**Logic Circuits**

- A *gate* is a circuit element that operates on a binary signal.

- Logic operations typically have three methods of description:
  1. Equation symbol
  2. Truth table
  3. Circuit symbol

- The binary levels in logic gates are called 1 and 0.

- When levels refer to Boolean expressions they are referred to as true and false.
  Logic levels are $T$=True and $F$=False.
  Binary levels are 1=True and 0=False.

- When levels refer to electronic voltage levels they are called high and low.
  Logic levels are $H$=High and $L$=Low.
  Binary levels are 1=High and 0=Low.
Boolean Logic

• Boolean logic is based on the Boolean algebra with two operations: AND and OR.

\[
A \cdot \overline{A} = 0
\]
\[
A + \overline{A} = 1
\]

• A bar over a variable or expression represents the inverse value.

• Boolean algebra is commutative and distributive.

\[
A \cdot B = B \cdot A
\]
\[
A + B = B + A
\]
\[
A \cdot (B + C) = (A \cdot B) + (A \cdot C)
\]

• DeMorgan’s theorem links negation and the operations.

\[
\overline{A \cdot B} = \overline{A} + \overline{B}
\]
\[
\overline{A + B} = \overline{A} \cdot \overline{B}
\]
Unary Operations - 1 Input

- There are two unary operations.
- The identity operation leaves the value unchanged.

\[
\begin{array}{c|c|c}
A_{in} & A_{out} \\
0 & 0 \\
1 & 1 \\
\end{array}
\]

- The inverse operation reverses the value and is called NOT.

\[
\begin{array}{c|c|c}
A_{in} & \bar{A}_{out} \\
0 & 1 \\
1 & 0 \\
\end{array}
\]
Transistor Gates

- A single FET can form an identity, ...

Source Buffer Follower

\[ v_G = 0,\ v_S = v_G \]
\[ v_G = V_{DD},\ v_S = v_G \]

- ... or an inverse.

Common Source Inverter

\[ v_G = 0,\ v_D = V_{DD} - v_G = V_{DD} \]
\[ v_G = V_{DD},\ v_D = V_{DD} - v_G = 0 \]
Binary Operations - 2 Inputs

- There are two basic binary operations.
- The AND operation acts like multiplication.

\[ A \cdot B \]

\[
\begin{array}{c|c|c}
A & B & A \cdot B \\
0 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 1 \\
\end{array}
\]

- The OR operation acts like addition.

\[ A + B \]

\[
\begin{array}{c|c|c}
A & B & A + B \\
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 1 \\
\end{array}
\]
Compound Operations

- The combination of NOT and AND is NAND.

\[
\begin{array}{c|c|c}
A & B & \overline{A \cdot B} \\
0 & 0 & 1 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
\end{array}
\]

- The combination of NOT and OR is NOR.

\[
\begin{array}{c|c|c}
A & B & \overline{A + B} \\
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 0 \\
\end{array}
\]

- Either NAND or NOR gates can be used to create other logic gates.

\[
\overline{(A \cdot B) \cdot (A \cdot B)} = A \cdot B
\]

\[
\overline{(A + A) + (B + B)} = A \cdot B
\]
Electronic NAND

- The CMOS implementation of a NAND can be found in the 4011B.

The CMOS MOSFETs are connected as switches.
HIGH at A and B turn on Q₃ and Q₄ while turning off Q₁ and Q₂.
LOW at A and B turn on Q₁ and Q₂ while turning off Q₃ and Q₄.
- If both Q₃ and Q₄ are on then $V_{out}$ is at ground, otherwise either Q₁ or Q₂ will be on pulling $V_{out}$ up to $V_{DD}$. 
Exclusive OR

- The exclusive or is a common compound gate.
- The XOR selects inputs that differ.

\[
A \oplus B = (A + B) \cdot \overline{(A \cdot B)}
\]

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

- There are a number of equivalent logical constructions for XOR.

\[
A \oplus B = (A \cdot \overline{B}) + (\overline{A} \cdot B)
\]

\[
A \oplus B = (A \cdot \overline{B}) \cdot (\overline{A} \cdot B)
\]
Logic Types

- There are two common types of logic gates: TTL - Transistor-Transistor Logic and CMOS - Complementary Metal-Oxide-Semiconductor.

### Properties of TTL vs. CMOS

<table>
<thead>
<tr>
<th>Property</th>
<th>TTL</th>
<th>CMOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Supply</td>
<td>4.75 to 5.25 V</td>
<td>2 to 15 V</td>
</tr>
<tr>
<td>Input Current</td>
<td>LOW sources 0.5 mA</td>
<td>none (static charge may damage)</td>
</tr>
<tr>
<td>Input Threshold</td>
<td>1.3 V</td>
<td>0.3 $V_{DD}$ to 0.5 $V_{DD}$ (except HCT)</td>
</tr>
<tr>
<td>Output</td>
<td>LOW - 0.2 V</td>
<td>LOW - 0 V</td>
</tr>
<tr>
<td></td>
<td>HIGH - 3.7 to 4.4 V</td>
<td>HIGH - $V_{DD}$</td>
</tr>
<tr>
<td>Power Consumption</td>
<td>considerable</td>
<td>minimal except at high frequency</td>
</tr>
<tr>
<td>Other</td>
<td>threshold sensitive to noise</td>
<td>high output impedance (~100 $\Omega$)</td>
</tr>
</tbody>
</table>
Flip Flops

- A flip-flop can be forced into one state or the other and hold it when some inputs change.
- The RS (Set-Reset) flip-flop is forced into one state or another with an input to SET or RESET.
- Negative inputs mean that the effect is active when the signal is low.

- The SET and RESET cannot be used simultaneously or there is an ambiguous result.
Switch Debouncer

A mechanical switch will bounce on its contact and provide multiple pulses.

A flip-flop is sensitive to the first bounce, but none after until it is reset.
Triggered Flip-Flops

- A clock can be used to control which input is used.

The extra NAND gates allow the CLOCK to control whether S and R make it to the flip-flop. If CLOCK=0 the inputs to the flip-flop are disabled and Q stays constant. If CLOCK=1 the inputs are active and Q samples S and R.

- A disadvantage to the circuit is that if S or R change during the clock pulse, only the final state of the RS flip-flop is preserved when CLOCK=0 again.
**D-Type Flip-Flop**

- The D-type flip-flop is identical to a clocked RS flip-flop, except one input is inverted to form the other input.

$$\begin{array}{c}
\text{CLK} \\
\downarrow \\
\text{D} \\
\downarrow \\
\text{CLOCK} \\
\downarrow \\
\text{Q} \\
\downarrow \\
\bar{Q}
\end{array}$$

- The truth table shows states where the circuit retains the previous value.

<table>
<thead>
<tr>
<th>CLK</th>
<th>D</th>
<th>Q</th>
<th>\bar{Q}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Q</td>
<td>\bar{Q}</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Q</td>
<td>\bar{Q}</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>

- With only one input the indeterminate state (R=S=1) is avoided.
- D can make many transitions while CLK = 1, only the last level is stored when CLK = 0.
**Edge-Triggered Flip-Flops**

- Preceding flip-flops were level-sensitive and active when the clock level was correct.
- The master-slave flip-flop is an edge-sensitive trigger.

- This circuit consists of two level-sensitive D-type flip-flops.

During CLK=1, the first flip-flop is enabled, but the second is disabled (memory only).
During CLK=0, the first flip-flop is disabled, but the second is enabled, so it samples whatever is held at that time on flip-flop 1.

The output Q can only change exactly as CLK goes from 1 to 0.
**This is a negative edge trigger.**
Flip-Flop Timing

- An edge-triggered flip-flop can be built from three RS flip flops.

- The timing diagram shows the positive edge triggers in the stages.
Set and Clear Inputs

- Most edge-triggered flip-flops come with both set and clear options that work like S and R from an RS flip-flop.

\[
\begin{array}{c}
\text{S} \\
\text{D} \\
\text{Q} \\
\text{R} \\
\text{Q}
\end{array}
\]

If SET=0, then Q is forced to 1, and \(\overline{Q}\) to 0.
If CLEAR=0 (RESET), then \(\overline{Q}\) is forced to 1, and Q to 0.
SET and CLEAR take effect regardless of the state of CLK.

- D-type flip-flops also come with SET and CLEAR.
**JK Flip-Flops**

- The JK flip-flop uses 2-input combinatoric logic with feedback to set the D input of flip-flop.

\[
D = (J \oplus K) \cdot J + J \cdot K \cdot \overline{Q} + \overline{J} \cdot \overline{K} \cdot Q
\]

- When both J and K are low then the output remains as it was.
- When either J or K alone are high then the output matches the state of J.
- When both J and K are high then the output switches state.
- The JK flip-flop also comes with SET and CLEAR in a single chip.
Dividers

- A divider uses the negative output to toggle the output at slower rate than the input.
- A D-type flip-flop can easily become a divide by two circuit.

\[ D \] is always set with \( \overline{Q} \), so at each rising edge \( Q \) switches.
There is no confusion at the rising edge since there is a 10ns propagation delay through the flip-flop (74HC74), and the output needs to be stable for only 3 ns.

- The JK flip-flop can use the internal toggle setting to become a divide by two circuit.
Multistage Divider

- Multiple divide by two circuits can be combined to form a divide by $2^n$ circuit.

Each JK flip-flop is set to divide the clock by 2.

$D_0$ is dividing the input clock by 2.
$D_1$ is dividing the input clock by 4.
$D_2$ is dividing the input clock by 8.
$D_3$ is dividing the input clock by 16.

This can be extended to any arbitrary length.
Divide by 16 Truth Table

<table>
<thead>
<tr>
<th>C L K</th>
<th>D₀</th>
<th>D₁</th>
<th>D₂</th>
<th>D₃</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>