MOS Transistors

- Silicon substrate doped with impurities
- Adding or cutting away insulating glass (SiO₂)
- Adding wires made of polycrystalline silicon (polysilicon, poly) or metal, insulated from the substrate by SiO₂

Silicon Lattice

- Transistors are built on a silicon substrate
- Silicon is a Group IV material
- Forms crystal lattice with bonds to four neighbors

```
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Si</td>
<td>Si</td>
<td>Si</td>
<td>Si</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Si</td>
<td>Si</td>
<td>Si</td>
<td>Si</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Si</td>
<td>Si</td>
<td>Si</td>
<td>Si</td>
</tr>
</tbody>
</table>
```
Dopants

- Silicon is a semiconductor
- Pure silicon has no free carriers and conducts poorly
- Adding dopants increases the conductivity
- Group V: extra electron (n-type)
- Group III: missing electron, called hole (p-type)

```
<table>
<thead>
<tr>
<th>Si</th>
<th>Si</th>
<th>Si</th>
<th>Si</th>
<th>Si</th>
<th>Si</th>
<th>Si</th>
</tr>
</thead>
<tbody>
<tr>
<td>Si</td>
<td>As</td>
<td>Si</td>
<td>Si</td>
<td>Si</td>
<td>Si</td>
<td>Si</td>
</tr>
<tr>
<td>Si</td>
<td>Si</td>
<td>Si</td>
<td>Si</td>
<td>Si</td>
<td>Si</td>
<td>Si</td>
</tr>
</tbody>
</table>
```

p-n Junctions

- A junction between p-type and n-type semiconductor forms a diode.
- Current flows only in one direction

```
p-type   n-type
anode    cathode
```

nMOS Transistor

- Four terminals: gate, source, drain, body
- Gate – oxide – body stack looks like a capacitor
  - Gate and body are conductors
  - SiO₂ (oxide) is a very good insulator
  - Called metal – oxide – semiconductor (MOS) capacitor
  - Even though gate is no longer made of metal

![nMOS Diagram](image)

nMOS Operation

- Body is commonly tied to ground (0 V)
- When the gate is at a low voltage:
  - P-type body is at low voltage
  - Source-body and drain-body diodes are OFF
  - No current flows, transistor is OFF

![nMOS Operation Diagram](image)
nMOS Operation Cont.

- When the gate is at a high voltage:
  - Positive charge on gate of MOS capacitor
  - Negative charge attracted to body
  - Inverts a channel under gate to n-type
  - Now current can flow through n-type silicon from source through channel to drain, transistor is ON

pMOS Transistor

- Similar, but doping and voltages reversed
  - Body tied to high voltage ($V_{DD}$)
  - Gate low: transistor ON
  - Gate high: transistor OFF
  - Bubble indicates inverted behavior
Power Supply Voltage

- GND = 0 V
- In 1980’s, $V_{DD} = 5$V
- $V_{DD}$ has decreased in modern processes
  - High $V_{DD}$ would damage modern tiny transistors
  - Lower $V_{DD}$ saves power
- $V_{DD} = 3.3, 2.5, 1.8, 1.5, 1.2, 1.0, \ldots$

Transistors as Switches

- We can view MOS transistors as electrically controlled switches
- Voltage at gate controls path from source to drain

$\text{nMOS}$

\[ \begin{array}{ccc}
  g & \uparrow & d \\
  \downarrow & s & s \\
    & \text{OFF} & \text{ON} \\
\end{array} \]

$\text{pMOS}$

\[ \begin{array}{ccc}
  g & \downarrow & d \\
  \uparrow & s & s \\
    & \text{ON} & \text{OFF} \\
\end{array} \]
**MOS Transistor Switches**

- **N-switch**
  - S = 0 (Good 0)
  - S = 1 (Degraded 1)

- **P-switch**
  - S = 0 (Good 0)
  - S = 1 (Degraded 1)

- **CMOS switch**
  - S = 0 (Good 0)
  - S = 1 (Good 1)
**Signal Strength**

- *Strength* of signal
  - How close it approximates ideal voltage source
- $V_{DD}$ and GND rails are strongest 1 and 0
- nMOS pass strong 0
  - But degraded or weak 1
- pMOS pass strong 1
  - But degraded or weak 0
- Thus nMOS are best for pull-down network

**CMOS Inverter**

<table>
<thead>
<tr>
<th>A</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

![CMOS Inverter Diagram](image)
CMOS Logic Gates-1

Inverter

2-input NAND

CMOS Inverter

<table>
<thead>
<tr>
<th>A</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
CMOS Inverter

<table>
<thead>
<tr>
<th>A</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ \text{ON} \quad \text{OFF} \]

\[ \text{V}_{\text{DD}} \]

CMOS NAND Gate

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>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>1</td>
</tr>
</tbody>
</table>

\[ \text{A} \quad \text{B} \quad \text{Y} \]
CMOS NAND Gate

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>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>

A=0 Y=1
B=0

CMOS NAND Gate

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>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>

A=0 Y=1
B=1
**CMOS NAND Gate**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></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></td>
</tr>
</tbody>
</table>

**Truth Table**

- **A=1, B=0:** 
  - **Y=1** (ON)

**Schematic**

- **A=1**
  - Output **ON**
- **B=0**
  - Output **OFF**

---

**CMOS NAND Gate**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></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>

**Truth Table**

- **A=1, B=1:** 
  - **Y=0** (OFF)

**Schematic**

- **A=1**
  - Output **OFF**
- **B=1**
  - Output **ON**
CMOS NOR Gate

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>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>

CMOS Logic Gates-2

2-input NOR

- There is always (for all input combinations) a path from either 1 or 0 to the output.
- No direct path from 1 to 0 (low power dissipation).
- **Fully restored** logic.
- No ratio-ing is necessary (**ratio-less** logic).
- Generalize to n-input NAND and n-input NOR?
3-input NAND Gate

• Y pulls low if ALL inputs are 1
• Y pulls high if ANY input is 0
CMOS Compound (Complex) Gates-1

- What function is implemented by this circuit?

How to implement $F = ab + bc + ca$?

- $F = ab + bc + ca$
And-Or-Invert (AOI) Gates

- Generally, complex CMOS gates can be derived directly from maxterms of the function (as in a Karnaugh map)

Or-And-Invert (OAI) Gate
Transmission Gates

- Pass transistors produce degraded outputs
- Transmission gates pass both 0 and 1 well
Tristates

- *Tristate buffer* produces $Z$ when not enabled

<table>
<thead>
<tr>
<th>EN</th>
<th>A</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Tristates

- *Tristate buffer* produces $Z$ when not enabled

<table>
<thead>
<tr>
<th>EN</th>
<th>A</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Z</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Z</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Nonrestoring Tristate

- Transmission gate acts as tristate buffer
  - Only two transistors
  - But *nonrestoring*
    - Noise on A is passed on to Y

```
EN
/|
/ |
A  Y
/|
/ |
EN
```

Tristate Inverter

- Tristate inverter produces restored output
  - Violates conduction complement rule
  - Because we want a Z output

```
A
|
|
EN
|
|

Y
|
|
EN
|
|
EN
```
Tristate Inverter

- Tristate inverter produces restored output
  - Violates conduction complement rule
  - Because we want a Z output

\[
\begin{align*}
A & \quad EN = 0 \\
Y & \quad EN = 1 \\
\end{align*}
\]

Y = 'Z' 
Y = A

Multiplexers

- 2:1 multiplexer chooses between two inputs

<table>
<thead>
<tr>
<th>S</th>
<th>D1</th>
<th>D0</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>X</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
Multiplexers

- 2:1 multiplexer chooses between two inputs

<table>
<thead>
<tr>
<th>S</th>
<th>D1</th>
<th>D0</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

Gate-Level Mux Design

\[ Y = SD_1 + \overline{S}D_0 \] (too many transistors)

- How many transistors are needed?
Gate-Level Mux Design

\[ Y = SD_1 + \overline{SD}_0 \] (too many transistors)

- How many transistors are needed? 20

Transmission Gate Mux

- Nonrestoring mux uses two transmission gates
Transmission Gate Mux

- Nonrestoring mux uses two transmission gates
  - Only 4 transistors

Inverting Mux

- Inverting multiplexer
  - Use compound AOI22
  - Or pair of tristate inverters
  - Essentially the same thing

- Noninverting multiplexer adds an inverter
4:1 Multiplexer

• 4:1 mux chooses one of 4 inputs using two selects

– Two levels of 2:1 muxes
– Or four tristates
CMOS Exclusive-Nor Gate

• 8-transistor implementation

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>TG(_1)</th>
<th>TG(_2)</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>nonconducting</td>
<td>conducting</td>
<td>B (1)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>nonconducting</td>
<td>conducting</td>
<td>B (0)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>conducting</td>
<td>nonconducting</td>
<td>B (0)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>conducting</td>
<td>nonconducting</td>
<td>B (1)</td>
</tr>
</tbody>
</table>

• Better, 6-transistor implementation is possible!

D Latch

• When CLK = 1, latch is transparent
  – D flows through to Q like a buffer
• When CLK = 0, the latch is opaque
  – Q holds its old value independent of D
• a.k.a. transparent latch or level-sensitive latch
D Latch Design

- Multiplexer chooses D or old Q

D Latch Operation

- CLK = 1
- CLK = 0
D Flip-flop

- When CLK rises, D is copied to Q
- At all other times, Q holds its value
- a.k.a. positive edge-triggered flip-flop, master-slave flip-flop

D Flip-flop Design

- Built from master and slave D latches
D Flip-flop Operation

Race Condition

- Back-to-back flops can malfunction from clock skew
  - Second flip-flop fires late
  - Sees first flip-flop change and captures its result
  - Called *hold-time failure* or *race condition*
Nonoverlapping Clocks

- Nonoverlapping clocks can prevent races
  - As long as nonoverlap exceeds clock skew
- We will use them in this class for safe design
  - Industry manages skew more carefully instead

Design Representation Levels

- Design domains
  - Behavioral
  - Structural
  - Physical

Gajski and Kuhn’s Y-chart (layered like an onion)

- Hardware description languages commonly used at behavioral level, e.g. VHDL, Verilog
- Example: Consider the carry function \( c_o = ab + bc \)
Verilog Example (Behavioral)

Boolean equation form:

```
module carry (co, a, b, ci);
output co;
input a, b, ci;
assign
  co = (a & b) | (a & ci) | (b & ci);
end module
```

Boolean truth table form:

```
primitive carry (co, a, b, ci);
output co;
input a, b, ci;

// a  b  c    co
1   1  ?  :  1;
1   ?  1  :  1;
?   1  1  :  1;
0   0  ?  :  0;
0   ?  0  :  0;
?   0  0  :  0;

end table
end module
```

Timing information:

```
module carry (co, a, b, ci);
output co;
input a, b, ci;
Wire #10 co = (a & b) | (a & ci) | (b & ci);
end module
```

co changes 10 time units after a, b, or c changes

Verilog Example (Structural)

Structural representation of 4-bit adder (top-down):

```
module add4  (s, c4, ci, a, b);
  output [3:0] s;
  output [3:0] c4;
  input [3:0] a, b;
  input ci;
  wire [2:0] co;
  add a0 (co[0], s[0], a[0], b[0], ci);
  add a1 (co[1], ..., b[1], co[0]);
  add a2 (co[2], ..., , co[1]);
  add a3 (c4, s[3], a[3], b[3], co[2]);
end module
```

Technology-independent