Topics

- CMOS Design
- Multi-input delay analysis
CMOS Logic Implementations

• Transmission Gate

<table>
<thead>
<tr>
<th>A</th>
<th>S</th>
<th>OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Z</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Z</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
CMOS Logic Implementations

- Transmission Gate

\[ \text{A} \quad \text{S} \quad \text{OUT} \]

- When S is low, the output is at high impedance
- When S is high, the output follows A
  - However, OUT can only rise to \( V_{DD} - V_T \), at which point the transistor will shut off
CMOS Logic Implementations

- Transmission Gate

\[ \text{A} \quad \text{S} \quad \text{OUT} \]

- When \( S \) is high, the output is at high impedance
- When \( S \) is low, the output follows \( A \)
  - However, \( \text{OUT} \) can only fall to \( V_T \), at which point the transistor will shut off
CMOS Logic Implementations

- Transmission gate
  - Requires both nMOS and pMOS transistors
  - Single nMOS or single pMOS will cause signal degradation
CMOS Logic Implementations

- Transmission Gate
CMOS Logic Implementations

- Multiplexer

\[
\begin{array}{c|c|c|c}
A & B & S & \text{OUT} \\
\hline
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 \\
\end{array}
\]

\[\text{OUT} = A\bar{S} + BS\]
CMOS Logic Implementation

- Multiplexer

\[ \text{OUT} = A \overline{S} + BS \]
CMOS Logic Implementations

- Multiplexer
  - Transmission gate based
CMOS Logic Implementations

- Memory
  - Use feedback loops to store bits
  - How do you get the bit in the loop?
CMOS Logic Implementations

- Memory
  - Use transmission gates to control entry to the loop

- Level sensitive D-latch
CMOS Logic Implementations

• Latches
  – Level sensitive latches do not allow you to isolate output from input when control is high.
  – Solution is to use a master-slave setup where master latches input and then slave latches the output
CMOS Logic Implementations

• Master-slave latch
CMOS Logic Implementations

• Master-slave latch

  CTL

  D

  Q1

  Q2

• Negative edge-triggered flip-flop
CMOS Logic

- Setup time is how much time before the clock edge that the data should be ready
  - If setup time is not met, the data will not have time to get through transmission gate and into the feedback loop.

- Hold time is the time that the data is required to be stable after the clock edge.
  - If hold time is not met, invalid data may get past the transmission gate and into the feedback loop.
# CMOS Logic

- **Flip-Flops**

<table>
<thead>
<tr>
<th>D</th>
<th>Q0</th>
<th>Q1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Q0</th>
<th>Q1</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>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>J</th>
<th>K</th>
<th>Q0</th>
<th>Q1</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>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</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>0</td>
</tr>
</tbody>
</table>
CMOS Logic

• Registers
  – N-bit collection of flip-flops
CMOS Logic Gate Design

• How do you characterize delay for a gate more complicated than an inverter?

\[ t_d = A \frac{C_L}{k_{eff}} \]

• \( k_{eff} \) is determined by the structure of the logic gate
Delay time analysis

• Multi-input gates

\[ k_{neff} = \frac{1}{\frac{1}{k_{n1}} + \frac{1}{k_{n2}}} = \frac{k_n}{2} \]

• For pull down (falling delay time)

\[ k_{peff} = k_p \]
CMOS Logic Implementations

- NAND

\[ k_{\text{eff}-n} = \frac{k_n}{2} \]
\[ k_{\text{eff}-p} = k_p \]

Falling delay has doubled.
Delay time analysis

- Multi-input gates

- For equal delay times
  \[ k_n = 2k_p \]
  \[ \frac{W_n}{L_n} k'_n = 2 \frac{W_p}{L_p} k'_p \]
CMOS Logic Implementations

- NAND

\[
\begin{align*}
k_n &= 2k_p \\
\frac{W_n}{L_n} k'_n &= 2 \frac{W_p}{L_p} k'_p \\
\text{Since } k'_n &= 2 k'_p \text{ and assume } L_n = L_p \\
\text{we get } W_n &= W_p
\end{align*}
\]
Delay time analysis

\[ k_{n-eff} = k_n \quad \text{(single transistor on)} \]

\[ k_{p-eff} = \frac{1}{\frac{1}{k_{p1}} + \frac{1}{k_{p2}}} \]

\[ = \frac{k_p}{2} \]

• For pull up (rising delay time)
• For pull down (falling delay time)
CMOS Logic Implementations

• NOR

\[ k_{\text{eff}-n} = k_n \]
\[ k_{\text{eff}-p} = \frac{k_p}{2} \]
Rising delay has doubled
Delay time analysis

• For equal delay times
  \[ 2k_n = k_p \]
  \[ 2 \frac{W_n}{L_n} k'_n = \frac{W_p}{L_p} k'_p \]
CMOS Logic Implementations

- NOR

\[ 2k_n = k_p \]
\[ 2 \frac{W_n k'_n}{L_n} = \frac{W_p}{L_p} k'_p \]

Since \( k'_n \approx 2k'_p \) and assume \( L_n = L_p \)
we get \( 4W_n = W_p \)
CMOS Logic Implementations

- Multi-input NOR

\[ k_{eff-n} = k_n \]
\[ k_{eff-p} = \frac{k_p}{3} \]

Rising delay has tripled
CMOS Logic Implementations

- Multi-input NOR

\[
3k_n = k_p
\]
\[
\frac{3W_n - k_n'}{L_n} = \frac{W_p - k_p'}{L_p}
\]

Since \( k_n' \approx 2k_p' \) and assume \( L_n = L_p \)

we get \( 6W_n = W_p \)
CMOS Logic Implementations

\[ k_{eff-n} = \frac{k_n}{3} \]
\[ k_{eff-p} = \frac{k_p}{2} \]

Falling delay has tripled
Rising delay has doubled
CMOS Logic Implementations

Since $k'_n \approx 2k'_p$ and assume $L_n = L_p$

we get $4W_n = 3W_p$.

$$2k'_n = 3k_p$$

$$2 \frac{W_n}{L_n} k'_n = 3 \frac{W_p}{L_p} k'_p$$
CMOS Logic Implementations

Without sizing
\[ t_{p1} = A\left(\frac{C_{in}}{k}\right) + A\left(\frac{C_L}{3k}\right) \]

With sizing
\[ t_{p2} = A\left(\frac{7C_{in}}{3k}\right) + A\left(\frac{C_L}{k}\right) \]
CMOS Logic Gate Delays

• Increasing transistor sizes can reduce the delays, but it
  – Increases area
  – Increases load capacitance for driving gates

• Multi-input gates may not always be good
Next class

• Gate Delays
• Logical Effort
• Chapter 6.2