\[(a + b) \cdot (a + c) = a + a \cdot b + a \cdot c + b \cdot c\]
\[= a \cdot (1 + b) + a \cdot c + b \cdot c\]
\[= a \cdot (1 + c) + b \cdot c\]
\[= a + b \cdot c\]

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

CMOS

+ VDD

= CMOS logic circuit

V

VDD

logic 1 voltages

undefined

logic 0 voltages
Transistor Switching Characteristics

- **nMOS**
  - switching behavior
    - on = closed, when $V_{in} > V_{tn}$
      - $V_{tn} = \text{nMOS "threshold voltage"}$
      - $V_{in}$ is referenced to ground, $V_{in} = V_{gs}$
    - off = open, when $V_{in} < V_{tn}$

- **pMOS**
  - switching behavior
    - on = closed, when $V_{in} < V_{DD} - |V_{tp}|$
      - $|V_{tp}| = \text{pMOS "threshold voltage" magnitude}$
      - $V_{in}$ is referenced to ground, $V_{in} = V_{DD} - V_{sg}$
    - off = open, when $V_{in} > V_{DD} - |V_{tp}|$

**Rule to Remember**: 'source' is at
- lowest potential for nMOS
- highest potential for pMOS
Transistor Digital Behavior

- **nMOS**

<table>
<thead>
<tr>
<th>Vin</th>
<th>Vout (drain)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Vs=0</td>
<td>device is ON</td>
</tr>
<tr>
<td>0</td>
<td>?</td>
<td>device is OFF</td>
</tr>
</tbody>
</table>

- **pMOS**

<table>
<thead>
<tr>
<th>Vin</th>
<th>Vout (drain)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>?</td>
<td>device is OFF</td>
</tr>
<tr>
<td>0</td>
<td>Vs=VDD=1</td>
<td>device is ON</td>
</tr>
</tbody>
</table>

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
- \[ V_{gs} = V_{tn} \]
- \[ V_{y} = 0 \text{ V} \]

### pMOS

- ON when gate is 'low'
- **Passes a good high**
  - Min low is \(|V_{tp}|\)
- \[ V_{gs} = |V_{tp}| \]
- \[ V_{y} = |V_{tp}| \]

**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 \( V_g > V_i + V_{tn} \), then \( V_o = V_i \)     \((V_g-V_i > V_{tn})\)
      - here \( V_i \) is the “source” so the nMOS will pass \( V_i \) to \( V_o \)
    - case 2) if \( V_g < V_i + V_{tn} \), then \( V_o = V_g-V_{tn} \) \((V_g-V_i < V_{tn})\)
      - here \( V_o \) is the “source” so the nMOS output is limited
  For nMOS, \( \max(V_o) = V_g-V_{tn} \)

  - pMOS
    - case 1) if \( V_g < V_i - |V_{tp}| \), then \( V_o = V_i \) \((V_i-V_g > |V_{tp}|)\)
      - here \( V_i \) is the “source” so the pMOS will pass \( V_i \) to \( V_o \)
    - case 2) if \( V_g > V_i - |V_{tp}| \), then \( V_o = V_g+|V_{tp}| \) \((V_i-V_g < |V_{tp}|)\)
      - here \( V_o \) is the “source” so the pMOS output is limited
  For pMOS, \( \min(V_o) = V_g+|V_{tp}| \)

IMPORTANT:
Rules only apply if the devices is ON (e.g., \( V_g > V_{tn} \) for nMOS)
MOSFET Terminal Voltages: Examples

- **nMOS rules** \( \max(V_o) = V_g - V_{tn} \)
  - case 1) if \( V_g > V_i + V_{tn} \), then \( V_o = V_i \) \( (V_g - V_i > V_{tn}) \)
  - case 2) if \( V_g < V_i + V_{tn} \), then \( V_o = V_g - V_{tn} \) \( (V_g - V_i < V_{tn}) \)

- **nMOS examples** \( (V_{tn}=0.5\text{V}) \)
  - 1: \( V_g=5\text{V}, V_i=2\text{V} \)
    - \( V_g=5 > V_i + V_{tn} = 2.5 \Rightarrow V_o = 2\text{V} \)
  - 2: \( V_g=2\text{V}, V_i=2\text{V} \)
    - \( V_g=2 < V_i + V_{tn} = 2.5 \Rightarrow V_o = 1.5\text{V} \)

- **pMOS rules** \( \min(V_o) = V_g + |V_{tp}| \)
  - case 1) if \( V_g < V_i - |V_{tp}| \), then \( V_o = V_i \) \( (V_i - V_g > |V_{tp}|) \)
  - case 2) if \( V_g > V_i - |V_{tp}| \), then \( V_o = V_g + |V_{tp}| \) \( (V_i - V_g < |V_{tp}|) \)

- **pMOS examples** \( (V_{tp}=-0.5\text{V}) \)
  - 1: \( V_g=2\text{V}, V_i=5\text{V} \)
    - \( V_g=2 < V_i - |V_{tp}| = 4.5 \Rightarrow V_o = 5\text{V} \)
  - 2: \( V_g=2\text{V}, V_i=2\text{V} \)
    - \( V_g=2 > V_i - |V_{tp}| = 1.5 \Rightarrow V_o = 2.5\text{V} \)

\[ \begin{align*}
V_i & = 2 \text{V} \\
V_{tn} & = 0.5 \text{V} \\
V_{tp} & = -0.5 \text{V}
\end{align*} \]
Switch-Level Boolean Logic

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

  ![Switch Diagram](image)

  **Figure 2.1** Behavior of an assert-high switch
  - \( y = x \cdot A \), i.e. \( y = x \) if \( A = 1 \)
  - nMOS acts like an assert-high switch

  **Figure 2.2** Series-connected switches
  - a AND b

  **Figure 2.4** Parallel-connected switches
  - a OR b
Switch-Level Boolean Logic

- Characteristics of an assert-low switch

  \[ y = x \cdot \overline{A}, \text{ i.e. } y = x \text{ if } A = 0 \]

  error in figure 2.5

  Series assert-low switches \( \Rightarrow \) ?

  NOT (a OR b)

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

  DeMorgan relations

  \[ \overline{a} \cdot \overline{b} = a + b, \quad \overline{a} + \overline{b} = a \cdot b \]
**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 transistors
  - connect the output to ground through nMOS transistors
  - 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
    - \(1 + x = 1\)
    - \(0 + x = x\)
    - \(1 \cdot x = x\)
    - \(0 \cdot x = 0\)
    - \(x + x' = 1\)
    - \(x \cdot x' = 0\)
    - \(a \cdot a = a\)
    - \(a + a = a\)
    - \(ab + ac = a(b+c)\)
  - Properties which can be proven
    - \((a+b)(a+c) = a+bc\)
    - \(a + a'b = a + b\)
Review: Basic Transistor Operation

CMOS Circuit Basics

- **nMOS**
  - $V_{gs} = V_{tn} = \text{on}$
  - $V_{gs} > V_{tn} = \text{on}$
  - $V_{gs} = V_{DD} - V_{in}$
  - Strong '0', weak '1'

- **pMOS**
  - $V_{gs} > |V_{tp}| = \text{on}$
  - $V_{gs} = V_{DD} - V_{in}$
  - Strong '1', weak '0'

CMOS Pass Characteristics

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

- **nMOS**
  - 0 in $\rightarrow$ 0 out
  - VDD in $\rightarrow$ VDD-Vtn out
  - Strong '0', weak '1'

- **pMOS**
  - VDD in $\rightarrow$ VDD out
  - 0 in $\rightarrow$ |Vtp| out
  - Strong '1', weak '0'

Review: Switch-Level Boolean Logic

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

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

- **Inverter Function**
  - toggle binary logic of a signal

- **Inverter Switch Operation**
  - CMOS Inverter Schematic
    - 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**
  - \[ x \rightarrow y \]

- **Inverter Truth Table**
  - \[
    \begin{array}{c|c}
      x & y = \overline{x} \\
      \hline
      0 & 1 \\
      1 & 0 \\
    \end{array}
  \]

- **CMOS Inverter Schematic**
  - [Diagram of CMOS Inverter]
**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)

- 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**

\[
\text{NOR Symbol: } \overline{x + y}
\]

- **Karnaugh map**

\[
\begin{array}{c|c|c}
0 & x & y \\
0 & 0 & 1 \\
1 & 1 & 0 \\
\end{array}
\]

- **NOR Truth Table**

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>(\overline{x+y})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

- 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

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

• output is LOW if \( x \) OR \( y \) is true
  - parallel nMOS
• output is HIGH when \( x \) AND \( y \) are false
  - series pMOS

• 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**

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$\overline{xy}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

- Truth Table

- **K-map**

- **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**
  - $g(x, y, z) = x + y + z$

- **NAND3**
  - $g(x, y, z) = \overline{x \cdot y \cdot z}$

### 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
    - $f = a \cdot (b + c)$, $f = (d \cdot e) + a \cdot (b + c)$

- How do we construct the CMOS gate?
  - use DeMorgan principles to modify expression
    - construct nMOS and pMOS networks
      - $a \cdot b = \overline{\overline{a} + \overline{b}}$  
      - $a + b = \overline{\overline{a} \cdot \overline{b}}$
  
- use Structured Logic (covered only briefly in ECE410)
  - AOI (AND OR INV)
  - OAI (OR AND INV)
Using DeMorgan

- **DeMorgan Relations**
  - NAND-OR rule
    - bubble pushing illustration
    \[ x \cdot y = x + y \]
    \[ x + y = x \cdot y \]
  - NOR-AND rule
    \[ a + b = \overline{a \cdot b} \]
    \[ a \cdot b = \overline{a + b} \]
  - bubbles = inversions

- **pMOS and bubble pushing**
  - Parallel-connected pMOS
    \[ g(x,y) = \overline{x + y} = x \cdot y \]
    - assert-low OR
    - creates NAND function
  - Series-connected pMOS
    \[ g(x,y) = \overline{x \cdot y} = x + y \]
    - assert-low AND
    - creates NOR function

To implement pMOS this way, must push all bubbles to the inputs and remove all NAND/NOR output bubbles.
Review: CMOS NAND/NOR Gates

• NOR Schematic
  - output is LOW if $x \text{ OR } y$ is true
    - parallel nMOS
  - output is HIGH when $x \text{ AND } y$ are false
    - series pMOS

• NAND Schematic
  - output is LOW if $x \text{ AND } y$ are true
    - series nMOS
  - output is HIGH when $x \text{ OR } y$ 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
  
  \[ F_p = f(a', b', c') \]

- Form the nMOS network by complementing the output
  
  \[ F_n = f(a, b, c)' = \overline{F} \]

- Construct \( F_n \) and \( F_p \) using AND/OR series/parallel MOSFET structures
  - series = AND, parallel = OR

**EXAMPLE:**

\[ F = ab \Rightarrow \]

\[ F_p = a' \cdot b = a + b; \quad \text{OR/parallel} \]

\[ F_n = ab = ab; \quad \text{AND/series} \]
CMOS Combinational Logic Example

- Construct a CMOS logic gate to implement the function:
  \[ F = a \cdot (b + c) \]

- **pMOS**
  - Apply DeMorgan expansions
    - \[ F = a + (b + c) \]
    - \[ F = a + (b \cdot c) \]
  - Invert inputs for pMOS
    - \[ F_p = a + (b \cdot c) \]
  - Resulting Schematic

- **nMOS**
  - Invert output for nMOS
    - \[ F_n = a \cdot (b + c) \]
  - Apply DeMorgan
    - none needed
  - Resulting Schematic

14 transistors (cascaded gates)
6 transistors (CMOS)
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: \( F = a \cdot b + c \cdot d \)
  - 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: \( G = (x+y) \cdot (z+w) \)
  - 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

\[ X = a \cdot b + c \cdot d \]

- nMOS OAI structure
  - series of parallel txs

\[ Y = a + e \cdot b + f \]

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
Implementing Logic in CMOS

- **Reducing Logic Functions**
  - fewest operations ⇒ fewest txs
  - minimized function to eliminate txs
  - Example: \( x y + x z + x v = x (y + z + v) \)
  
<table>
<thead>
<tr>
<th>5 operations:</th>
<th>3 operations:</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 AND, 2 OR</td>
<td>1 AND, 2 OR</td>
</tr>
</tbody>
</table>
  
  # 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**
  \[ F = a + b \cdot (c + d); \] 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:
  \[ F = \overline{a} \cdot (b + c) \]

- **pMOS**
  - Apply DeMorgan expansions
    - none needed
  - Invert inputs for pMOS
    \[ F_p = a \cdot (\overline{b} + c) \]
  - Resulting Schematic?

- **nMOS**
  - Invert output for nMOS
    \[ F_n = a \cdot (b + c) \]
  - Apply DeMorgan
    \[ F_n = a + (b + c) \]
    \[ F_n = a + (\overline{b} \cdot c) \]
  - Resulting Schematic?
Yet Another Combinational Logic Example

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

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

- **nMOS**
  - Invert Output
    - \[ F_n = \overline{a} \cdot b \cdot (a + c) = a \cdot b + (a + c) \]
  - Eliminate NANDs and NORs
    - \[ F_n = a \cdot b + (\overline{a} \cdot \overline{c}) \]
  - Reduce Function
    - \[ F_n = \overline{a} \cdot (b + c) \]
  - Resulting Schematic
  - Complement operations for pMOS
    - \[ F_p = a + (b \cdot \overline{c}) \]
**XOR and XNOR**

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

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

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

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

<table>
<thead>
<tr>
<th>( a )</th>
<th>( b )</th>
<th>( a \oplus b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
XOR and XNOR AOI Schematic

(a) Exclusive-OR

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

(b) Exclusive-NOR

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

note: errors in textbook figure

uses exact same structure as generic AOI
**CMOS Transmission Gates**

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

- **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

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

\[ y = x \, s, \text{ for } s=1 \]
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**

  \[ F = P_0 \cdot \overline{s} + P_1 \cdot s \]

![Diagram of 2-to-1 MUX using TGs]

<table>
<thead>
<tr>
<th>s</th>
<th>TG0</th>
<th>TG1</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Closed</td>
<td>Open</td>
<td>(P_0)</td>
</tr>
<tr>
<td>1</td>
<td>Open</td>
<td>Closed</td>
<td>(P_1)</td>
</tr>
</tbody>
</table>
More TG Functions

- **TG XOR and XNOR Gates**

\[
\text{XOR: } a \oplus b = a \cdot \overline{b} + \overline{a} \cdot b \\
\text{XNOR: } a \oplus b = a \cdot b + \overline{a} \cdot \overline{b}
\]

\[
\begin{align*}
a \oplus b &= \overline{a} \cdot b + a \cdot \overline{b} \\
&= \overline{a} \cdot b, b = 1 \\
&= a \cdot \overline{b}, b = 1
\end{align*}
\]

- **Using TGs instead of “static CMOS”**
  - **TG OR gate**

\[
\begin{align*}
f &= a + b \\
\text{OR: } f &= a + \overline{a} \cdot b \\
\end{align*}
\]