Electronics Revolution

- Age of electronics
  - microcontrollers, DSPs, and other VLSI chips everywhere
- Electronics of today and tomorrow
  - high-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
PDAs
Camcorder

MP3/CD Player
Laptop
Cell phone

Nintendo
Gameboy

Figure 1.1 (p. 2)
The VLSI design funnel.

Figure 1.2 (p. 4)
General overview of the design hierarchy.

The VLSI design funnel.
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

Integrated Circuit Technologies

- **Why does CMOS dominate?**
  - other technologies
  - passive circuits
  - III-V devices
  - Silicon BJT
- **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 not the fastest technology!**
  - BJT and III-V devices are faster

MOSFET Physical View

- **Physical Structure of a MOSFET Device**
  - critical dimension = "feature size"
- **Schematic Symbol for 4-terminal MOSFET**
- **Simplified Symbols**
  - nMOS
  - pMOS
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.

![Feature Size Chart](http://www.intel.com/intel/museum/25anniv/hof/moore.htm)

“Electronics” Building block(s)

- MOSFET Device-- 1950+ to 2020

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

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
  - FET = Field Effect Transistor
  - CMOS = Complementary MOS
  - gate: gate oxide (insulator)- very thin
  - source and drain
  - channel

- Primary Features
  - use of both nMOS and pMOS to form a circuit with lowest power consumption.

- Process models
  - SPICE

- Process Characterization
  - Process Design
  - Process Design Rules

- Process Capabilities and Requirements

- System Specifications

- Digital Cell Library

- Analog/Mixed Signal Blocks

- Physical Design
  - DRC (design rule check)
  - LVS (layout vs. schematic)

- Post-Layout Simulation

- Chip Level Integration

- Manufacturing

- Finished VLSI Chip

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
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: $C = \frac{\varepsilon A}{d}$
  - 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?

CMOS Cross Section View

- Cross section of a 2 metal, 1 poly CMOS process
  - Layout (top view) of the devices above (partial, simplified)

CMOS Circuit Basics

- CMOS = complementary MOS
  - uses 2 types of MOSFETs
    - 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 $V_{in} > V_{tn}$
    - off = open, when $V_{in} < V_{tn}$
  - pMOS
    - switching behavior
    - on = closed, when $V_{in} < V_{DD} - |V_{tp}|$
    - off = open, when $V_{in} > V_{DD} - |V_{tp}|$
- Digital Behavior
  - nMOS
    - $V_{in} = V_{out} (drain)$
    - $V_{on} = 0$ device is ON
    - $V_{off} = V_{DD}$ device is OFF
  - pMOS
    - $V_{in} = V_{out} (drain)$
    - $V_{on} = V_{DD}$ device is OFF
    - $V_{off} = 0$ 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)

Rule to Remember:
- 'source' is at lowest potential (nMOS) and highest potential (pMOS)

\[
\begin{align*}
\text{nMOS} & : & \text{VDD} \rightarrow VDD, \quad \text{Vgs}=|\text{Vtp}|, & \text{Passes a good high} \quad \text{Min low is } |\text{Vtp}|, \\
\text{pMOS} & : & 0 \rightarrow VDD, \quad \text{Vgs}=\text{VDD}, & \text{Passes a good low} \quad \text{Max high is } \text{VDD}-|\text{Vtp}|
\end{align*}
\]

Switch-Level Boolean Logic

- Logic gates are created by using sets of controlled switches
- Characteristics of an assert-high switch
  \[ y = x \cdot A, \quad \text{i.e. } y = x \iff A = 1 \quad (\text{iff means if and only if}) \]

Series switches \(\Rightarrow\) AND function

Parallel switches \(\Rightarrow\) OR function

Switch-Level Boolean Logic

- Characteristics of an assert-low switch
  \[ y = x \oplus A, \quad \text{i.e. } y = x \iff A = 0 \]

Series assert-low switches \(\Rightarrow\) NOR

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

Remember This??:
- DeMorgan relations
  \[ a \cdot b = a + b, \quad a + b = a \cdot b \]
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
- only one logic network (p or n) is required to produce 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

- pMOS
  - \( V_{DD} \) in = \( V_{DD} - V_{TN} \) out
  - strong '0', weak '1'
- nMOS
  - \( V_{DD} \) in = \( V_{DD} \) out
  - strong '1', weak '0'

CMOS Pass Characteristics

- nMOS
  - Source is at lowest potential (nMOS) and highest potential (pMOS)
  - \( V_{DD} - |V_{TP}| \) = on
  - \( V_{GS} = V_{DD} - V_{IN} \)
- pMOS
  - \( V_{GS} > |V_{TP}| = \text{on} \)
  - \( V_{GS} = V_{DD} - V_{IN} \)

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 \( V_{DD} \) 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
  - \( (a + b)' = a' \cdot b' \)
  - \( (a \cdot b)' = a' + b' \)
- Properties which can be proven
  - \( a + a \cdot b = a + b \)

Useful Logic Properties

- \( 1 \cdot x = x \)
- \( 0 \cdot x = 0 \)
- \( x \cdot x = x \)
- \( x + x = x \)
- \( x + 0 = x \)
- \( x + 1 = 1 \)
- \( 0 + x = x \)
- \( 1 + x = 1 \)
- \( x + x' = 1 \)
- \( x \cdot x' = 0 \)
- \( a + ab = a \cdot b + a \cdot b \)
**CMOS Inverter**

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

- **Inverter Truth Table**

- **Inverter Symbol**

- **Inverter Switch Operation**

- **CMOS Inverter Schematic**

**CMOS Inverter**

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- **nMOS Logic Gates**

- **CMOS NOR Gate**

- **NOR Symbol**

- **NOR Truth Table**

- **Karnaugh map**

- **CMOS NOR Schematic**

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

- **g(x,y) = \overline{x} \cdot \overline{y} \cdot 1 + x \cdot 0 + y \cdot 0**
### CMOS NAND Gate

- **NAND Symbol**
- **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>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
- **CMOS Schematic**

  \[
g(x,y) = (x'y'1) + (x'y1) + (x'y1) + (x'y1) = x'y1 + x'y1 + x'y1 + x'y1
\]

- **K-map**

### 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 AND are true
  - output is HIGH when OR is false
- **NAND Schematic**
  - output is LOW if OR is true
  - output is HIGH when AND is false

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

- **DeMorgan Relations**
  - NAND-OR rule
    - bubble pushing illustration
  - NOR-AND rule
    - bubbles = inversions

- **pMOS and bubble pushing**
  - Parallel-connected pMOS
    - assert-low OR
    - creates NAND function
  - Series-connected pMOS
    - 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**
  \[ F_p = f(a, b, c) \]
- **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:**

\[ F = ab \]

\[ F_p = a + b = a \cdot b \] (OR/parallel)

\[ F_n = ab = ab \] (AND/series)

CMOS Combinational Logic Example

- **Construct a CMOS logic gate to implement the function:**
  \[ F = a \cdot (b + c) \]
- **PMOS**
  - Apply DeMorgan expansions
    \[ F = \overline{a \cdot (b + c)} \]
  - Invert inputs for pMOS
    \[ F_p = a \cdot (b + c) \]
  - Resulting Schematic

- **nMOS**
  - Invert output for nMOS
    \[ F_n = a \cdot (b + c) \]
  - Apply DeMorgan
    \[ F_n = a \cdot (b + c) \]
  - 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
  \[ \text{AND, OR, NOT (Invert)} \]
  - Example: \[ F = a + b + c + d \]
    - operation order: i) a AND b, c AND d, ii) (ab) OR (cd), iii) NOT
    - Inverted Sum-of-Products (SOP) form
  - **OAII**: implements logic function in the order
    \[ \text{OR, AND, NOT (Invert)} \]
    - Example: \[ F = \overline{ab} \cdot \overline{cd} \]
      - operation order: i) x OR y, z OR w, ii) (xy) AND (zw), 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

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

AOI/OAI pMOS Circuits

- pMOS AOI structure
  - series of parallel txs
  - opposite of nMOS

- pMOS OAI structure
  - series txs in parallel
  - opposite of nMOS

Complete CMOS AOI/OAI circuits

Implementing Logic in CMOS

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

- 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 vice versa

CMOS Logic Example

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

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

- pMOS
  - Group 2: c & d in series
  - Group 1: 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
    - Resulting Schematic?

  - **nMOS**
    - Invert output for nMOS
    - Resulting Schematic?

Yet Another Combinational Logic Example

• Implement the function below by constructing the nMOS network and complementing operations for the pMOS:
  \[ F = \overline{a} - b \cdot (a + c) \]

  - **nMOS**
    - Invert Output
      - \[ F_n = \overline{a} \cdot b \cdot (a + c) \]
      - Eliminate NANDs and NORs
      - \[ F_n = \overline{a} \cdot b + (a + c) \]
      - Reduce Function
      - \[ F_n = \overline{a} \cdot (b + c) \]
      - Resulting Schematic?

  - Complement operations for pMOS
    - \[ F_p = a + (b \cdot c) \]

XOR and XNOR

• **Exclusive-OR (XOR)**
  - \[ a \oplus b = \overline{a} \cdot b + a \cdot \overline{b} \]
  - not AOI form

• **Exclusive-NOR**
  - \[ \overline{a \oplus b} = a + b + \overline{a} \cdot \overline{b} \]
  - 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: \[ \overline{a \oplus b} = a \cdot b + a \cdot \overline{b} \]
  - formed by complementing XOR
  - thus, interchanging \( a \) and \( \overline{b} \) (or \( b \) and \( \overline{a} \)) converts from XOR to XNOR

XOR and XNOR AOI Schematic

- XOR: \[ a \oplus b = a \cdot b + a \cdot \overline{b} \]
- XNOR: \[ a \oplus b = a \cdot b + a \cdot \overline{b} \]
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 $s^*$
  - if $s = 1$, $y = x$, switch is closed, txs are on
  - if $s = 0$, $y = \text{unknown}$ (high impedance), $y = x \cdot s$, for $s = 1$
    switch open, txs off

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

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
  $$ F = P_0 \cdot s + P_1 \cdot s $$

More TG Functions

- TG XOR and XNOR Gates

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