Switch Networks
Basic Ideal Switch

• Simplest structure in a computing system is a switch

**IDEAL SWITCH**

INPUT → OUTPUT

• Path exists between INPUT and OUTPUT if Switch is CLOSED or ON
• Path does not exist between INPUT and OUTPUT if SWITCH is OPEN or OFF
Switch Networks
Switches in Series

SWITCHES IN SERIES

INPUT

S1

S2

OUTPUT

Truth Table

<table>
<thead>
<tr>
<th>S1</th>
<th>S2</th>
<th>PATH?</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
<td>OFF</td>
<td>NO</td>
</tr>
<tr>
<td>OFF</td>
<td>ON</td>
<td>NO</td>
</tr>
<tr>
<td>ON</td>
<td>OFF</td>
<td>NO</td>
</tr>
<tr>
<td>ON</td>
<td>ON</td>
<td>YES</td>
</tr>
</tbody>
</table>

• AND configuration
Switch Networks
Switches in Parallel

SWITCHES IN PARALLEL

Truth Table

<table>
<thead>
<tr>
<th>S1</th>
<th>S2</th>
<th>PATH?</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
<td>OFF</td>
<td>NO</td>
</tr>
<tr>
<td>OFF</td>
<td>ON</td>
<td>YES</td>
</tr>
<tr>
<td>ON</td>
<td>OFF</td>
<td>YES</td>
</tr>
<tr>
<td>ON</td>
<td>ON</td>
<td>YES</td>
</tr>
</tbody>
</table>

- OR configuration
Switch Networks
Input Selector

Truth Table

<table>
<thead>
<tr>
<th>S1</th>
<th>S2</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
<td>OFF</td>
<td>NONE</td>
</tr>
<tr>
<td>OFF</td>
<td>ON</td>
<td>INPUT 2</td>
</tr>
<tr>
<td>ON</td>
<td>OFF</td>
<td>INPUT 1</td>
</tr>
<tr>
<td>ON</td>
<td>ON</td>
<td>UNKNOWN</td>
</tr>
</tbody>
</table>

- *Crowbarred* level where logic level is indeterminate. Likely avoid this case.
Switch Networks
CMOS Switches

• The idea is to use the series and parallel switch configurations to route signals in a desired fashion.
• Unfortunately, it is difficult to implement an ideal switch as given.
• Complementary Metal Oxide Semiconductor (CMOS) devices give us some interesting components.
CMOS
Transfer Characteristics

<table>
<thead>
<tr>
<th>S</th>
<th>Switch</th>
<th>nMOS when CLOSED</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>open</td>
<td>Transmits logic level 0 well</td>
</tr>
<tr>
<td>1</td>
<td>close</td>
<td>Transmits logic level 1 poorly</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S</th>
<th>Switch</th>
<th>pMOS when CLOSED</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>close</td>
<td>Transmits logic level 1 well</td>
</tr>
<tr>
<td>1</td>
<td>open</td>
<td>Transmits logic level 0 poorly</td>
</tr>
</tbody>
</table>
CMOS
Transmission Gate (1)

**IDEAL SWITCH**

**CMOS TRANSMISSION GATE (SWITCH)**

<table>
<thead>
<tr>
<th>S</th>
<th>nMOS</th>
<th>pMOS</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>OFF</td>
<td>OFF</td>
<td>Z</td>
</tr>
<tr>
<td>1</td>
<td>ON</td>
<td>ON</td>
<td>INPUT</td>
</tr>
</tbody>
</table>

![Diagram of CMOS Transmission Gate](image)
CMOS
Transmission Gate (2)

SPLIT OF CURRENT ACROSS A TRANSMISSION GATE FOR LOGIC-0 AND LOGIC-1 INPUT

LOGIC-0 AT INPUT

LOGIC-1 AT INPUT

\( S = 0 \)

\( S = 1 \)

\( S = 0 \)

\( S = 1 \)
Switch Networks
High Impedance Z (1)

- With switches, we can consider three states for an output:
  - Logic-0
  - Logic-1
  - High Impedance Z

- Path exists for Logic-0 and Logic-1 when the switch is CLOSED.

  S
  0/1 —— OUTPUT = 0/1

- High impedance is a state where the switch is OPEN.

  S
  0/1 —— OUTPUT = Z
Switch Networks
High Impedance Z (2)

• Another way of thinking of switches is as follows
  • Path exists for Logic-0 and Logic-1 when the switch is CLOSED, meaning that the **impedance/resistance is small** enough to allow amply flow of current.

  $1 = \text{CLOSED}$

  SOURCE $\rightarrow$ DRAIN

  $\ll 10K\Omega$

  SOURCE $\leftarrow$ DRAIN

• High impedance is a state where the switch is OPEN, meaning that the **impedance/resistance is very large** allowing nearly no current flow.

  $0 = \text{OPEN}$

  SOURCE $\rightarrow$ DRAIN

  $\gg 100M\Omega$

  SOURCE $\leftarrow$ DRAIN
Switch Networks
Inverter (NOT)

\[ B = \overline{A} \]

- This network inverts the binary input value.
Switch Networks
NAND Network

\[ C = \overline{AB} \]
Switch Networks
NOR Network

\[ C = \overline{A + B} \]

<table>
<thead>
<tr>
<th>PULL-DOWN</th>
<th>PULL-UP</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\text{A} & \quad \text{B} & \quad \text{C} \\
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 0
\end{align*}
\]
Switch Networks
AND Network

C = AB

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</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>
Switch Networks
OR Network

\[ C = A + B \]

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

XOR Network

\[ C = AB + \overline{AB} \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</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>
Switch Networks
XNOR Network

\[ C = AB + AB \]

- Can this be implemented without the extra inverter at the output? Answer: Yes!
Switch Networks
Pull-Up/Pull-Down

\[ D = \overline{AC} + B \]

**Pull-Up**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Z</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Z</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Z</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Pull-Down**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Z</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Z</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Z</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Z</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Z</td>
</tr>
</tbody>
</table>
Switch Networks
Function Implementation

- Most Boolean functions can be easily implemented using switches.
- The basic rules are as follows
  - **Pull-up** section of switch network
    - Use *complements* for all literals in expression
    - Use only *pMOS devices*
    - Form *series* network for an **AND** operation
    - Form *parallel* network for an **OR** operation
  - **Pull-down** section of switch network
    - Use *complements* for all literals in expression
    - Use only *nMOS devices*
    - Form *parallel* network for an **AND** operation
    - Form *series* network for an **OR** operation
Switch Networks
Example Pull-Up

• To implement the Boolean function given below, the following pull-up network could be designed.

\[ F = E(\overline{AD} + \overline{B}(A + \overline{C})) \]
Switch Networks
Example Pull-Down

- To complete the switch design, the pull-down section for the Boolean function must also be designed.

\[ F = E(\overline{A}D + \overline{B}(A + \overline{C})) \]

- Notice how AND and OR become OR and AND circuits, respectively.
Switch Networks
Completed Example

- Putting the pull-up and pull-down pieces together gives the following CMOS switch implementation of the Boolean function.

\[ F = E(\overline{A}D + B(A + \overline{C})) \]
**Noise Margin**

- **Noise**: All undesirable voltage variations that are superimposed on normal operating voltage levels.

- **Noise Margin**: The maximum noise voltage level that a logic signal can tolerate without errors.

- High-level noise margin: $V_{OH} - V_{IH}$.

- Low-level noise margin: $V_{IL} - V_{OL}$.
**Fan-in/-out**

- **Fan-in:** Number of inputs available on a gate.

- **Fan-out (standard load):** Number of inputs that each output can drive.

  \[
  \text{Fan-out} = \min\{\frac{I_{OH}}{I_{IH}}, \frac{I_{OL}}{I_{IL}}\}.
  \]

- **Exp:** \( I_{OH} = 600\mu A, I_{IH} = 40\mu A, I_{OL} = 16\mu A, I_{IL} = 1.6\mu A \implies \min\{\frac{600}{40}, \frac{16}{1.6}\} = 10. \)

![Diagram of gates as current sources and sinks](image_url)
Propagation Delay

- **Rise time**: The delay for a signal to switch from 10% to 90% of its normal value.

- **Fall time**: The delay for a signal to switch from 90% to 10% of its normal value.

- **H-to-L propagation delay** ($t_{PHL}$): The time for the output signal to reach 50% of its normal value on the H-to-L transition after the input signal reached 50% of its normal signal. $t_{PLH}$ is defined similarly.

- In modern technologies, the propagation delay $t_{pd}$ is measured by

  \[
  t_{pd} = \frac{(t_{PHL} + t_{PLH})}{2}.
  \]

- In older technologies, the propagation delay $t_{pd}$ is measured by

  \[
  t_{pd} = \max\{t_{PHL}, t_{PLH}\}.
  \]
Propagation Delay for an Inverter

- High-to-low ($t_{PHL}$) and low-to-high ($t_{PLH}$) propagation delays are measured on output transitions.
Digital Logic Families: Bipolar Transistors

- Gates could be constructed using two transistor technologies: bipolar and metal-oxide-semiconductor (MOS); based on physical characteristics, they are organized into families.

- Bipolar technologies: using resistors, diodes, transistors.
  - RTL (Resistor-Transistor Logic) and DTL (Diode-Transistor Logic): used in 1960s.
  - ECL (Emitter-Coupled Logic): mainly for high-speed applications, e.g., supercomputers.

Bipolar transistor model
**Digital Logic Families: MOS Transistors**

- **MOS technologies:** a layer of metal above a layer of semiconductor with a layer of silicon dioxide serving as an insulation between them.

  - MOS, CMOS (Complementary MOS), BiCMOS (Bipolar CMOS) dominate modern digital designs.

  - CMOS dominates nMOS and pMOS, due to CMOS's lower power dissipation, high fan-out, simple process, and high regularity.

  The nMOS switch passes "0" well.

  The pMOS switch passes "1" well.

---

The nMOS switch passes "0" well.

The pMOS switch passes "1" well.