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
- 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 heirarchy.
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

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/voltage 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.

“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.

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

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
  - 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: \( C = \frac{\varepsilon A}{d} \)
  - 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 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 \( V_{in} > V_{tn} \)
    - 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
    - \( V_{in} \) to \( V_{out} \) (drain)
      - \( 1 \) to \( 0 \) device is ON
      - \( 0 \) device is OFF
  - pMOS
    - \( V_{in} \) to \( V_{out} \) (drain)
      - \( 1 \) to \( 0 \) device is 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)

<table>
<thead>
<tr>
<th>Rule to Remember</th>
<th>nMOS</th>
<th>pMOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source is at lowest potential</td>
<td>passes a good low</td>
<td>passes a good high</td>
</tr>
<tr>
<td>Max high is VDD-Vtn</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Min low is</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**MOSFET Terminal Voltages**

- How do you find 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
  - Example ($V_{tn}=0.5V$): $V_g=5V, V_i=2V$  ⇒  $V_o = 2V$
  - $V_g=2V, V_i=2V$  ⇒  $V_o = 1.5V$

- 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 ($V_{tp}=-0.5V$): $V_g=2V, V_i=5V$  ⇒  $V_o = 5V$
  - $V_g=2V, V_i=2V$  ⇒  $V_o = 2.5V$

**Switch-Level Boolean Logic**

- Logic gates are created by using sets of controlled switches
- Characteristics of an assert-high switch
  - $y = x \cdot A$, i.e. $y = x$ if $A = 1$  (iff=if and only if)
- Series switches ⇒ AND function
- Parallel switches ⇒ OR function

**Switch-Level Boolean Logic (Error in Figure 2.5)**

- Characteristics of an assert-low switch
  - $y = x \cdot \overline{A}$, i.e. $y = x$ if $A = 0$
  - pMOS acts like an assert-low switch
  - Series assert-low switches ⇒ NOR
  - NOT function, combining assert-high and assert-low switches
  - $a + b = a \cdot b$,  $a + b = a \cdot b$
  - DeMorgan relations
**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

- **CMOS Circuit Basics**
  - nMOS
    - \( V_{DD} \) in = \( V_{DD} - V_{tn} \) out
    - strong ‘0’, weak ‘1’
  - pMOS
    - \( V_{DD} \) in = \( V_{DD} \) out
    - strong ‘1’, weak ‘0’

- **CMOS Pass Characteristics**
  - ‘source’ is at lowest potential (nMOS) and highest potential (pMOS)

**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**
  - \( y = x \cdot \bar{A} \), i.e. \( y = \bar{x} \) 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 \( V_{DD} \) through pMOS txs
  - connect the output to ground through nMOS txs
  - 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

**Properties which can be proven**

- **DeMorgan’s Rules**
  - \( (a \cdot b)' = a' + b' \)
  - \( (a + b)' = a' \cdot b' \)
  - \( a \cdot ab = a \cdot b \)
  - \( a + ab = a + b \)
CMOS Inverter

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

- **Inverter Symbol**
  - ![Inverter Symbol Image]

- **Inverter Truth Table**
  - | x | y | x \( \oplus \) y |
    |---|---|---|
    | 0 | 0 | 0 |
    | 0 | 1 | 1 |
    | 1 | 0 | 1 |
    | 1 | 1 | 0 |

- **CMOS Inverter Schematic**
  - input low \( \rightarrow \) output high
  - nMOS off/open
  - pMOS on/closed
  - CMOS Inverter
    - ![CMOS Inverter Schematic Image]

- **Inverter Switch Operation**
  - input high \( \rightarrow \) output low
  - nMOS on/closed
  - pMOS off/open

- **CMOS Inverter Operation**
  - input low \( \rightarrow \) output high
  - nMOS open
  - pMOS closed

- **CMOS Inverter Operation**
  - input high \( \rightarrow \) output low
  - nMOS closed
  - pMOS open

- **CMOS Inverter Function**
  - CMOS Inverter
    - ![CMOS Inverter Function Image]

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**
  - ![nMOS Inverter Image]

- **nMOS NOR**
  - ![nMOS NOR Image]

- **nMOS NAND**
  - ![nMOS NAND Image]

- **nMOS OR**
  - ![nMOS OR Image]

- **nMOS AND**
  - ![nMOS AND Image]

- **Parallel switches = OR function**
- **Series switches = AND function**

- **nMOS pull low (NOTs the output)**
- **nMOS pull low (NOTs the output)**

CMOS NOR Gate

- **NOR Symbol**
  - ![NOR Symbol Image]

- **NOR Truth Table**
  - | x | y | x \( \oplus \) y |
    |---|---|---|
    | 0 | 0 | 1 |
    | 0 | 1 | 0 |
    | 1 | 0 | 0 |
    | 1 | 1 | 0 |

- **Karnaugh map**
  - ![Karnaugh map Image]

- **G(x,y) = \( x \cdot y \cdot 1 + x \cdot 0 + y \cdot 0 \)**

- **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 NOR Schematic**
  - ![CMOS NOR Schematic Image]
**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**
- **K-map**
  \[ g(x, y) = (\overline{x} \overline{y} + 1) + (\overline{x} y + 1) + (x \overline{y} + 1) \]
  \[ (x + y + 0) \]
- **Output**
  - LOW if \( x \) AND \( y \) are true
  - Parallel nMOS
  - 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**
- **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
    \[ f = a \cdot (b + c), \quad f = (d \cdot e) + a \cdot (b + c) \]
  \[ a + b = a \cdot b \]
  \[ a \cdot b + a = b \]
- How do we construct the CMOS gate?
  - use DeMorgan principles to modify expression
    - construct nMOS and pMOS networks
  - use Structured Logic
    - AOI (AND OR INV)
    - OAI (OR AND INV)
Using DeMorgan

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

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

- 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
- Reduce (using DeMorgan) to eliminate inverted operations
- Inverted variables are OK, but not operations (NAND, NOR)
- Form pMOS network by complementing the inputs
- Form the nMOS network by complementing the output
- Construct Fn and Fp using AND/OR series/parallel MOSFET structures
  - series = AND, parallel = OR

EXAMPLE:

\[
F = ab \\
Fp = a + b = a \cdot b; \text{ OR/parallel} \\
Fn = ab = ab; \text{ AND/series}
\]

CMOS Combinational Logic Example

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

- PMOS
  - Apply DeMorgan expansions
  - Invert inputs for pMOS
  - Resulting Schematic

- nMOS
  - Invert output for nMOS
  - Apply DeMorgan
  - Resulting Schematic

- 14 transistors (labeled gates)

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: \(F = a \cdot b \cdot c + d\)
  - 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
  \[ F = a \cdot b + c \cdot d \]

• nMOS OAI structure
  - series of parallel txs

AOI/OAI pMOS Circuits

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

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

Implementing Logic in CMOS

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

  5 operations
  3 operations
  \# 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
  \[ F = a \cdot b \cdot (c + d) \]

  remember AND operations occur before OR

• nMOS
  - Group 2: \( c \& d \) in parallel
  - Group 1: \( b \) parallel to \( G1 \)
  - Group 3: \( a \) parallel to \( G2 \)

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

- **pMOS**
  - Apply DeMorgan expansions none needed
  - Invert inputs for pMOS
  - Resulting Schematic?

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

Yet 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
    - Eliminate NANDs and NORs
    - Reduce Function
  - Resulting Schematic?
  - Complement operations for pMOS

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: \( a \oplus b = a \cdot b + a \cdot \overline{b} \), formed by complementing XOR
  - thus, interchanging a and b (or b and \( \overline{b} \)) 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

  ![CMOS Transmission Gates Schematic]

  - controlled by gate select signals, s and $\overline{s}$
    - if $s = 1$, $y = x$, switch is closed
    - if $s = 0$, $y = \text{unknown}$ (high impedance)

  $y = x \cdot s$, 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**

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

**More TG Functions**

- **TG XOR and XNOR Gates**

  ![XOR and XNOR Gates Diagrams]

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

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