Logic Design with MOSFETs

Dae Hyun Kim

EECS
Washington State University
References

  – Chapter 2
  – Chapter 1
Goal

- Design logic gates using MOSFETs (NMOS and PMOS)
Signals and Wires

- **Signals**
  - $0 = V_{SS} = \text{Ground} = \text{GND} = \text{Low} = 0\text{V}$
  - $1 = V_{DD} = \text{Power} = \text{PWR} = \text{High} = 5\text{V}, 3.3\text{V}, 1.5\text{V}, 1.2\text{V}, 1.0\text{V}$, etc.

- **Wires**

  ![Diagram](image)

  **No connection**  Connection
**Ideal Switches**

- **Switch**
  - Electrically open
  - Assert-high switch
    - $A = 0$
    - Open ($y$ is undefined)
  - Assert-low switch
    - $A = 1$
    - Closed ($y = x$)

- **Assert-high switch**
  - Electrically short
  - $x = y = x$
  - Closed ($y = x$)
  - Open ($y$ is undefined)

- **Assert-low switch**
  - $A = 0$
  - Closed ($y = x$)
  - $A = 1$
  - Open ($y$ is undefined)
Series/Parallel Connections of Switches

- **Series**

  
  ![Series Connection Diagram]

  \[ y = (x \cdot a) \cdot b = x \cdot (a \cdot b) \]

  **AND operation**
  
  \( y \) is defined only when \( a = 1 \) and \( b = 1 \)
  
  \( y \) is undefined if \( a = 0 \) or \( b = 0 \)

- **Parallel**

  
  ![Parallel Connection Diagram]

  \[ x \cdot a + x \cdot b = x \cdot (a + b) \]

  **OR operation**
  
  \( y \) is defined only when \( a = 1 \) or \( b = 1 \)
  
  \( y \) is undefined if \( a = 0 \) and \( b = 0 \)
Inverter Design with Switches

- **Inverter**
  - The output is defined both when \( a = 0 \) and when \( a = 1 \).

<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>

\[
y = 1 \cdot \bar{a} + 0 \cdot a = \bar{a}
\]
Inverter Design with Switches

- Two inverter designs

Why?
MOSFETs as Switches

- **MOSFET**: Metal-Oxide-Semiconductor Field-Effect Transistor
  - n-channel MOSFET = nFET = NMOS
  - p-channel MOSFET = pFET = PMOS
  - Complementary MOS: CMOS

- **Symbols**

  ![Diagram of nFET and pFET symbols]

  - nFET
    - $V_G$: Gate
    - $V_S$: Source
    - $V_D$: Drain
    - $(V_D \geq V_S)$

  - pFET
    - $V_G$: Gate
    - $V_D$: Drain
    - $V_S$: Source
    - $(V_S \geq V_D)$
MOSFETs as Switches

- Threshold voltage
  - nFET: $V_{Tn} > 0$
  - pFET: $V_{Tp} < 0$

- nFET
  - OFF: $V_{GSn} \leq V_{Tn}$
  - ON: $V_{GSn} > V_{Tn}$

- pFET
  - OFF: $V_{SGp} \leq |V_{Tp}|$
  - ON: $V_{SGp} > |V_{Tp}|$
MOSFETs as Switches

• Example (PTM High-Performance 45nm High-K Metal Gate)
  – $V_{DD}$: 1.0V
  – $V_{Tn}$: 0.46893V
  – $V_{Tp}$: -0.49158V

• Example (PTM High-Performance 32nm High-K Metal Gate)
  – $V_{DD}$: 0.9V
  – $V_{Tn}$: 0.49396V
  – $V_{Tp}$: -0.49155V

• Example (PTM High-Performance 22nm High-K Metal Gate)
  – $V_{DD}$: 0.8V
  – $V_{Tn}$: 0.50308V
  – $V_{Tp}$: -0.4606V
Pass Characteristics

- **nFET**

  \[ V_D = V_{in} \]
  \[ V_G = V_{DD} \]
  \[ V_S = V_{out} \]

- **pFET**

  \[ V_S = V_{in} \]
  \[ V_G = 0 \]
  \[ V_D = V_{out} \]

### nFET

<table>
<thead>
<tr>
<th>( V_{in} \uparrow )</th>
<th>( V_{GS} \downarrow )</th>
<th>( V_{out} \uparrow )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( V_{DD} )</td>
<td>0</td>
</tr>
<tr>
<td>0.1</td>
<td>( V_{DD} - 0.1 )</td>
<td>0.1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>( V_{DD} - V_{Tn} )</td>
<td>( V_{Tn} )</td>
<td>( V_{DD} - V_{Tn} )</td>
</tr>
<tr>
<td>( V_{DD} )</td>
<td>( V_{Tn} )</td>
<td>( V_{DD} - V_{Tn} )</td>
</tr>
</tbody>
</table>

Logic 0 transfer: **strong logic 0**

Logic 1 transfer: **weak logic 1**

### pFET

<table>
<thead>
<tr>
<th>( V_{in} \downarrow )</th>
<th>( V_{SG} \downarrow )</th>
<th>( V_{out} \downarrow )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_{DD} )</td>
<td>( V_{DD} )</td>
<td>( V_{DD} )</td>
</tr>
<tr>
<td>( V_{DD} - \varepsilon )</td>
<td>( V_{DD} - \varepsilon )</td>
<td>( V_{DD} - \varepsilon )</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>(</td>
<td>V_{Tp}</td>
<td>)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>(</td>
</tr>
</tbody>
</table>

Logic 1 transfer: **strong logic 1**

Logic 0 transfer: **weak logic 0**
Pass Characteristics

- SPICE simulation (45nm technology)
  - nFET

\[ \begin{align*}
V_{in} & \quad V_{DD} \\
V_{out} & \quad 0
\end{align*} \]
Pass Characteristics

- SPICE simulation (45nm technology)
  - pFET
Pass Characteristics

- **nFET**
  - *Strong logic 0* transfer
  - Weak logic 1 transfer

- **pFET**
  - *Strong logic 1* transfer
  - Weak logic 0 transfer

- **CMOS**
  - Use pFETs to pass logic 1.
  - Use nFETs to pass logic 0.
Basic Logic Gates in CMOS

- **Principles**
  - Construct the nFET network using only nFETs and the pFET network using only pFETs.
  - If the output is 1, the pFET network connects $V_{DD}$ to the output and the nFET network disconnects $V_{SS}$ and the output.
  - If the output is 0, the nFET network connects $V_{SS}$ to the output and the pFET network disconnects $V_{DD}$ and the output.
Basic Logic Gates in CMOS

- Inverter

\[ f = \bar{x} \]

\[ f = \bar{x} \cdot 1 + x \cdot 0 = \bar{x} \]

- Number of Transistors (TRs): 2
- nFET: 1
- pFET: 1
Basic Logic Gates in CMOS

- SPICE simulation
Basic Logic Gates in CMOS

- Two-input NAND (NAND2)

\[ f = \overline{a \cdot b} \]

\[ f = \overline{a} \cdot \overline{b} \cdot 1 + \overline{a} \cdot b \cdot 1 + a \cdot \overline{b} \cdot 1 + a \cdot b \cdot 0 = \overline{a} + \overline{b} = \overline{a \cdot b} \]

# TRs: 4
nFETs: 2
pFETs: 2
Basic Logic Gates in CMOS

- SPICE simulation
Basic Logic Gates in CMOS

- Two-input NOR (NOR2)

\[ f = \overline{a + b} \]

\[ f = \overline{a} \cdot \overline{b} + \overline{a} \cdot b + a \cdot \overline{b} + a \cdot b = \overline{a} \cdot \overline{b} = a + b \]

# TRs: 4  
nFETs: 2  
pFETs: 2
Basic Logic Gates in CMOS

• SPICE simulation
Complex Logic Gates in CMOS

- Example

\[ f = a \cdot (b + c) \]

- Using logic gates

\[ \begin{array}{c}
    b \\
    c \\
    a
\end{array} \quad \begin{array}{c}
    \text{# TRs: 14} \\
    \text{nFETs: 7} \\
    \text{pFETs: 7}
\end{array} \]

- Using logic gates

\[ \begin{array}{c}
    b \\
    c \\
    a
\end{array} \quad \begin{array}{c}
    \text{# TRs: 10} \\
    \text{nFETs: 5} \\
    \text{pFETs: 5}
\end{array} \]

- Using TRs

\[ \begin{array}{c}
    b \\
    c \\
    a
\end{array} \quad \begin{array}{c}
    \text{# TRs: 6} \\
    \text{nFETs: 3} \\
    \text{pFETs: 3}
\end{array} \]
Complex Logic Gates in CMOS

• How to design
  – Inverter
    \[ f = \overline{x} = \overline{x} \cdot 1 + x \cdot 0 \]
    
    pFET network (connects 1 and the output)
    nFET network (connects 0 and the output)
  
  – NAND2
    \[ f = a \cdot b = (a \cdot b) \cdot 1 + (a \cdot b) \cdot 0 = (\overline{a} + \overline{b}) \cdot 1 + (a \cdot b) \cdot 0 \]
    
    pFET network (expressed by \( \overline{a} \) and \( \overline{b} \))
    nFET network (expressed by \( a \) and \( b \))
Complex Logic Gates in CMOS

• How to design \( f \)
  – Express \( f = A \cdot 1 + B \cdot 0 = F(\overline{x_1}, \ldots, \overline{x_n}) \cdot 1 + F(x_1, \ldots, x_n) \cdot 0 \)
  – Design a pFET network using \( A = F(\overline{x_1}, \ldots, \overline{x_n}) \).
    • pFETs are ON when the inputs are 0.
  – Design an nFET network using \( B = F(x_1, \ldots, x_n) \).
    • nFETs are ON when the inputs are 1.

• Example

\[
f = a \cdot (b + c)
\]

\[
f = a \cdot (b + c) \cdot 1 + a \cdot (b + c) \cdot 0 = (\overline{a} + \overline{b} \cdot \overline{c}) \cdot 1 + a \cdot (b + c) \cdot 0
\]

pFET network

nFET network
Complex Logic Gates in CMOS

• Example

\[ f = a \cdot (b + c) \]

# TRs: 6  
nFETs: 3  
pFETs: 3
Complex Logic Gates in CMOS

• Structured logic design
  – Design a given Boolean equation using nFETs and pFETs.

• Assume that only non-inverted input signals are given.
  – $a, b, c, \ldots$ are given.
  – $\bar{a}, \bar{b}, \bar{c}, \ldots$ are not given. If you need them, you should generate them.
Complex Logic Gates in CMOS

• Design methodology 1
  - When \( f = \overline{S(x_1, \ldots, x_n)} \) (\( S \) is a function of non-inverted variables)
    • \( f = \overline{S} = \overline{S} \cdot 1 + \overline{S} \cdot 0 \)
    • Design an nFET network for \( S \) using \( x_1, \ldots, x_n \).
    • Design a pFET network for \( \overline{S} \) using \( \overline{x_1}, \ldots, \overline{x_n} \).
    • Connect them to \( V_{DD}, V_{SS}, f \).
  - Example: \( f = a \cdot (b + c) \)
    • \( f = a \cdot (b + c) \cdot 1 + a \cdot (b + c) \cdot 0 \)
    • Design an nFET network for \( a \cdot (b + c) \).
    • Design a pFET network for \( a \cdot (b + c) = a \overline{b} \cdot \overline{c} \).
    • Connect them.
### Complex Logic Gates in CMOS

- **Design methodology 2**
  - When \( f = S(x_1, \ldots, x_n) \)
    - \( f = \overline{S} = \overline{S} \cdot 1 + S \cdot 0 \)
    - Design an nFET network for \( S \).
    - Design a pFET network with a dual logic of the nFET network.
      - Dual of \( f(x_1, \ldots, x_n, 0,1, AND, OR) = f(x_1, \ldots, x_n, 1,0, OR, AND) \)
    - Connect them.
  - **Example:** \( f = a \cdot (b + c) \)
    - \( f = a \cdot (b + c) \cdot 1 + a \cdot (b + c) \cdot 0 \)
    - Design an nFET network for \( a \cdot (b + c) \).
    - Dual of \( a \cdot (b + c) = a + (b \cdot c) = a + b \cdot c \).
    - Connect them.
Complex Logic Gates in CMOS

• Dual logic
  – \( f(x_1, \ldots, x_n, 0, 1, \text{AND, OR})^D = f(x_1, \ldots, x_n, 1, 0, \text{OR, AND}) \)
  – Example
    • \((A \cdot B)^D = A + B\)
    • \((A + B)^D = A \cdot B\)
    • \((1 \cdot A)^D = 0 + A = A\)
    • \((1 + A)^D = 0 \cdot A = 0\)
    • \((0 \cdot A)^D = 1 + A = 1\)
    • \((0 + A)^D = 1 \cdot A = A\)

• Principles of the dual logic
  – The nFET and the pFET networks work complementarily.
  – If the nFET network is ON (i.e., connects \(V_{SS}\) to the output), the pFET network is OFF (i.e., disconnect the output from \(V_{DD}\)) and vice versa.
  – If two networks are dual, they work complementarily.
    • Prove!
Complex Logic Gates in CMOS

• Principles of the dual logic
  - \[ f = \overline{S(x_1, ..., x_n)} = f = \overline{S(x_1, ..., x_n) \cdot 1 + S(x_1, ..., x_n) \cdot 0} \]
  - \[ \overline{S(x_1, ..., x_n)} = \overline{S(x_1, ..., x_n, 0,1,AND,OR)} = \overline{S(x_1, ..., x_n, 1,0,OR,AND)} = S(\overline{x_1}, ..., \overline{x_n}) \] (De Morgan’s law)
  - A pFET is ON when its control variable \((x_i)\) is 0.
  - Thus, the pFET network is the dual of the nFET network.
Complex Logic Gates in CMOS

- Design methodology 3
  - When $f = S(x_1, ..., x_n)$ ($S$ is a function of non-inverted variables)
    - $f = S = \bar{S}$
    - Design $\bar{S}$ and add an inverter at the output.
  - Example: $f = a \cdot (b + c)$
    - $f = a \cdot (b + c) = \overline{a \cdot (b + c)}$
    - Design $\overline{a \cdot (b + c)}$.
    - Add an inverter at the output.
Complex Logic Gates in CMOS

• Design methodology 4
  – When \( f = S(\overline{x_1}, \ldots, \overline{x_n}) \) (\( S \) is a function of inverted variables)
    • Generate inverted inputs \((\overline{x_1}, \ldots, \overline{x_n})\) from the given inputs \((x_1, \ldots, x_n)\).
    • Design \( S \) using the inverted inputs.
  – Example: \( f = \overline{a} \cdot (\overline{b} + \overline{c}) \)
    • Inverters are not shown for brevity.
Complex Logic Gates in CMOS

- Design methodology 5
  - When \( f = S(x_1, \ldots, x_n) \)
    - \( f = S(x_1, \ldots, x_n) = S(x_1, \ldots, x_n)^D = S(x_1, \ldots, x_n)^D \)
    - Design \( S(x_1, \ldots, x_n)^D \) using the given inputs.
    - Add an inverter at the output.
  - Example: \( f = \overline{a} + (\overline{b} \cdot \overline{c}) \)
    - \( f = a \cdot (b + c) = a \cdot (b + c) \)
Complex Logic Gates in CMOS

• Design methodology 6
  – When \( f = S(x_1, \ldots, x_n) \)
    • \( f = S = \overline{\bar{S}} \)
    • Generate inverted inputs \((\overline{x_1}, \ldots, \overline{x_n})\) from the given inputs \((x_1, \ldots, x_n)\).
    • Design \( \bar{S} \) using the inverted inputs and add an inverter at the output.

• Design methodology 7
  – When \( f = S(x_1, \ldots, x_n) \)
    • \( f = \overline{S(x_1, \ldots, x_n)} = \overline{S(x_1, \ldots, x_n)}^D \)
    • Design \( S^D \) using the given non-inverted inputs \((x_1, \ldots, x_n)\).

• Design methodology 8
  – When \( f = S(x_1, \overline{x_1}, \ldots, \overline{x_n}) \) or \( S(x_1, \overline{x_1}, \ldots, \overline{x_n}) \)
    • Convert the given function into an appropriate form to simplify the logic.
    • Design it.
Complex Logic Gates in CMOS

• Examples (assuming only non-inverted inputs are available)
  – \( f = a \cdot b \) (AND2)
    • Design \( f = \overline{a \cdot b} \) and add an inverter at the output. (# TRs: 6)
    • Design \( f = \overline{a \cdot b} = \overline{a} + \overline{b} \) with two inverters to generate \( \overline{a} \) and \( \overline{b} \). (# TRs: 8)

  – \( f = \overline{a} \cdot b + \overline{c} \cdot d \)
    • Add two inverters to generate \( \overline{a} \) and \( \overline{c} \), then design \( f \). (# TRs: 12)

  – \( f = s \cdot a + s \cdot b \) (2:1 MUX)
Complex Logic Gates in CMOS

- Bubble pushing (how to construct a pFET network)
  \[ f = A \cdot 1 + B \cdot 0 = F(\overline{x_1}, ..., \overline{x_n}) \cdot 1 + F(x_1, ..., x_n)^D \cdot 0 \]

\[ f = a \cdot b \cdot 0 \]
\[ f = (a + b) \cdot 0 \]

\[ f = \overline{a} \cdot \overline{b} \cdot 1 \]
\[ f = (\overline{a} + \overline{b}) \cdot 1 \]
Complex Logic Gates in CMOS

- Bubble pushing (how to construct a pFET network)
  - Example

\[
\begin{align*}
  &a \quad b \\
  &c \quad d \\
  &e \quad f \\
  &a \quad b \\
  &c \quad d \\
  &e \quad f
\end{align*}
\]
Complex Logic Gates in CMOS

- **XOR**
  - \( a \oplus b = a \cdot \overline{b} + \overline{a} \cdot b = a \cdot b + \overline{a} \cdot \overline{b} \) (#TRs: 8+4(for the two inverters))

- **XNOR**
  - \( \overline{a} \oplus b = a \cdot b + \overline{a} \cdot \overline{b} = a \cdot \overline{b} + \overline{a} \cdot b \) (#TRs: 8+4(for the two inverters))
Complex Logic Gates in CMOS

• Structured logic analysis
  – Derive a Boolean equation for a given transistor-level schematic.

• Analysis methodology 1
  – Convert the nFET network into a Boolean equation (only when the pFET network is the dual of the nFET network.)
  – Notice that the nFET network passes logic 0.

• Example
  – \[ f = a \cdot b + \overline{a} \cdot \overline{b} = (\overline{a} + \overline{b}) \cdot (a + b) = a \cdot \overline{b} + \overline{a} \cdot b \]
Complex Logic Gates in CMOS

• Analysis methodology 2
  – Identify all the paths from $V_{SS}$ to the output (only when the pFET network is the dual of the nFET network.)
  – Merge them into a single Boolean equation.
  – Negate the output.

• Example
  – Path 1: $b \cdot a$
  – Path 2: $c \cdot a$
  – Merge: $b \cdot a + c \cdot a = a \cdot (b + c)$
  – Negate: $\overline{a \cdot (b + c)}$
  – $f = a \cdot (b + c)$
Pass Transistors

- **nFET**
  - $g = 0$: OFF
  - $g = 1$: ON
    - $a = 0$: $b = \text{strong } 0$
    - $a = 1$: $b = \text{weak } 1$

- **pFET**
  - $g = 1$: OFF
  - $g = 0$: ON
    - $a = 0$: $b = \text{weak } 0$
    - $a = 1$: $b = \text{strong } 1$
Transmission Gate Circuits

- Transistor circuit

- Behaviors
  - When $s = 0$: Both nFET and pFET are OFF.
  - When $s = 1$: Both nFET and pFET are ON.
    - If $x = 0$, the nFET perfectly transmits it to $y$ (nFET is good at transferring 0.)
    - If $x = 1$, the pFET perfectly transmits it to $y$ (pFET is good at transferring 1.)

- Disadvantage
  - Needs $\bar{s}$.
  - Does not restore the input signals.
Transmission Gate Circuits

- Logic design using transmission gates
  - MUX: \( f = \overline{s} \cdot x_0 + s \cdot x_1 \)
  
  ![MUX circuit diagram]

- XNOR

  ![XNOR circuit diagram]

\[ f = \overline{a \oplus b} \]
### Pass Transistors vs. Transmission Gates

<table>
<thead>
<tr>
<th></th>
<th>Pass TR.</th>
<th>Transmission Gates</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Symbols</strong></td>
<td><img src="image" alt="Symbol" /></td>
<td><img src="image" alt="Symbol" /></td>
</tr>
<tr>
<td><strong>Signal strength</strong></td>
<td>Strong 0 Weak 1</td>
<td>Weak 0 Strong 1</td>
</tr>
<tr>
<td><strong>Area</strong></td>
<td>$A$</td>
<td>$rA \ (r &gt; 1)$</td>
</tr>
<tr>
<td><strong>Control signal</strong></td>
<td>$g$</td>
<td>$g$</td>
</tr>
</tbody>
</table>
Buffer

- $Y = A$

- Buffers are used for
  - Signal restoration
  - Interconnect optimization
Tristate Inverter

• Truth table

<table>
<thead>
<tr>
<th>EN</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Z</td>
</tr>
<tr>
<td>1</td>
<td>\overline{A}</td>
</tr>
</tbody>
</table>

• Symbol & Schematic
Tristate Buffer

- Symbol

- Gate-level schematic
Sequential Circuit – D Latch

- Positive-level-sensitive D latch

<table>
<thead>
<tr>
<th>CLK</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>hold</td>
</tr>
<tr>
<td>1</td>
<td>D</td>
</tr>
</tbody>
</table>

![Diagram of D Latch behavior](image-url)
Sequential Circuit – D Latch

- Schematic
Sequential Circuit – D Flip-Flop

• Positive-edge-triggered D flip-flop

<table>
<thead>
<tr>
<th>CLK</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, 1</td>
<td>hold</td>
</tr>
<tr>
<td>↑</td>
<td>catch D</td>
</tr>
</tbody>
</table>

 CLK

<table>
<thead>
<tr>
<th>V_{DD}</th>
</tr>
</thead>
</table>

 D

<table>
<thead>
<tr>
<th>V_{DD}</th>
</tr>
</thead>
</table>

 Q

<table>
<thead>
<tr>
<th>V_{DD}</th>
</tr>
</thead>
</table>

 0

 t
Sequential Circuit – D Flip-Flop

• Schematic
Sequential Circuit

• Example
  – Inputs: \(D, \text{ARN}, \text{CLK}, \overline{\text{CLK}}\)
  – Outputs: \(Q, \overline{Q}\)