(a + b) \cdot (a + c) = 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 hierarchy.
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

1. System Specifications
2. Abstract High-level Model
   - VHDL, Verilog HDL
3. Functional Simulation
4. Logic Synthesis
5. Chip Floorplanning
6. Chip-level Integration
7. Manufacturing
8. Finished VLSI Chip

9. Process Design
11. Process Models SPICE
12. Process Capabilities and Requirements

13. Functional/Timing/Performance Specifications
14. Schematic Design
15. Simulation
16. Physical Design
17. DRC (design rule check)
18. LVS (layout vs. schematic)
19. Parasitic Extraction
20. Post-Layout Simulation

21. Digital Cell Library
22. Mixed-signal Analog Blocks

ECE 410, Prof. F. Salem
Integrated Circuit Technologies

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

- CMOS dominates because:
  - Silicon is cheaper \(\Rightarrow\) 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 not the fastest technology!
  - BJT and III-V devices are faster
MOSFET Physical View

• Physical Structure of a MOSFET Device

![MOSFET Diagram]

- **n-Channel**
- Gate
- Source
- Drain
- Substrate, Bulk, Well, or Back Gate

- Critical dimension = “feature size”

• Schematic Symbol for 4-terminal MOSFET

![Schematic Symbols]

- nMOS
- pMOS

• 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

low power/transistor is critical for future ICs

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


“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

- System Specifications
- Abstract High-level Model
  - VHDL, Verilog HDL
- Functional Simulation
- Digital Cell Library
- Logic Synthesis
- Chip Floorplanning
- Chip-level Integration
- Manufacturing
- Finished VLSI Chip
- Process Design
  - Process Design Rules
  - Process Models
  - SPICE
  - Process Capabilities and Requirements
- Process Characterization
- Manufacturing
- Finished VLSI Chip
- Post-Layout Simulation
- Parasitic Extraction
- LVS (layout vs. schematic)
- Physical Design
- DRC (design rule check)
- Simulation
- Schematic Design
- Functional/Timing/Performance Specifications
MOSFET Physical View

- Physical Structure of a MOSFET Device

![MOSFET Physical View Diagram]

- Schematic Symbol for 4-terminal MOSFET

- Simplified Symbols

nMOS  
pMOS
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: \( E = \frac{V}{d} \)
  - 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: \( C = \varepsilon \frac{A}{d} \)
  - 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?
CMOS Cross Section View

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

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

- 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 \( \text{Vin} > \text{Vtn} \)
    - off = open, when \( \text{Vin} < \text{Vtn} \)

- **pMOS**
  - switching behavior
    - on = closed, when \( \text{Vin} < \text{VDD} - |\text{Vtp}| \)
    - off = open, when \( \text{Vin} > \text{VDD} - |\text{Vtp}| \)

- **Digital Behavior**
  - **nMOS**
    - \( \text{Vgs} > \text{Vtn} = \text{on} \)
    - \( \text{Vgs} = \text{VDD} - \text{Vin} \)
  - **pMOS**
    - \( \text{Vsg} > |\text{Vtp}| = \text{on} \)
    - \( \text{Vsg} = \text{VDD} - \text{Vin} \)

**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>
<thead>
<tr>
<th>LEVEL</th>
<th>SYMBOL</th>
<th>SWITCH CONDITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strong 1</td>
<td>1</td>
<td>P-SWITCH gate = 0, source = VDD</td>
</tr>
<tr>
<td>Weak 1</td>
<td>1</td>
<td>N-SWITCH gate = 1, source = VDD or P-SWITCH connected to VDD</td>
</tr>
<tr>
<td>Strong 0</td>
<td>0</td>
<td>N-SWITCH gate = 1, source = VSS</td>
</tr>
<tr>
<td>Weak 0</td>
<td>0</td>
<td>P-SWITCH gate = 0, source = VSS or N-SWITCH connected to VSS</td>
</tr>
<tr>
<td>High impedance</td>
<td>Z</td>
<td>N-SWITCH gate = 0 or P-SWITCH gate = 1</td>
</tr>
</tbody>
</table>

**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 \( V_g > V_i + V_{tn} \), 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

- 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

- Example (\( V_{tn}=0.5V \)):
  - \( V_g=5V, V_i=2V \) \( \Rightarrow V_o = 2V \)
  - \( V_g=2V, V_i=2V \) \( \Rightarrow V_o = 1.5V \)
  - For nMOS, \( \max(V_o) = V_g-V_{tn} \)

- Example (\( V_{tp}=-0.5V \)):
  - \( V_g=2V, V_i=5V \) \( \Rightarrow V_o = 5V \)
  - \( V_g=2V, V_i=2V \) \( \Rightarrow V_o = 2.5V \)
  - For pMOS, \( \min(V_o) = V_g+|V_{tp}| \)
Switch-Level Boolean Logic

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

\[ y = x \cdot A, \text{ i.e. } y = x \iff A = 1 \]

Series switches \( \Rightarrow \) AND function

Parallel switches \( \Rightarrow \) OR function

---

**Figure 2.1** Behavior of an assert-high switch

- \( y = x \cdot A, \text{ i.e. } y = x \iff A = 1 \) (iff=if and only if)
Switch-Level Boolean Logic

- Characteristics of an assert-low switch

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

  ![Diagram](image1)

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

  pMOS acts like an assert-low switch

Series assert-low switches $\Rightarrow$ ?

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

  ![Diagram](image2)

  - Remember This??
    
    $$\overline{a} \cdot b = a + b, \quad \overline{a} + b = a \cdot b$$

  DeMorgan relations

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

  - a=0 $\Rightarrow$ SW1 open, SW2 closed $\Rightarrow y = 1 = a$
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**
Review: Basic Transistor Operation

CMOS Circuit Basics

- **nMOS**
  - \( Vgs > Vtn = \text{on} \)
  - \( Vgs = VDD - Vin \)
  - \( Vg = Vin \)
  - \( Vout = \begin{cases} 1 & \text{on = closed} \\ 0 & \text{on = closed} \end{cases} \)
  - Strong '0', weak '1'

- **pMOS**
  - \( Vsg > |Vtp| = \text{on} \)
  - \( Vsg = VDD - Vin \)
  - \( Vg = Vin \)
  - \( Vout = \begin{cases} 1 & \text{on = closed} \\ 0 & \text{on = closed} \end{cases} \)
  - Strong '1', weak '0'

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**
  
  – \( y = x \cdot A \), i.e. \( y = x \) iff \( A = 1 \)

  – series = AND

  – parallel = OR

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

  – series = NOR

  – parallel = NAND
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
  - 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
  
  **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**
  
  $(a+b)(a+c) = a+bc$
  $a + a'b = a + b$
CMOS Inverter

- Inverter Function
  - toggle binary logic of a signal

- Inverter Switch Operation

input low $\rightarrow$ output high
nMOS off/open
pMOS on/closed

input high $\rightarrow$ output low
nMOS on/closed
pMOS off/open

- CMOS Inverter Schematic

- Inverter Symbol

- Inverter Truth Table

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

- CMOS Inverter Schematic

input high $\rightarrow$ output low
nMOS on/closed
pMOS off/open

input low $\rightarrow$ output high
nMOS off/open
pMOS on/closed
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

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

nMOS NAND

- series switches = AND function
- nMOS pulls low (NOTs the output)
CMOS NOR Gate

- NOR Symbol
  \[ x \] \[ y \] \[ x + y \]

- Karnaugh map

- NOR Truth Table

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>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>

- 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

\[ g(x,y) = \overline{x} \cdot \overline{y} \cdot 1 + x \cdot 0 + y \cdot 0 \]
CMOS NOR Gate

• **CMOS NOR Schematic**

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

- 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
CMOS NAND Gate

- NAND Symbol
- CMOS Schematic
- Truth Table

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>( x \cdot y )</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>

- K-map

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

- 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**
  \[ g(x,y,z) = \overline{x+y+z} \]

- **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?
- schematic resembles how the circuit will look in *physical layout*
Review: CMOS NAND/NOR Gates

• NOR Schematic
  
  - output is LOW if \( x \lor y \) is true
  - parallel nMOS
  - output is HIGH when \( x \land y \) are false
  - series pMOS

\[ g(x,y) = x + y \]

• NAND Schematic
  
  - output is LOW if \( x \land y \) are true
  - series nMOS
  - output is HIGH when \( x \lor y \) is false
  - parallel pMOS

\[ g(x,y) = \overline{x \cdot y} \]
Complex Combinational Logic

- **General logic functions**
  - for example

  \[ f = a \cdot (b + c), \quad f = (d \cdot e) + a \cdot (b + c) \]

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

\[ a \cdot b = \overline{a + b} \quad \overline{a + b} = a \cdot \overline{b} \]

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

- **DeMorgan Relations**
  - NAND-OR rule: $a \cdot b = a + b$
  - NOR-AND rule: $a + b = \overline{a \cdot b}$
  - bubble pushing illustration

- **bubbles = inversions**

- **pMOS and bubble pushing**
  - Parallel-connected pMOS
    - assert-low OR
    - creates NAND function
  - Series-connected pMOS
    - 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**
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
  \[ F_p = f(\overline{a}, \overline{b}, \overline{c}) \]
- Form the nMOS network by complementing the output
  \[ F_n = f(a, b, c) = F \]
- Construct \( F_n \) and \( F_p \) using AND/OR series/parallel MOSFET structures
  - series = AND, parallel = OR

EXAMPLE:

\[ F = \overline{ab} \Rightarrow \]
\[ F_p = \overline{a} \cdot \overline{b} = a+b; \quad \text{OR/parallel} \]
\[ F_n = \overline{\overline{ab}} = ab; \quad \text{AND/series} \]
CMOS Combinational Logic Example

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

- **pMOS**
  - Apply DeMorgan expansions
    \[ F = a + (b + c) \]
    \[ F = \overline{a} + (\overline{b} \cdot \overline{c}) \]
  - 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

- nMOS OAI structure
  - series of parallel txs

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

error in textbook Figure 2.45
AOI/OAI pMOS Circuits

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

![AOI Circuit](image1)

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

![OAI Circuit](image2)

**Complete CMOS AOI/OAI circuits**

![Complete CMOS AOI/OAI circuits](image3)
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
CMOS Logic Example

- Construct the function below in CMOS
  \[ F = a + b \cdot (c + d); \text{ remember AND operations occur before OR} \]
  \[ F_n = a + b \cdot (c + d) \]

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

• 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
    \[ F_n = \overline{a} \cdot b + (\overline{a} \cdot \overline{c}) \]
  - 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

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

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

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

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

Note: see textbook, figure 2.57
CMOS Transmission Gates

• 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

\[ y = x \cdot s, \text{ for } s=1 \]
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**
  
  $F = P_0 \cdot \overline{s} + P_1 \cdot s$

<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

\[ a \oplus b = \overline{a} \cdot b + a \cdot \overline{b} \]

\[ a \oplus b = \overline{a} \cdot b + a \cdot \overline{b} \]

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

(a) Closed switch

(b) Open switch
Figure 2.67  (p. 61)
Data synchronization using transmission gates.
Figure 2.68  (p. 62)
Block-level system timing diagram.
Figure 2.69 (p. 62)
Control of binary words using clocking planes.