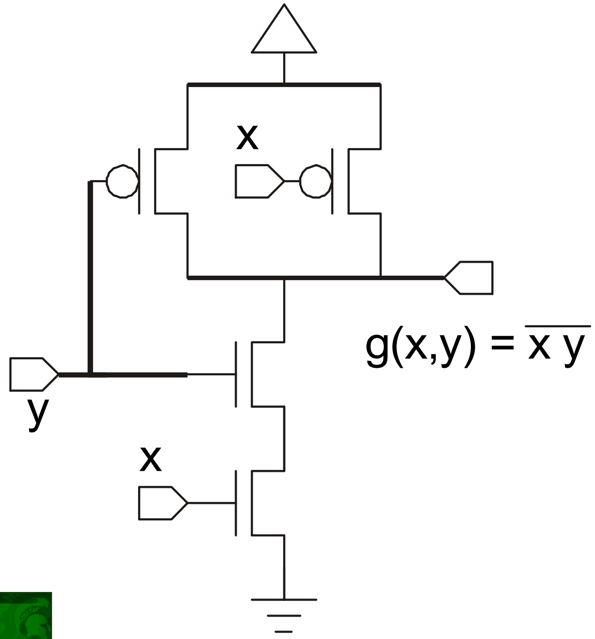
- Important Points
  - series-parallel arrangement
    - when nMOS in series, pMOS in parallel, and visa versa
    - true for all CMOS logic gates
    - allows us to construct more complex logic functions

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

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

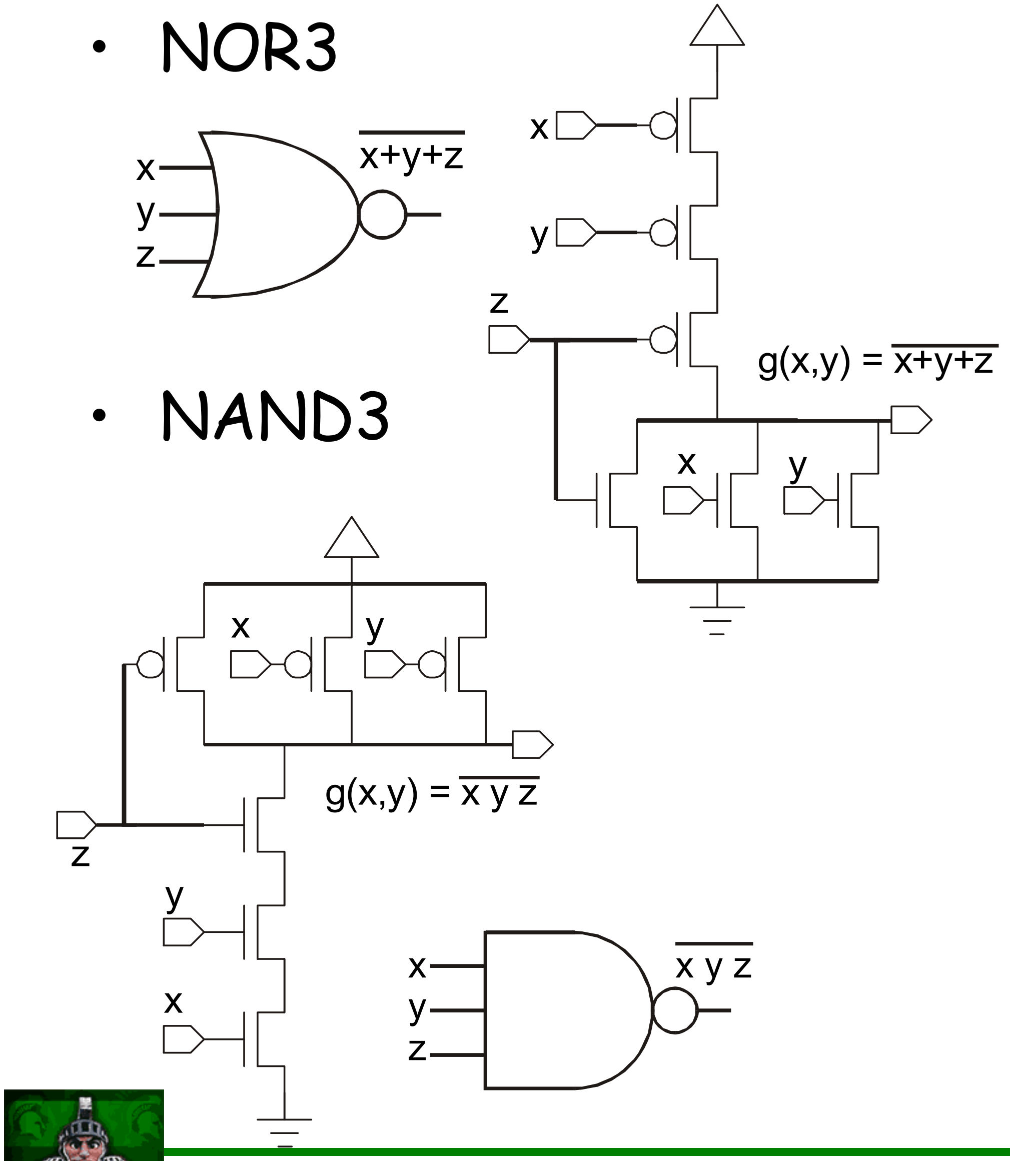
$$(x \bullet y \bullet 0)$$

$$= x.y.0 + \bar{x}.1 + \bar{y}.1$$

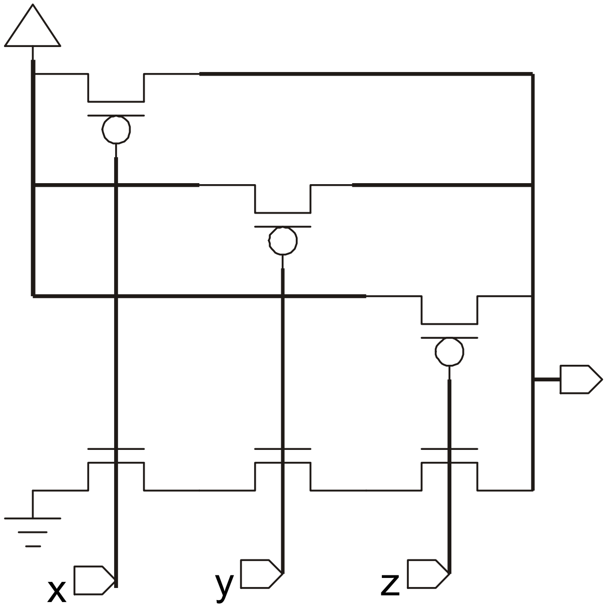
- **output is LOW if *x* AND *y* are true**
  - **series nMOS**
- **output is HIGH when *x* OR *y* is false**
  - **parallel pMOS**

3-Input Gates

- NOR3
- NAND3



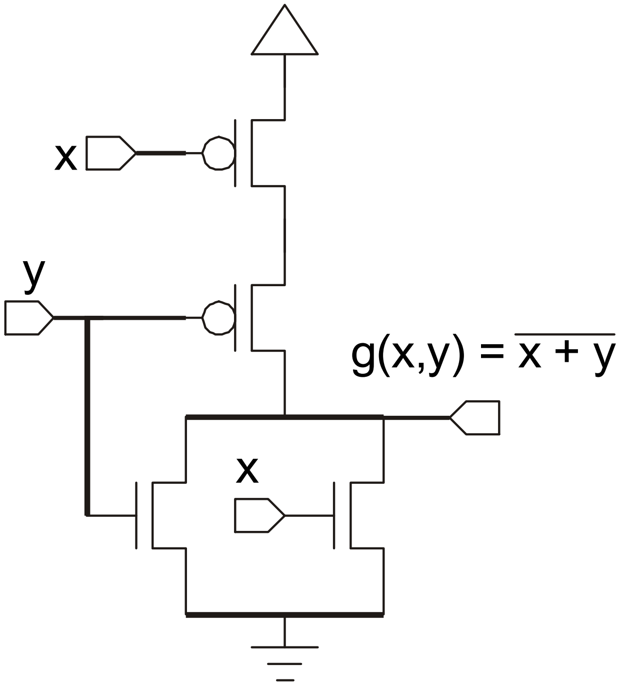
- Alternate Schematic
  - what function?



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

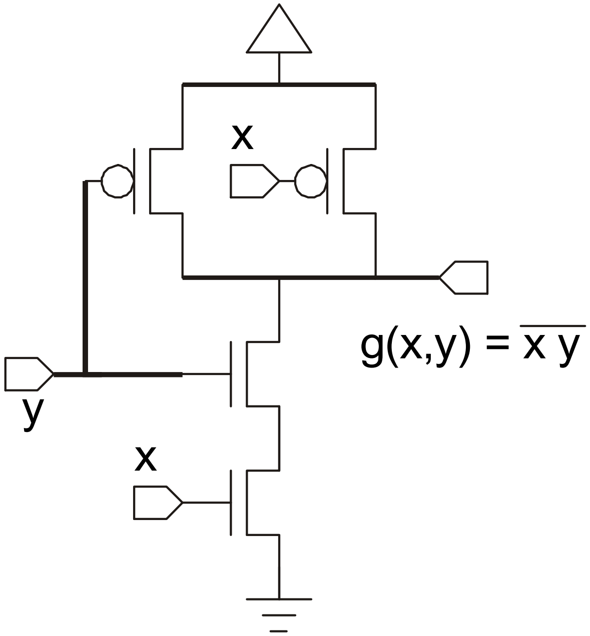
Review: CMOS NAND/NOR Gates

- 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

Complex Combinational Logic

- General logic functions
  - for example

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

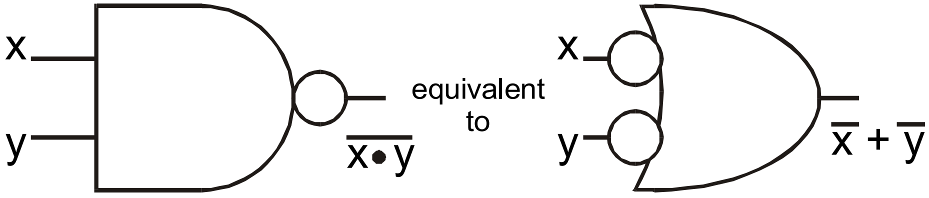
- How do we construct the CMOS gate?
  - use DeMorgan principles to modify expression
    - construct nMOS and pMOS networks

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

  - use Structured Logic
    - AOI (AND OR INV)
    - OAI (OR AND INV)

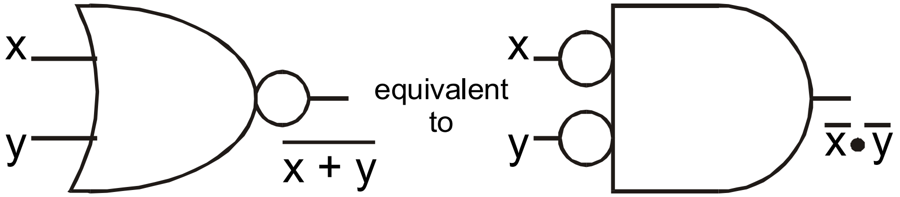
Using DeMorgan

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



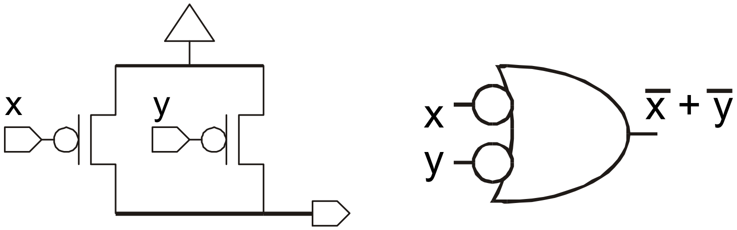
- bubbles = inversions

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



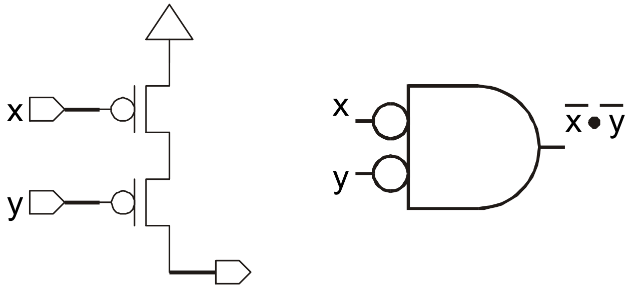
to implement pMOS this way, **must** push all bubbles to the inputs and remove all NAND/NOR output bubbles

- 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

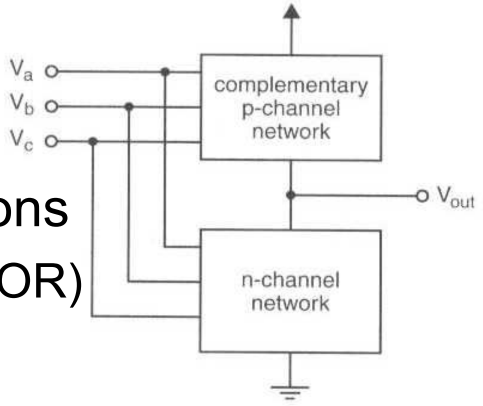


$g(x,y) = \overline{x}\ \overline{y} = \overline{x + y}$

    - assert-low AND
    - creates NOR function

Rules for Constructing CMOS Gates

The Mathematical Method



- Given a logic function

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

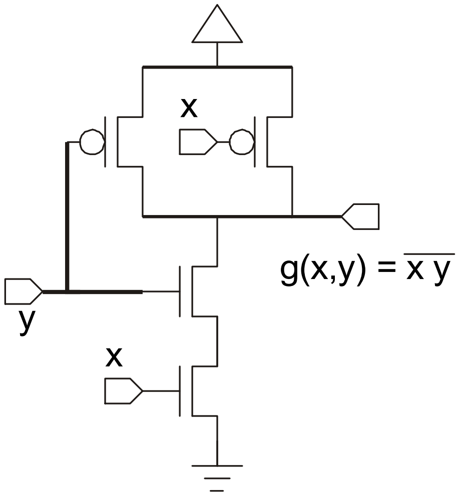
- Reduce (using DeMorgan) to eliminate inverted operations
  - inverted variables are OK, but not operations (NAND, NOR)
- Form pMOS network by complementing the **inputs**

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

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

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

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



EXAMPLE:

$F = \overline{ab} \Rightarrow$

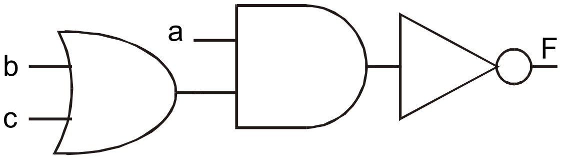
$Fp = \overline{\bar{a}\ \bar{b}} = a+b;$ OR/parallel

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

CMOS Combinational Logic Example

- Construct a CMOS logic gate to implement the function:

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



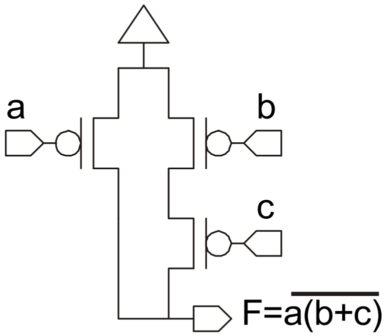
14 transistors (cascaded gates)

- pMOS
  - Apply DeMorgan expansions

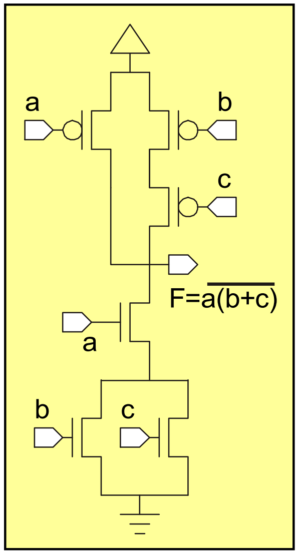
    $F = \overline{a} + \overline{(b + c)}$

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

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



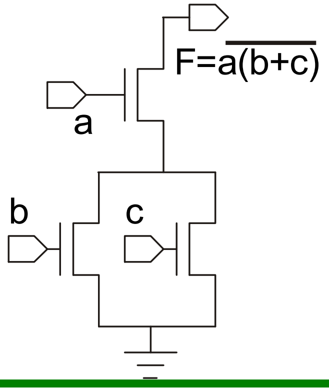
6 transistors (CMOS)



- 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 use structured circuits to implement general logic functions
- AOI: implements logic function in the order

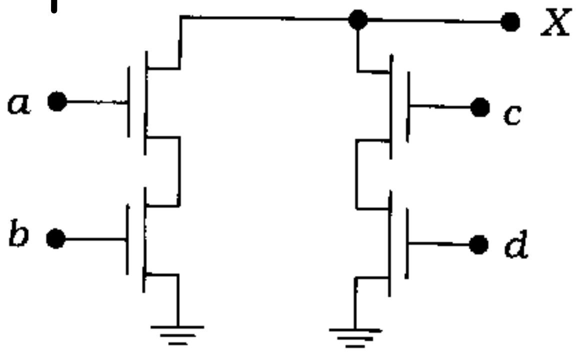
  AND, OR, NOT (Invert)
  - Example: $F = \overline{a \cdot b + c \cdot d}$
    - operation order: i) a AND b, c AND d, ii) (ab) OR (cd), iii) NOT
  - Inverted Sum-of-Products (SOP) form
- OAI: implements logic function in the order

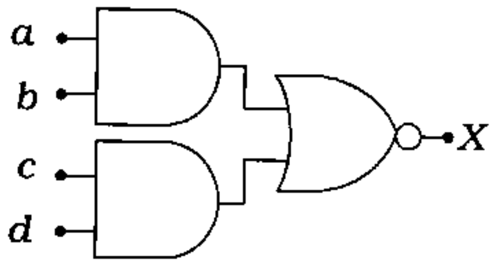
  OR, AND, NOT (Invert)
  - Example: $G = \overline{(x+y) \cdot (z+w)}$
    - operation order: i) x OR y, z OR w, ii) (x+y) AND (z+w), iii) NOT
  - Inverted Product-of-Sums (POS) form
- Use a ***structured CMOS array*** to realize such functions

AOI/OAI nMOS Circuits

- nMOS AOI structure
  - series txs in parallel

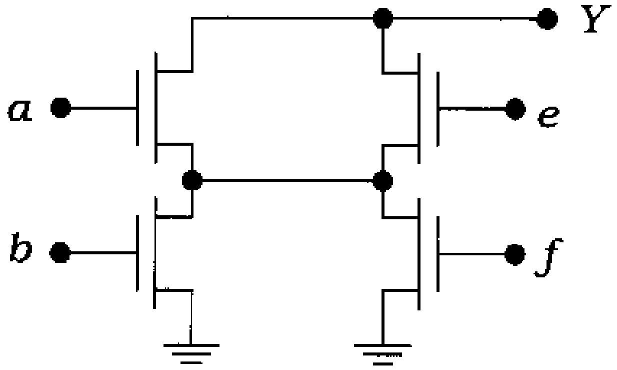
$F = \overline{a \cdot b + c \cdot d}$

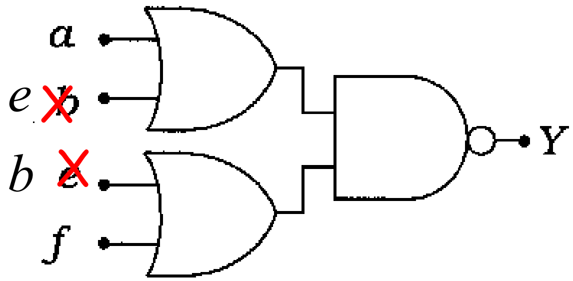




- nMOS OAI structure
  - series of parallel txs

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

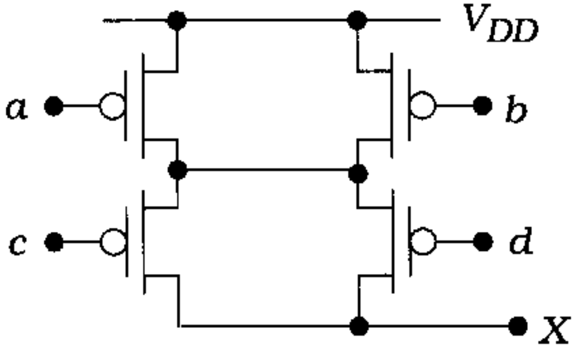




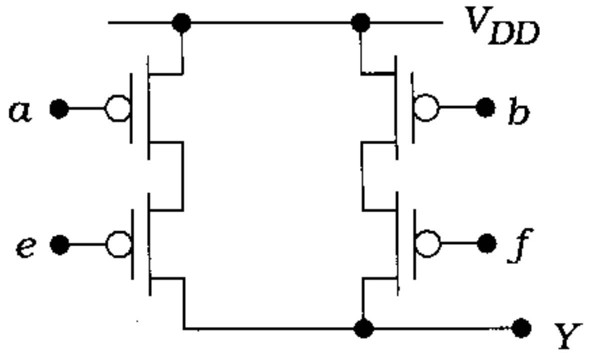
error in textbook Figure 2.45

AOI/OAI pMOS Circuits

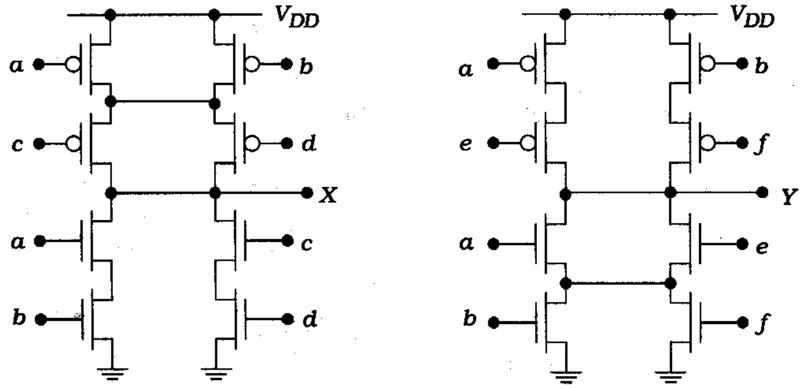
- pMOS AOI structure
  - series of parallel txs
  - opposite of nMOS (series/parallel)



- 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

    # txs =

    3 operations: 1 AND, 2 OR

    # 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

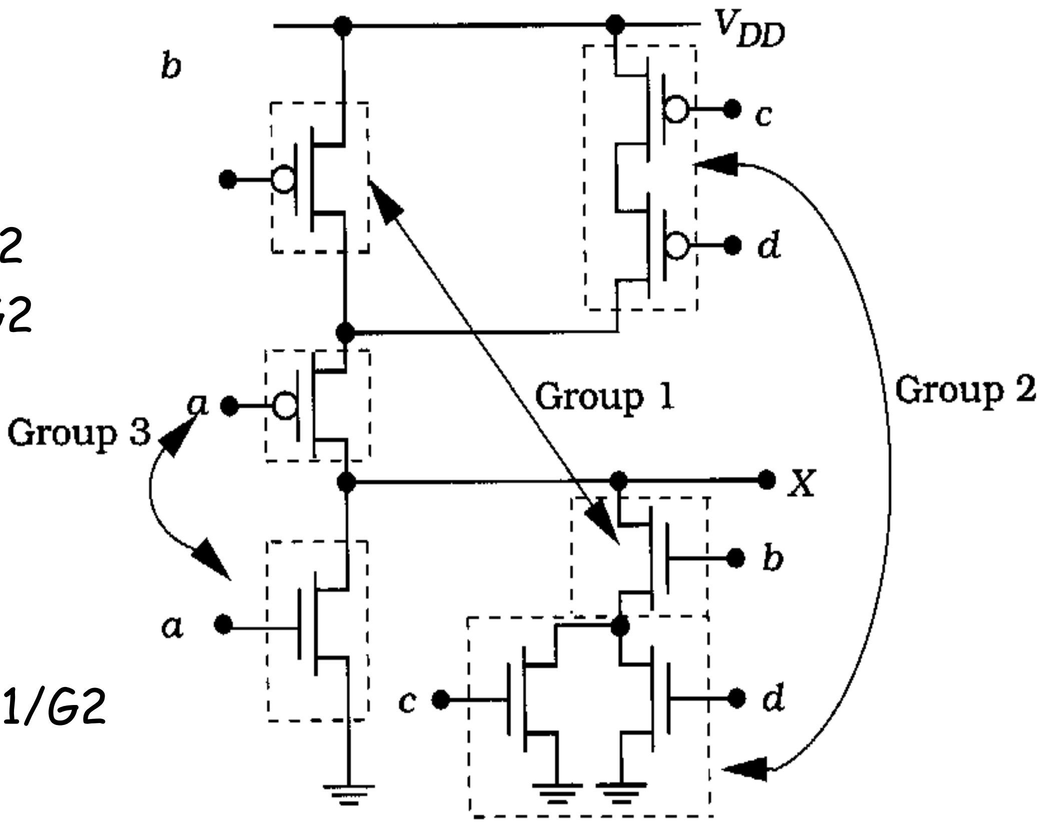
- Construct the function below in CMOS

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

  $Fn = a + b \cdot (c + d)$

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

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

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

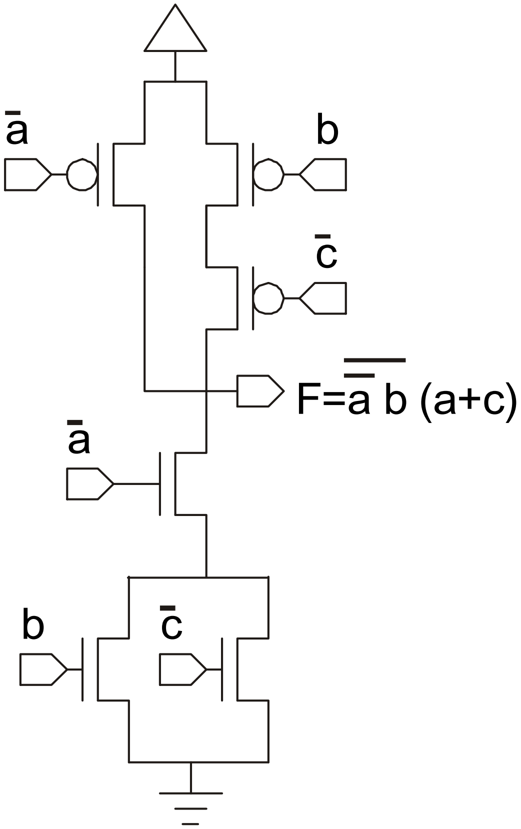
    $Fn = a + (\overline{b} \cdot c)$
  - Resulting Schematic ?

Yet Another Combinational Logic Example

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

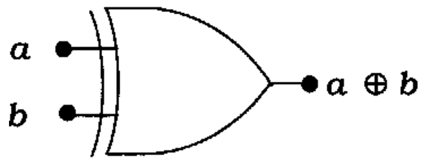
  $$F = \overline{\bar{a} \cdot b} \cdot (a + c)$$

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



XOR and XNOR

- Exclusive-OR (XOR)
  - $a \oplus b = \overline{a} \cdot b + a \cdot \overline{b}$
  - not AOI form



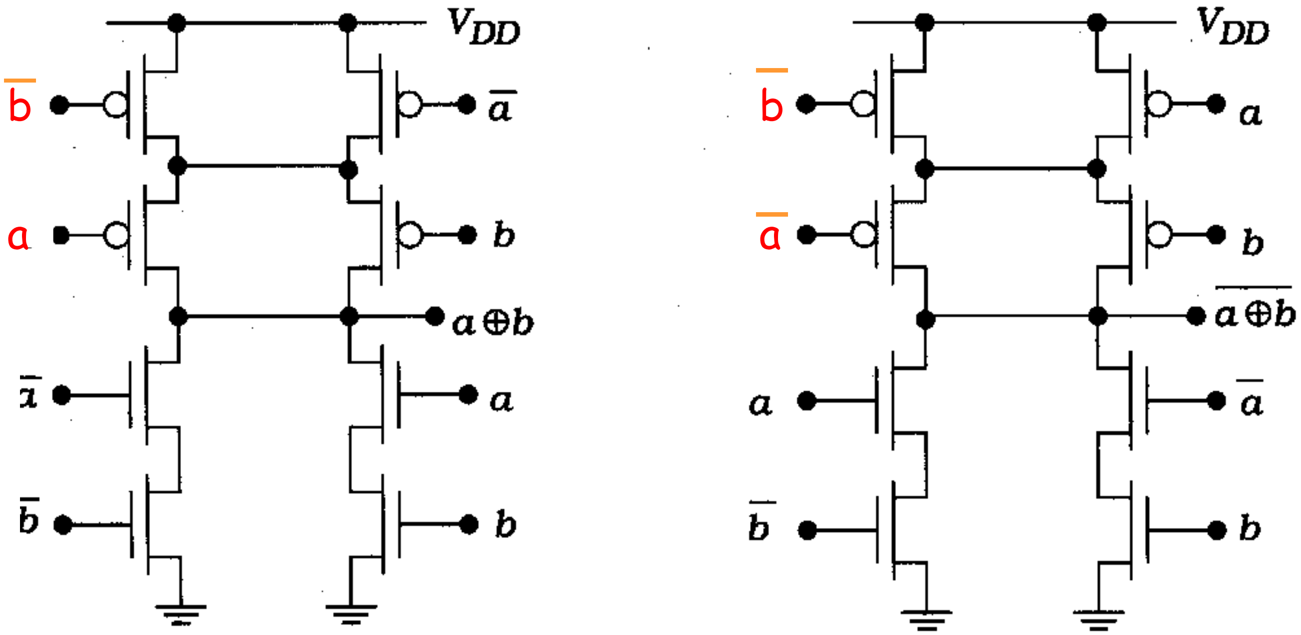
a	b	a ⊕ b
0	0	0
0	1	1
1	0	1
1	1	0

- 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: see textbook, figure 2.57

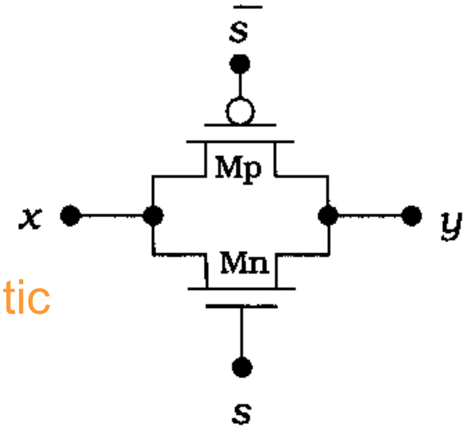
-XOR: $a \oplus b = \overline{a \cdot b + \overline{a} \cdot \overline{b}}$

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

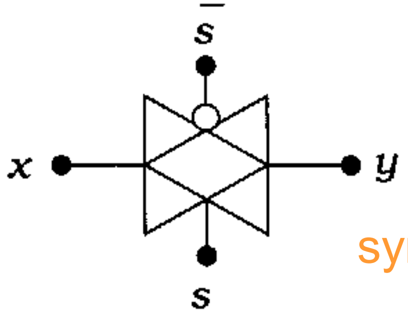
CMOS Transmission Gates

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**

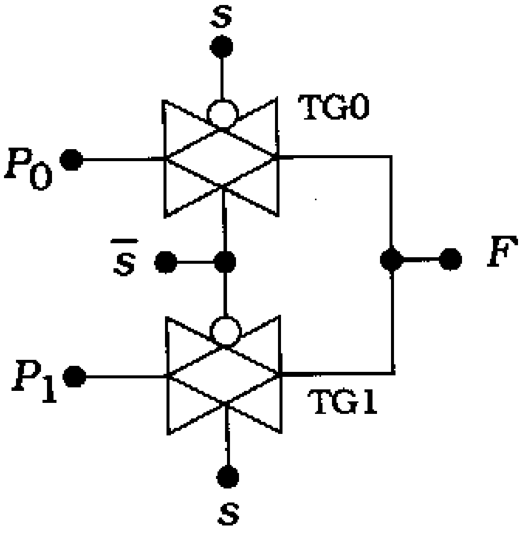
y = x s, for s=1

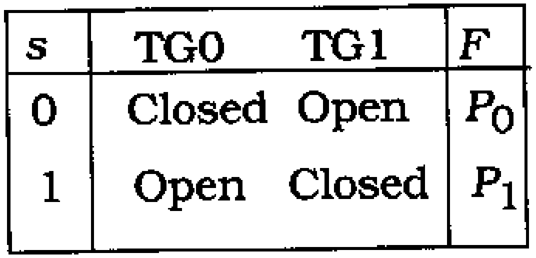
Transmission Gate Logic Functions

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

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

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

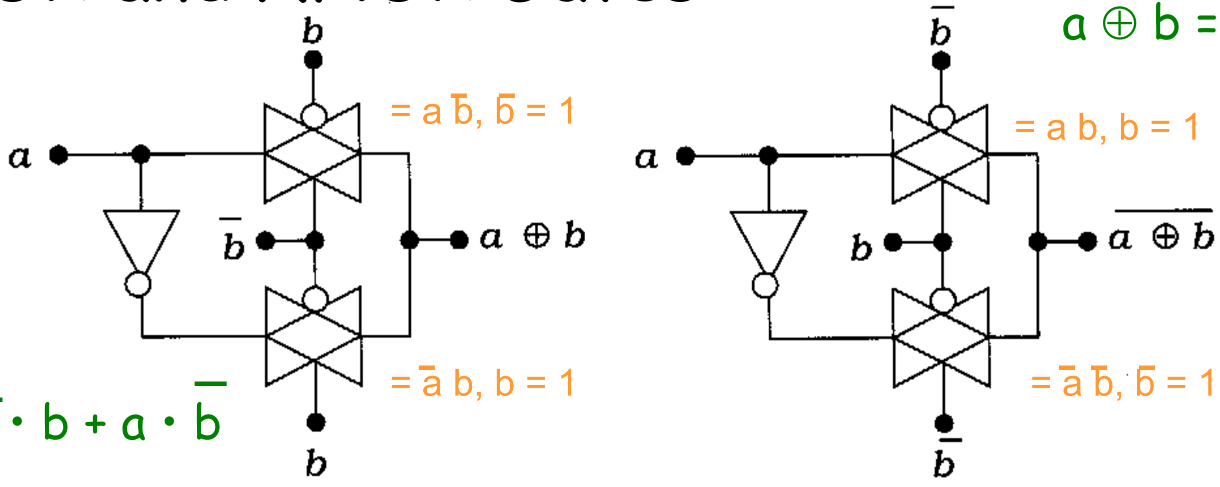




s	TG0	TG1	F
0	Closed	Open	P_0
1	Open	Closed	P_1

More TG Functions

- TG XOR and XNOR Gates



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

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

$= ab,\ b = 1$

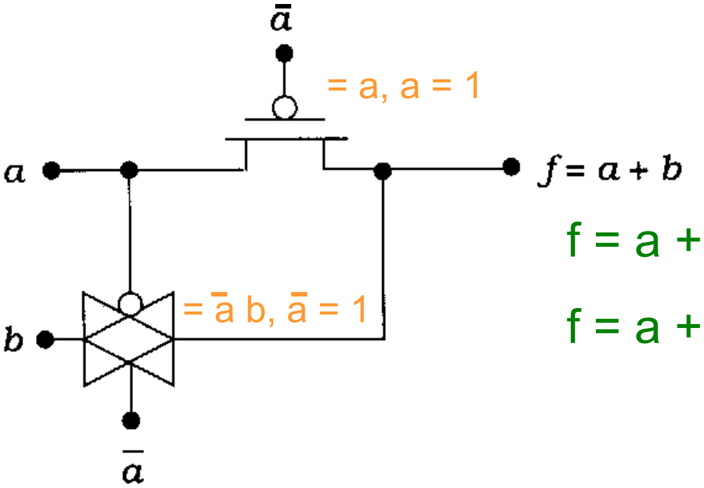
$= \bar{a}\bar{b},\ \bar{b} = 1$

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

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

(a) XOR circuit (b) XNOR circuit

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



$= a,\ a = 1$

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

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

$f = a + b$

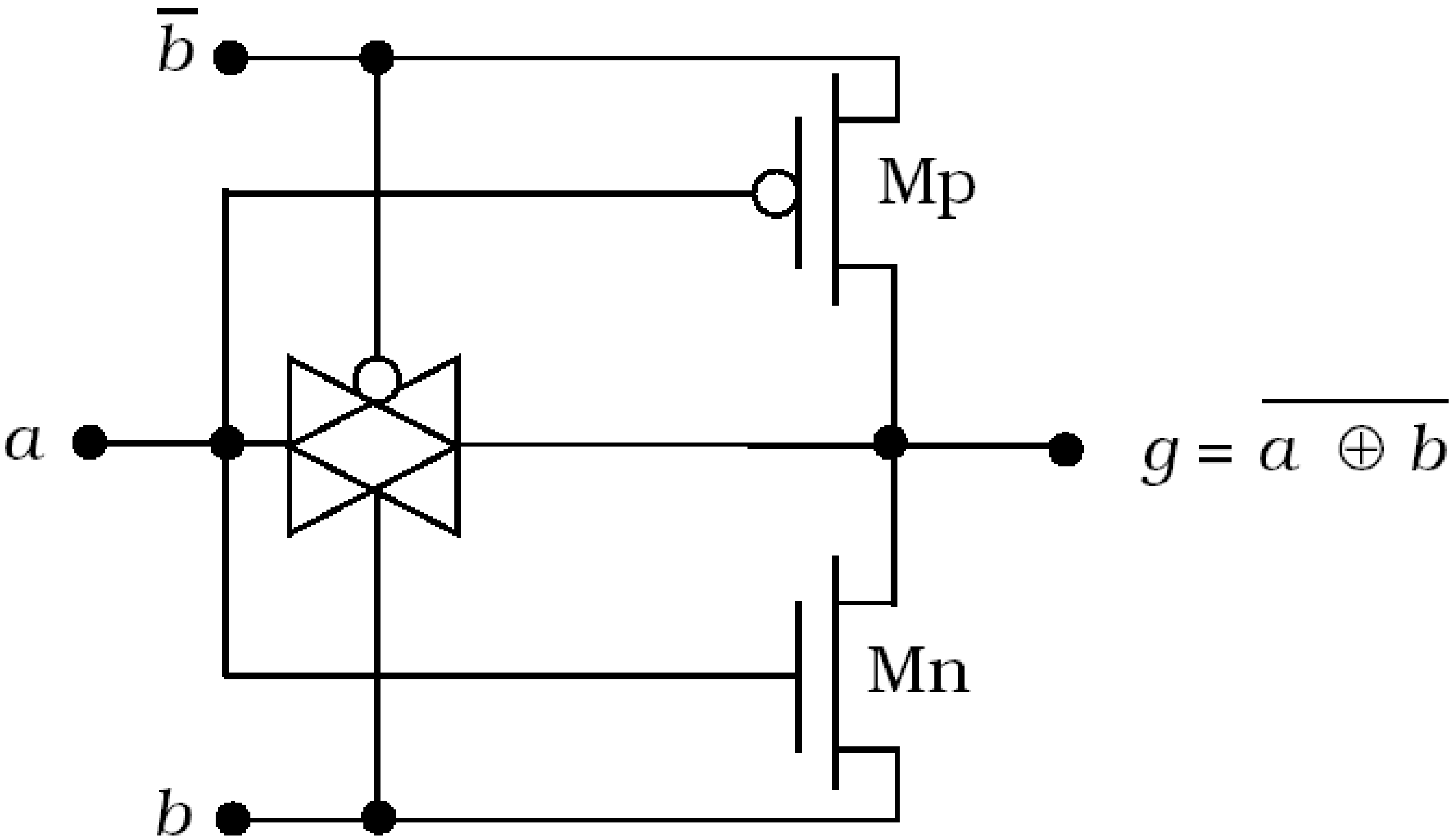


Figure 2.64 (p. 59)

An XNOR gate that uses both TGs and FETs.

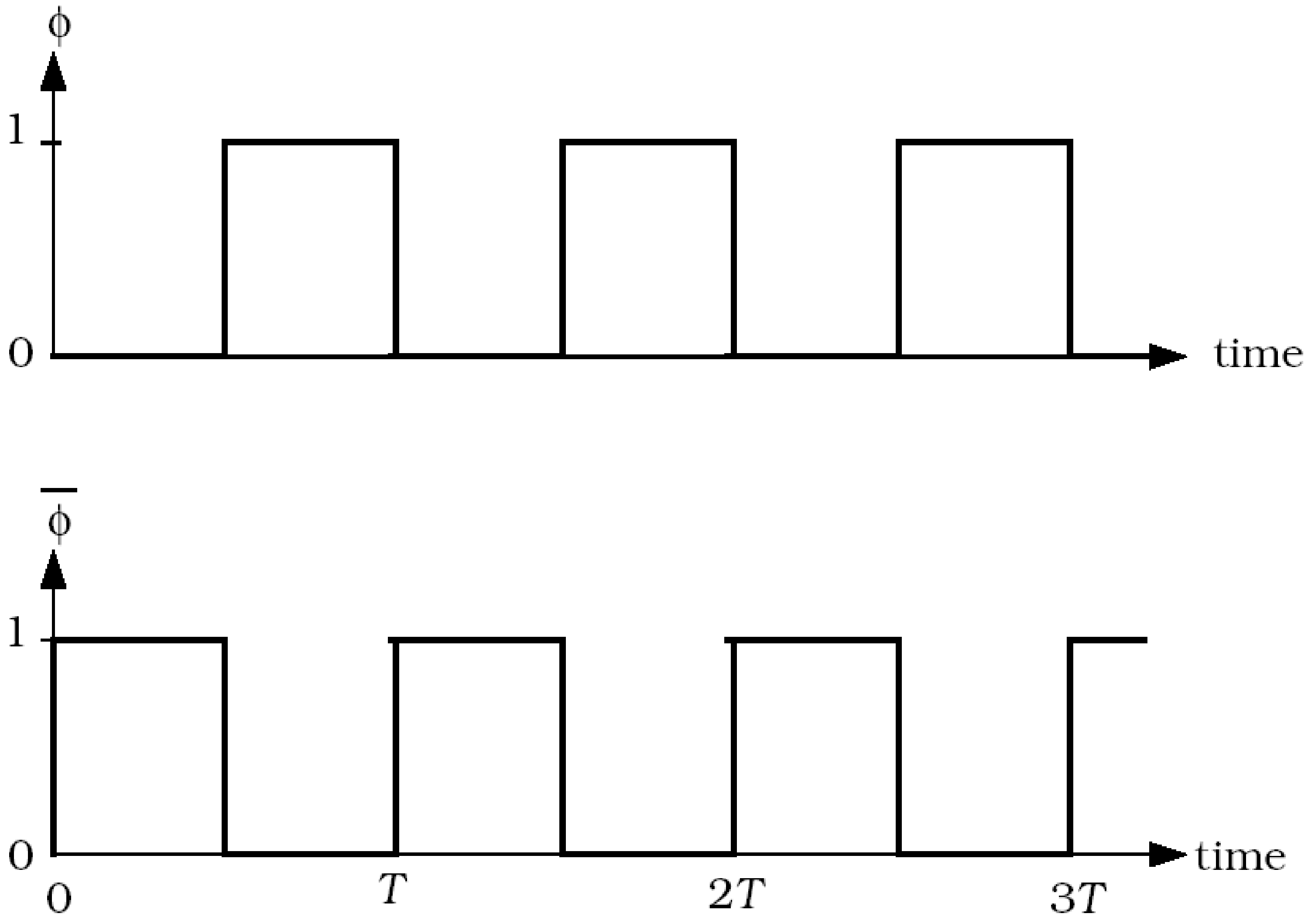


Figure 2.65 (p. 60)

Complementary clocking signals.

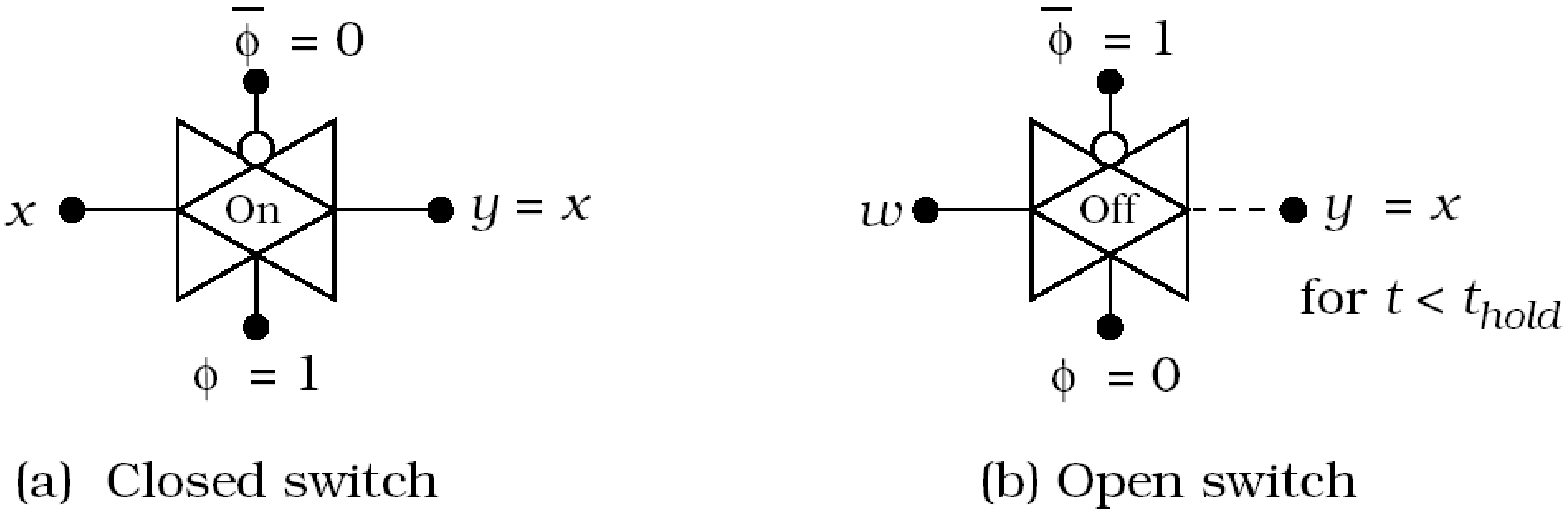


Figure 2.66 (p. 61)

Behavior of a clocked TG.

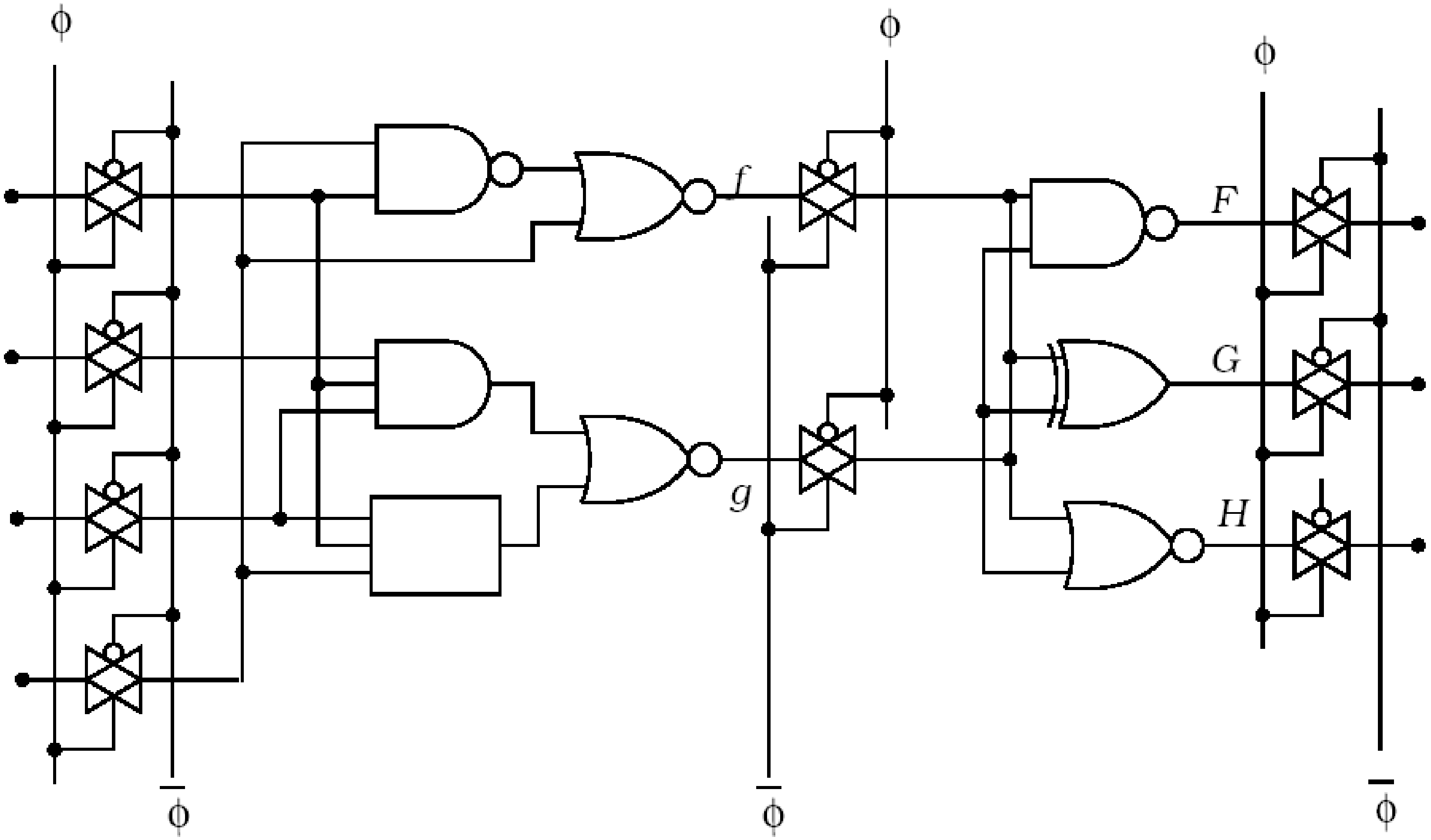


Figure 2.67 (p. 61)

Data synchronization using transmission gates.

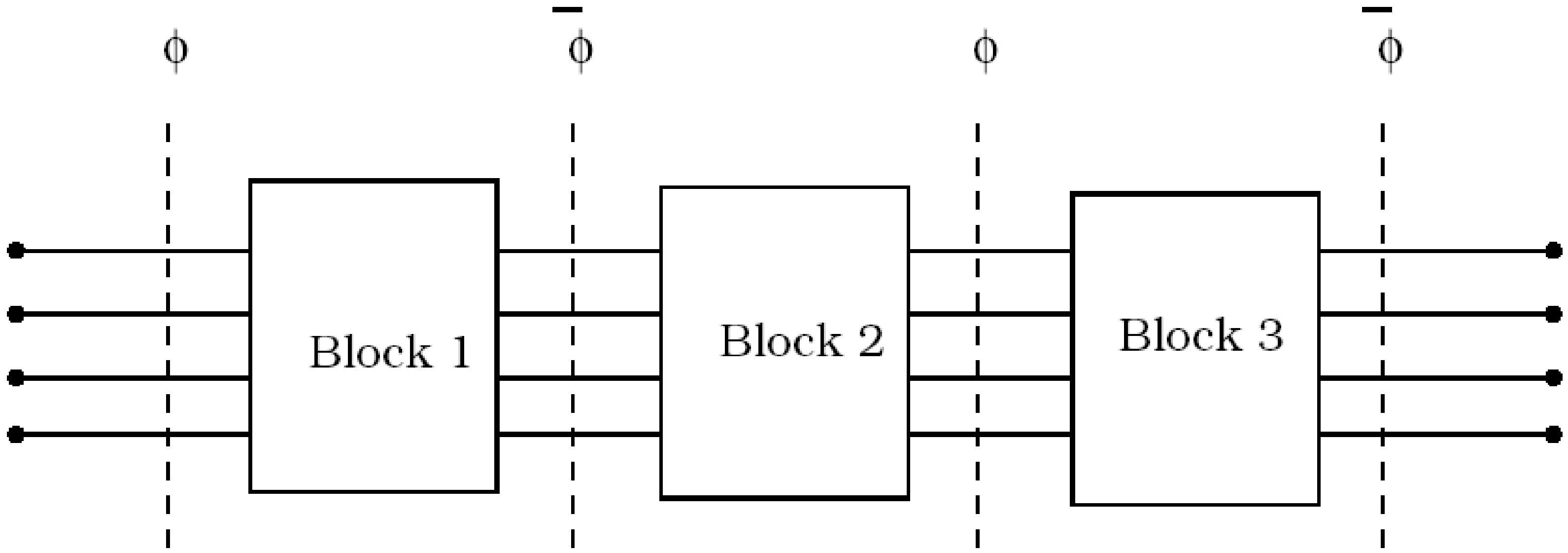
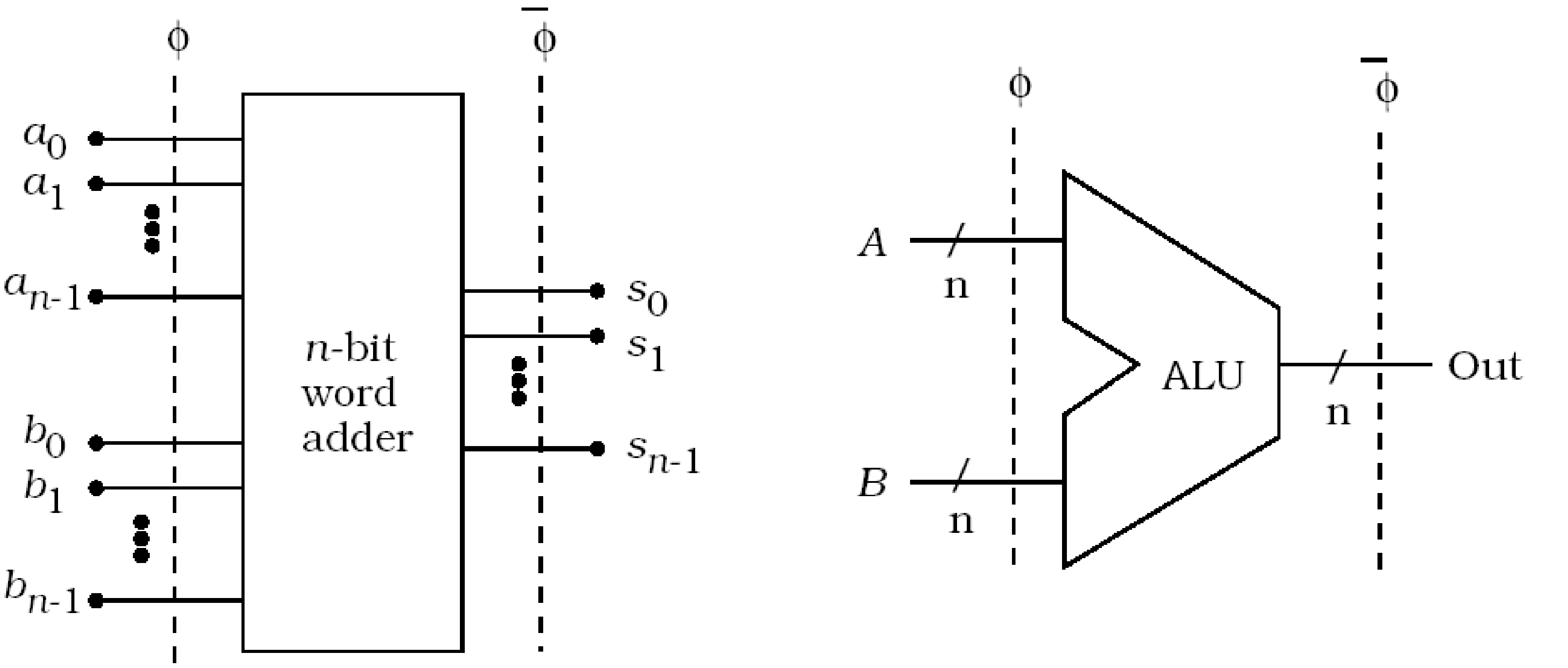


Figure 2.68 (p. 62)

Block-level system timing diagram.



(a) Clocked adder

(b) Clocked ALU

Figure 2.69 (p. 62)

Control of binary words using clocking planes.