BIRZEIT UNIVERSITY

DEPARTMENT OF COMPUTER SYSTEM ENGINEERING

Digital Integrated Circuits - ENCS333

Dr. Khader Mohammad

Lecture #8 Combinational Circuit

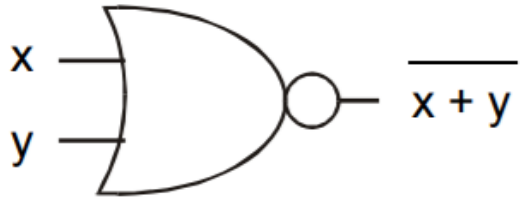
Integrated-Circuit Devices and Modeling

STUDENTS-HUB.com

Uploaded By: anonymous

CMOS NOR Gate

- NOR Symbol



- Karnaugh map

x \ y	0	1
0	1	0
1	0	0

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

- NOR Truth Table

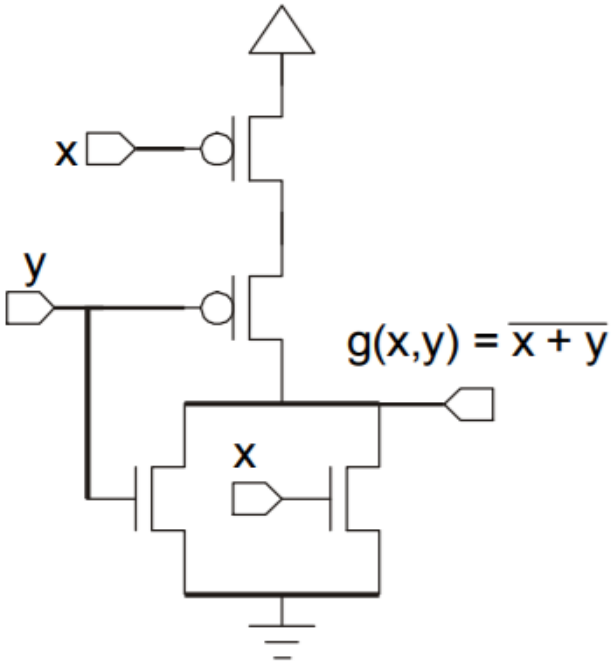
x	y	$\overline{x+y}$
0	0	1
0	1	0
1	0	0
1	1	0

- 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

STUDENTS-HUB.com Uploaded By: anonymous

CMOS NOR Gate

- CMOS NOR Schematic



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

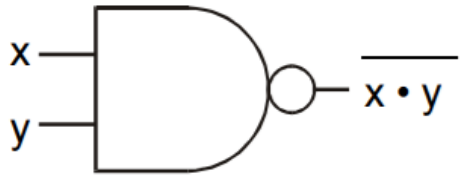
- **output is LOW if *x* OR *y* is true**
  - **parallel nMOS**
- **output is HIGH when *x* AND *y* are false**
  - **series pMOS**

- 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

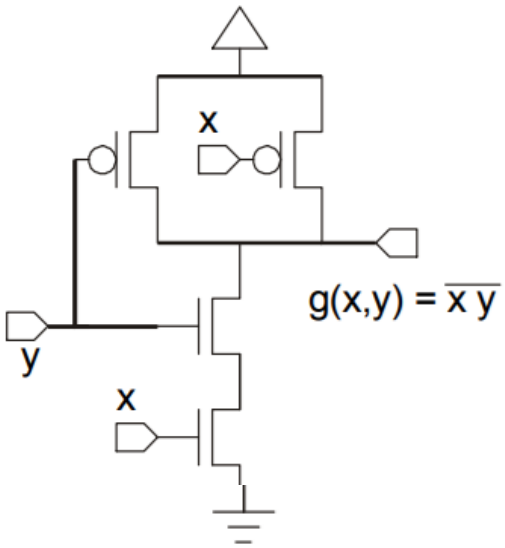
STUDENTS-HUB.com Uploaded By: anonymous

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

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

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

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

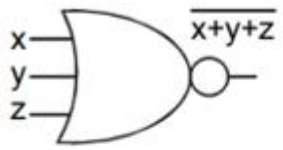
- **output is LOW if *x* AND *y* are true**
  - **series nMOS**
- **output is HIGH when *x* OR *y* is false**
  - **parallel pMOS**

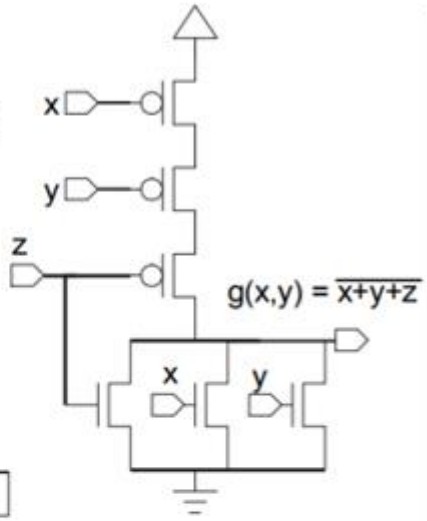
STUDENTS-HUB.com

Uploaded By: anonymous

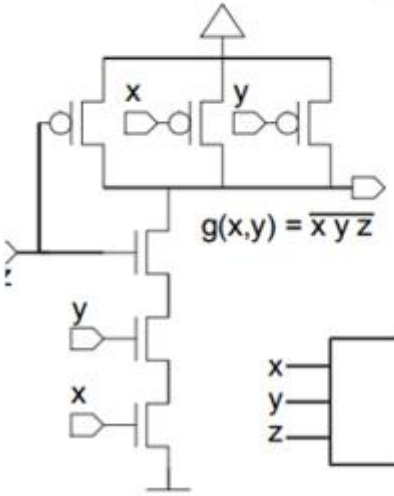
3-Input Gates

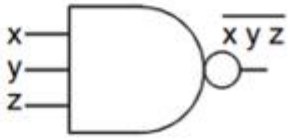
- NOR3



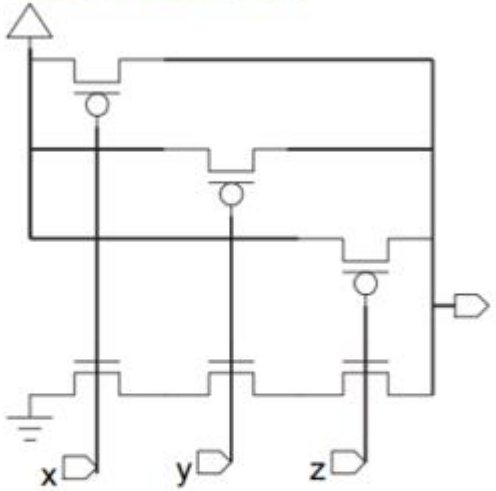


- NAND3





- Alternate Schematic
  - what function?

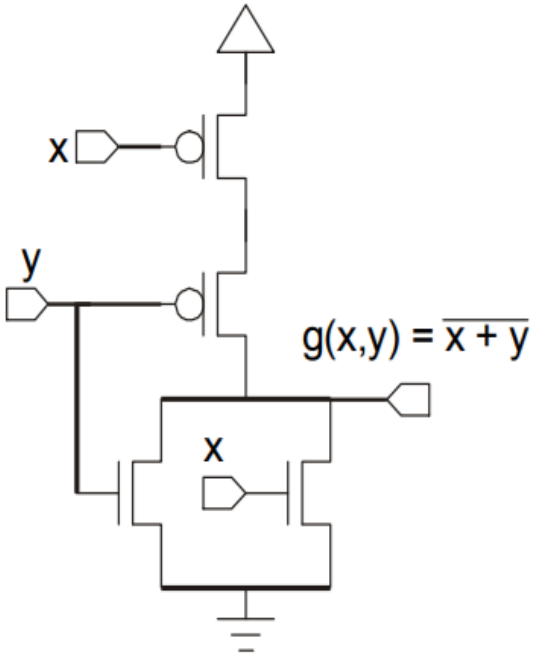


- note shared gate inputs
  - is input order important?
  - in series, parallel, both?
- schematic resembles how the circuit will look in *physical layout*

STUDENTS-HUB.com
Uploaded By: anonymous

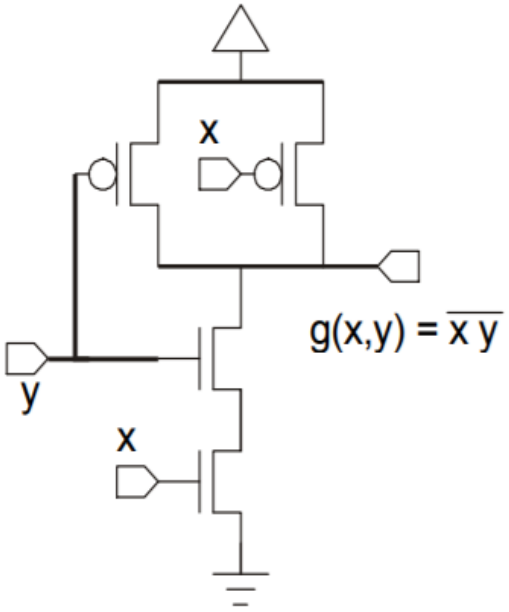
Review: CMOS NAND/NOR Gates

- 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

STUDENTS-HUB.com
Uploaded By: anonymous

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

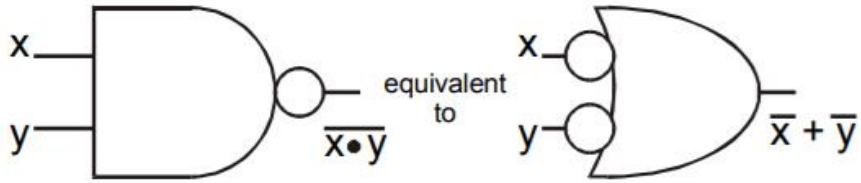
$$\overline{a \cdot b} = \overline{a} + \overline{b} \qquad \overline{a + b} = \overline{a} \cdot \overline{b}$$

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

STUDENTS-HUB.com Uploaded By: anonymous

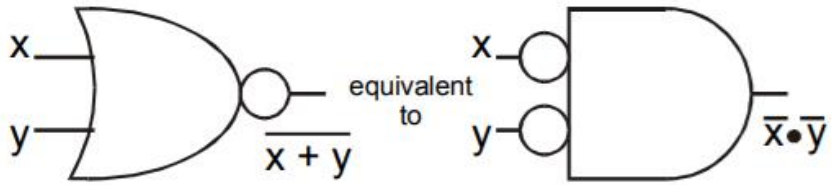
Using DeMorgan

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



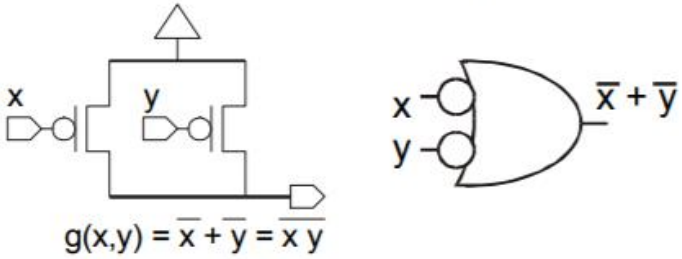
- bubbles = inversions

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

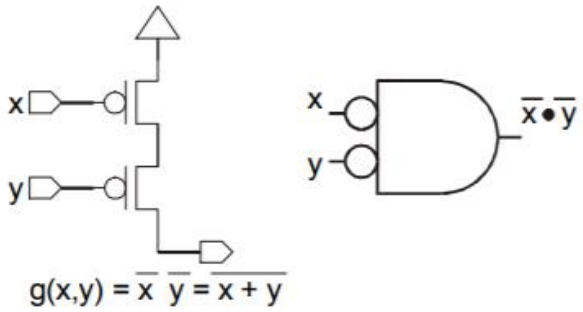


to implement pMOS this way, must push all bubbles to the inputs and remove all NAND/NOR output bubbles

- pMOS and bubble pushing
  - Parallel-connected pMOS



    - assert-low OR
    - creates NAND function
  - Series-connected pMOS



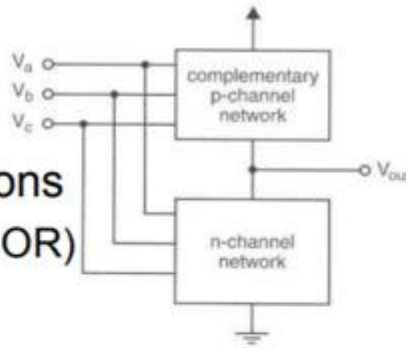
    - assert-low AND
    - creates NOR function

STUDENTS-HUB.com
Uploaded By: anonymous

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

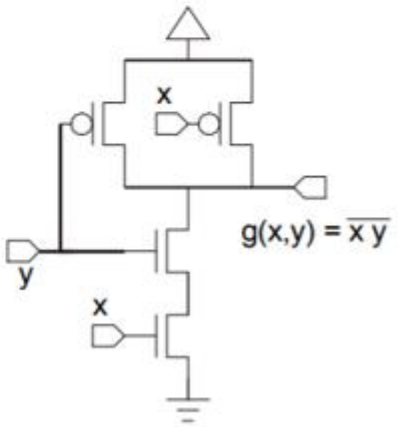


EXAMPLE:

$F = \overline{ab} \Rightarrow$

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

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

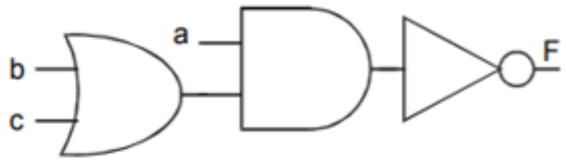


STUDENTS-HUB.com Uploaded By: anonymous

CMOS Combinational Logic Example

- Construct a CMOS logic gate to implement the function:

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



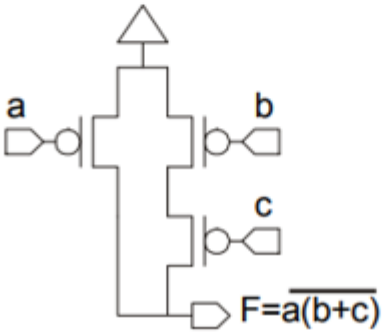
14 transistors (cascaded gates)

- pMOS
  - Apply DeMorgan expansions

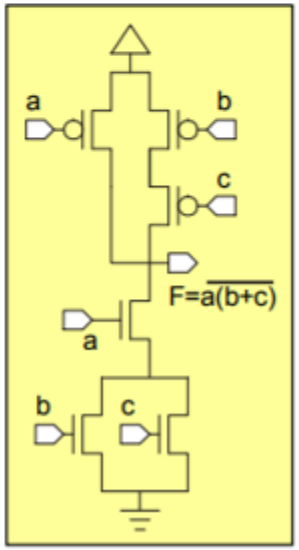
    $F = \overline{a} + \overline{(b + c)}$

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

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



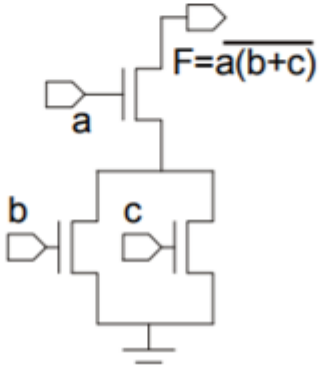
6 transistors (CMOS)



- nMOS
  - Invert output for nMOS

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

    none needed
  - Resulting Schematic



STUDENTS-HUB.com
Uploaded By: anonymous

STUDENTS-HUB.com Uploaded By: anonymous

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

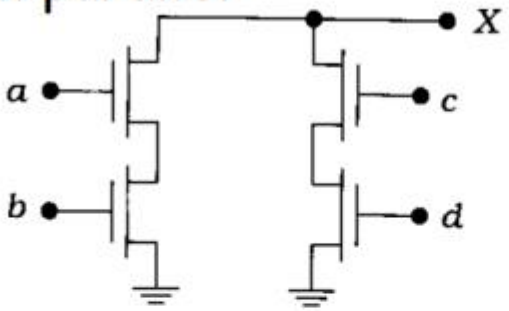
  Use a ***structured CMOS array*** to realize such functions

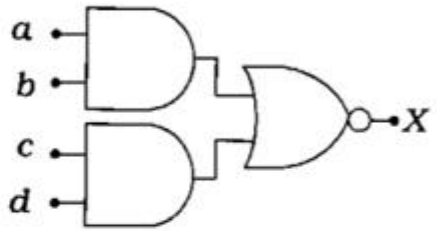
STUDENTS-HUB.com Uploaded By: anonymous

AOI/OAI nMOS Circuits

- nMOS AOI structure
  - series txs in parallel

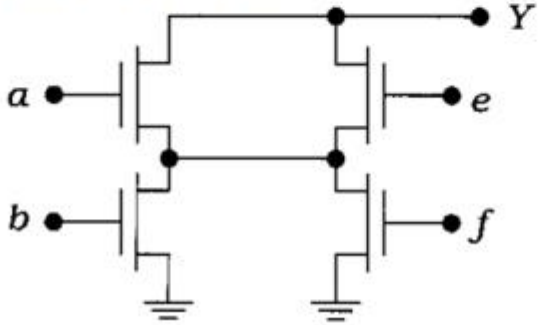
$$F = \overline{a \cdot b + c \cdot d}$$

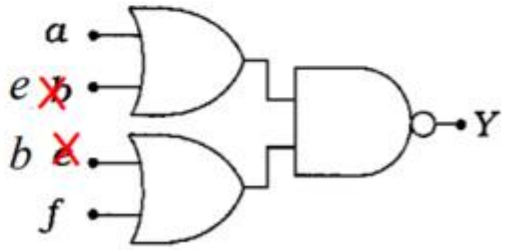




- nMOS OAI structure
  - series of parallel txs

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



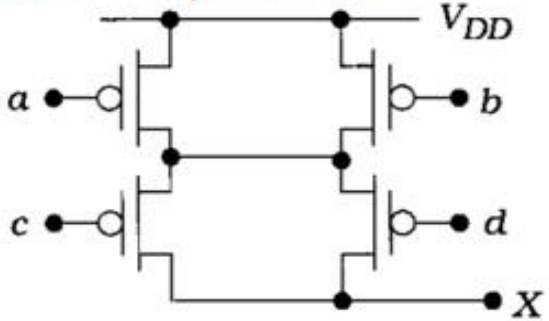


error in textbook Figure 2.45

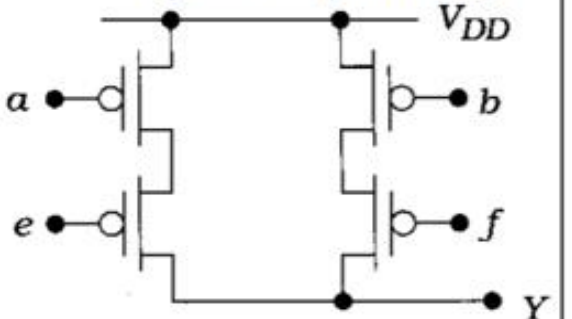
STUDENTS-HUB.com
Uploaded By: anonymous

AOI/OAI pMOS Circuits

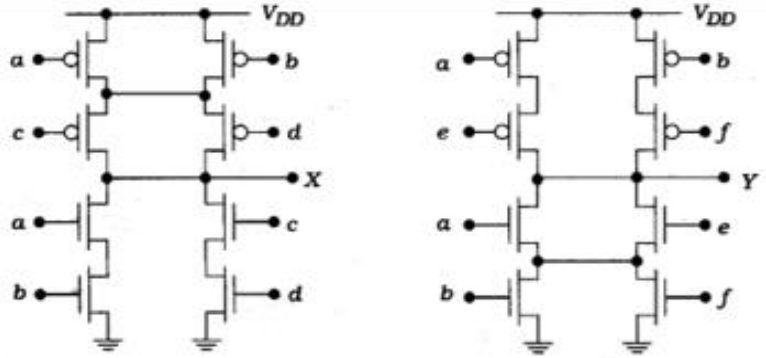
- pMOS AOI structure
  - series of parallel txs
  - opposite of nMOS (series/parallel)



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



Complete CMOS AOI/OAI circuits



STUDENTS-HUB.com Uploaded By: anonymous

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 operations:
3 AND, 2 OR	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

STUDENTS-HUB.com Uploaded By: anonymous

CMOS Logic Example

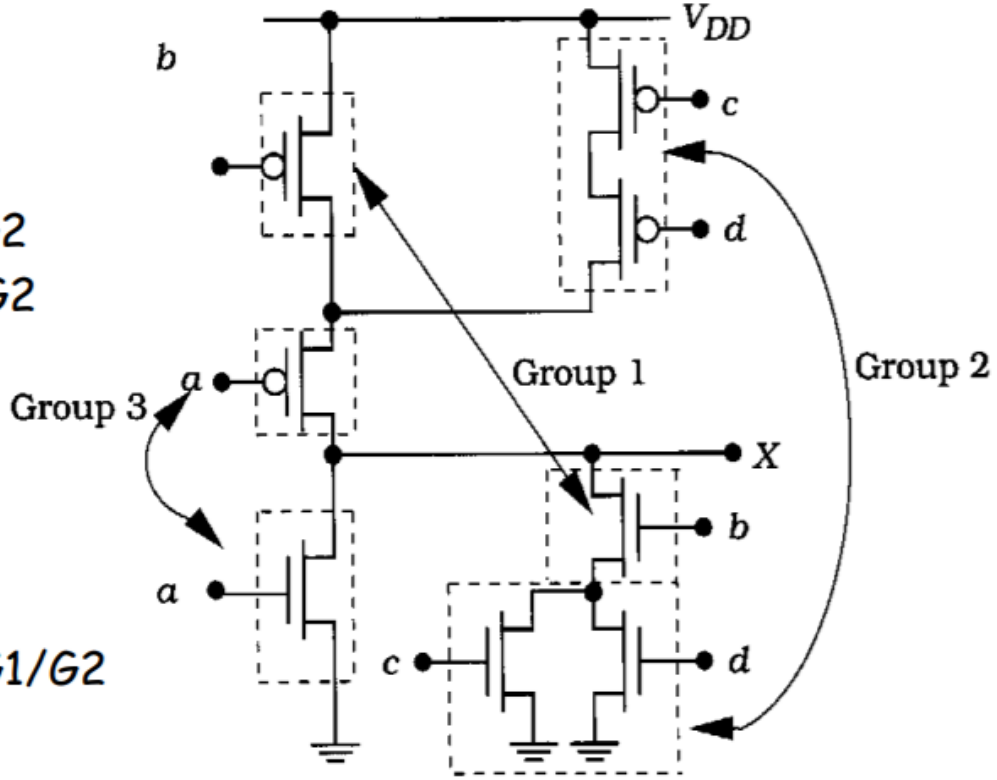
- Construct the function below in CMOS

  $F = a + b \cdot (c + d)$; 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)

STUDENTS-HUB.com
Uploaded By: anonymous

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

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

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

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

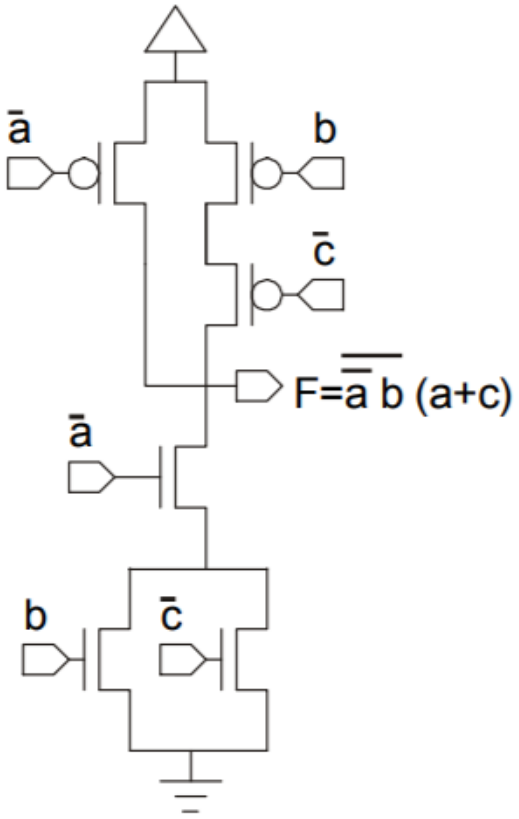
STUDENTS-HUB.com Uploaded By: anonymous

Another Combinational Logic Example

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

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

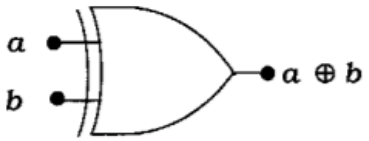
- nMOS
  - Invert Output
    - $Fn = \overline{\overline{a} \cdot b \cdot (a + c)} = \overline{a} \cdot b + \overline{(a + c)}$
  - Eliminate NANDs and NORs
    - $Fn = \overline{a} \cdot b + (\overline{a} \cdot \overline{c})$
  - Reduce Function
    - $Fn = \overline{a} \cdot (b + \overline{c})$
  -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



STUDENTS-HUB.com
Uploaded By: anonymous

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

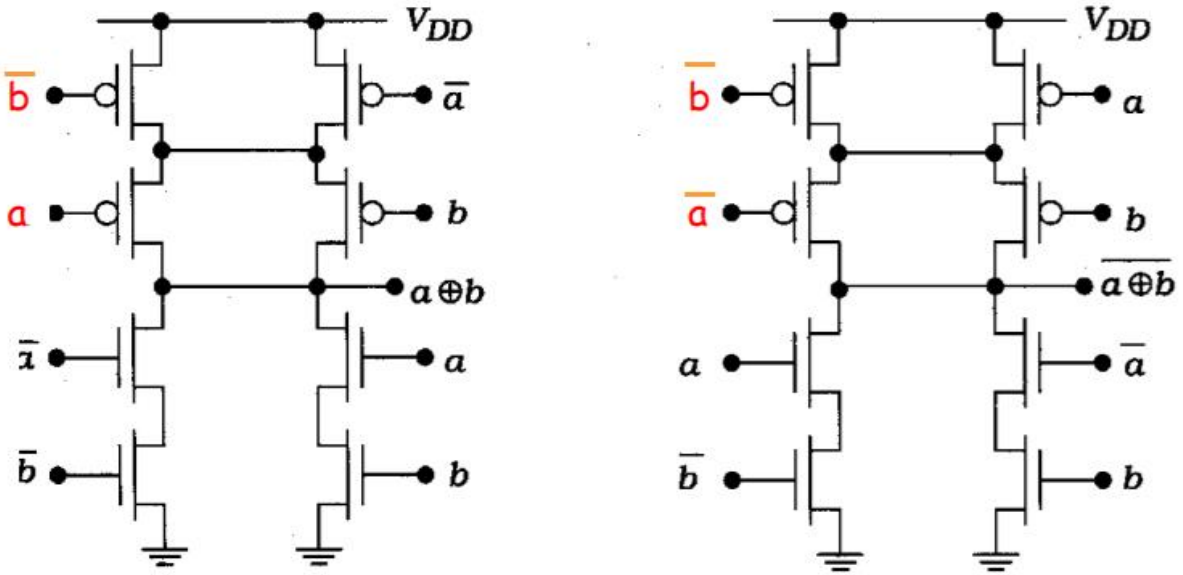
- Exclusive-NOR
  - $\overline{a \oplus b} = a \cdot b + \overline{a} \cdot \overline{b}$
  - inverse of XOR

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

STUDENTS-HUB.com
Uploaded By: anonymous

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

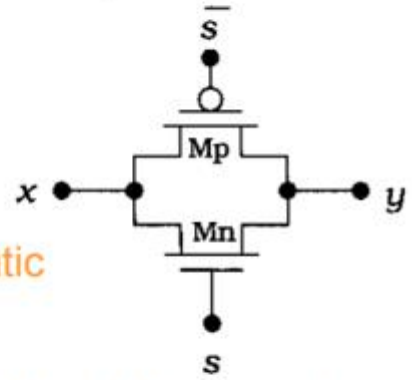
STUDENTS-HUB.com Uploaded By: anonymous

CMOS Transmission Gates

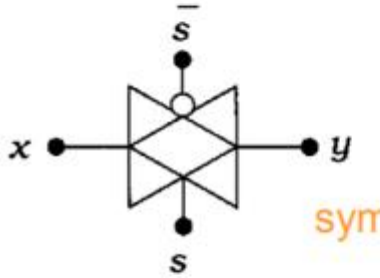
- Function

recall: pMOS passes a good '1' and nMOS passes a good '0'

  - 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 $\overline{s}$
  - 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

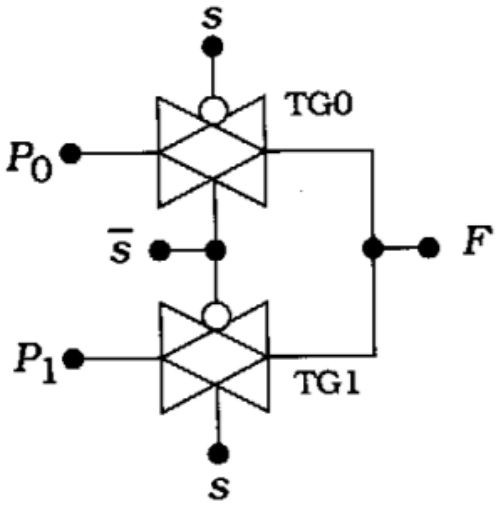
STUDENTS-HUB.com Uploaded By: anonymous

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

  $F = P_0 \cdot \bar{s} + P_1 \cdot s$

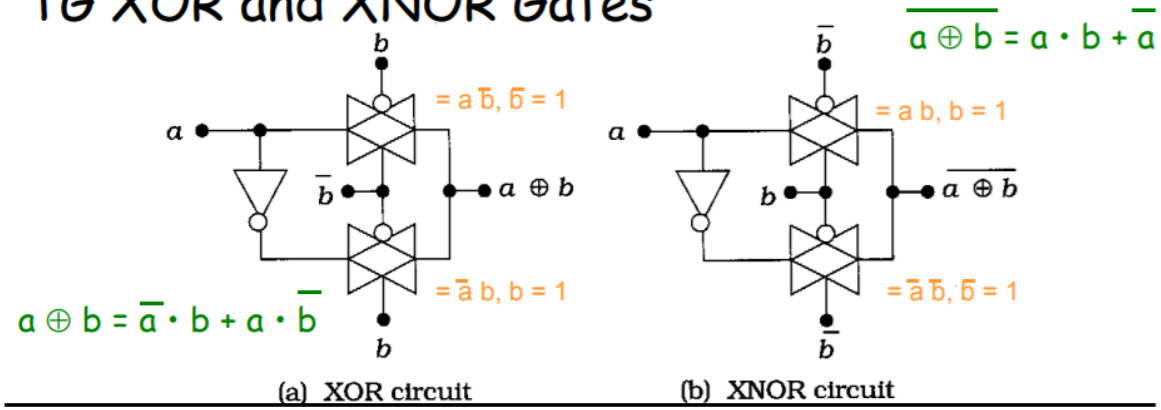


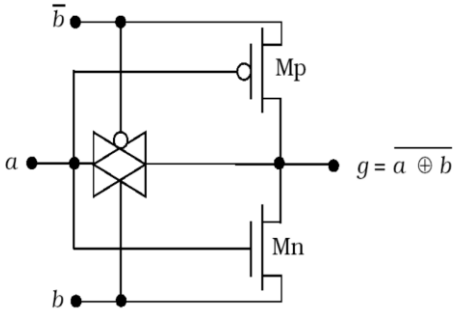
s	TG0	TG1	F
0	Closed	Open	P_0
1	Open	Closed	P_1

STUDENTS-HUB.com
Uploaded By: anonymous

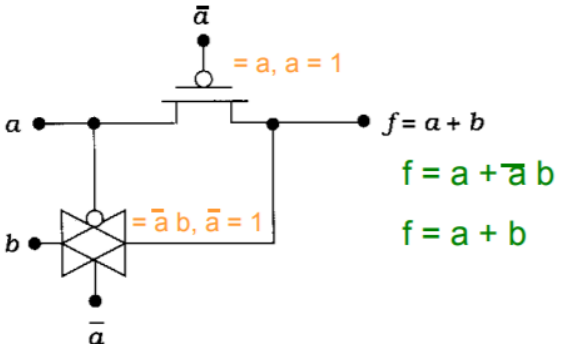
More TG Functions

- TG XOR and XNOR Gates





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



STUDENTS-HUB.com
Uploaded By: anonymous

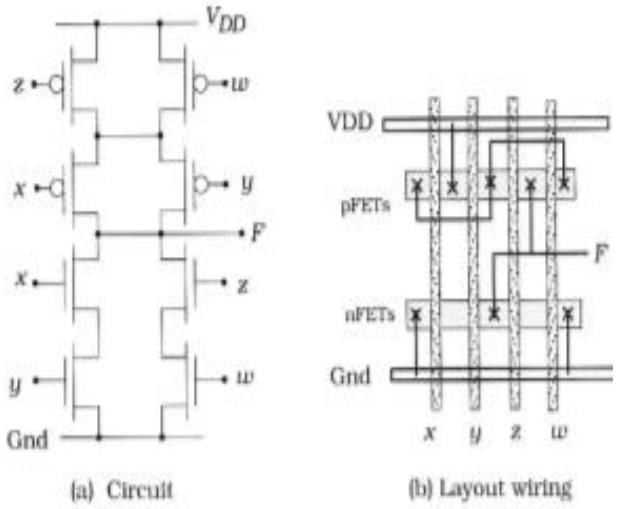
Structured Layout

General Approach
- – power rails
- – horizontal Active
- – vertical Poly (inputs from top/bottom)
- – Metal1 connects nodes as needed in schematic

• Structured Layout
- – AOI circuit figure
- – useful for many logic functions
- – see examples in textbook

• Disadvantages
- – not optimized for speed
  - • large S/D regions = higher capacitance
  - • interconnect paths could be shorter

– not optimized for area/size

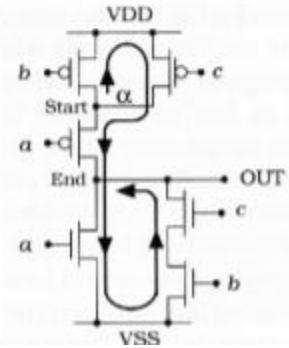


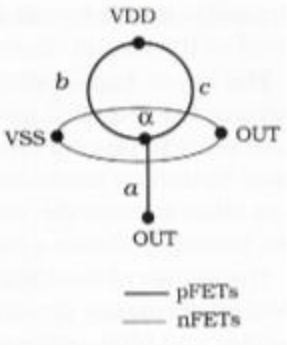
notice, need room inside cell (between VDD and Ground) to route internal connections

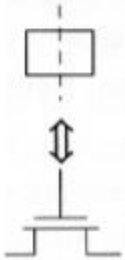
STUDENTS-HUB.com Uploaded By: anonymous

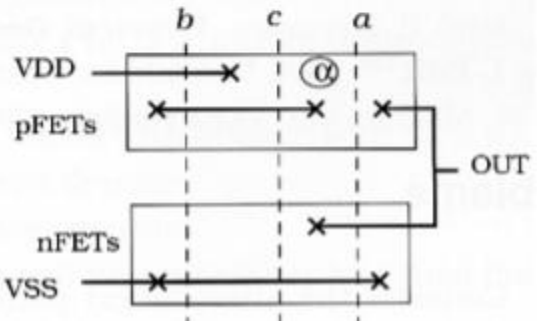
Euler Graphs

- Euler Graph
  - method for determining what order to layout txs
  - assign each circuit node as a point
  - assign each tx as a line between points
- Method
  - locate starting point
  - trace a loop from starting point through each transistor
    - can only re-cross a point once, separate nMOS loop must cross each pMOS
  - if above is possible, all tx can be in same Active
  - if not, will require multiple Active regions









STUDENTS-HUB.com
Uploaded By: anonymous