Instructors:
Wawrzynek

Lecture 10: CMOS2
Announcements

- Monday is an Academic Holiday
- New Midterm Date/Time:
  - Tue Mar 12 2024 7:00-10:00PM
  - MOFF101, VLSB2040
- Lecture Schedule change:
  - Monday Mar 11, in-class MT review
CMOS Transistors
Transistor Strength and Symmetry

1. Transistor “strength” proportional to $W/L$. In digital circuits, $L$ is almost always minimal allowed by process.

2. MOS transistors are symmetrical devices (Source and drain are interchangeable). But usually designed to be used in one direction.

For nFET, source is the node w/ the lowest voltage. For pFET source is node with highest voltage.
Circuit Layout Examples

- 2-input NAND

NAND gate layout from Lecture 3: CMOS Technology and Logic Gates. (Image by Professors Arvind and Asanovic.)

Finfet layout

from Ji Li
MOS Transistor as a Resistive Switch

MOS Transistor \[|V_{GS}| \geq |V_T|\] A Switch!

Let’s look beneath the abstraction: \[V_T \text{ and } R_{on}\]
MOSFET Threshold Voltage

Transistor "turns on" when $V_{gs}$ is $> V_t$.

$I_{off} = 0$ ???

$0.25 = V_t$

$1.2 \text{ mA} = I_{on}$

$V_{ds} = V_{dd}$
Transistor “resistance”

- Nonlinear I/V characteristic:

- But, linearizing makes all delay and power calculations simple (usually just 1st order ODEs):
ON/OFF Switch Model of MOS Transistor

\[ |V_{GS}| < |V_T| \]

\[ |V_{GS}| \geq |V_T| \]
Plot on a “Log” Scale to See “Off” Current

- Process engineers can:
  - Increase $I_{on}$ by lowering $V_t$ - but that raises $I_{off}$
  - Decrease $I_{off}$ by raising $V_t$ - but that lowers $I_{on}$

$V_{ds} = V_{dd}$

$I_{off} = 10 \text{ nA}$

$0.25 = V_t$

$1.2 \text{ mA} = I_{on}$
Latest Modern Process

Transistor channel is a raised fin.
Gate controls channel from sides and top.

Intel 22nm Process

Planar
Tri-Gate

Reduced Leakage

$V_{gs}$
$0.0 \quad 0.2 \quad 0.4 \quad 0.6 \quad 0.8$

Planar
FinFET
Gate-all-around
A More Realistic Switch

Transistors in the sub 100 nm age

|V_{GS}| < |V_T|

|V_{GS}| > |V_T|
A Logic Perspective

NMOS Transistor

\[ V_{GS} > 0 \]
\[ |V_{GS}| > |V_T| \]

Y = Z if X = 1
A Complementary Switch

Y = Z if X = 0

PMOS Transistor

V_{GS} < 0

|V_{GS}| > |V_T|

Remember, source is the node w/ the highest voltage.
The CMOS Inverter: A First Glance

$V_{in}$ $V_{out}$

$V_{DD}$

$C_{L}$

Represents the sum of all the capacitance at the output of the inverter and everything to which it connects: (drains, interconnections gate capacitance of next gate(s))
The Switch Inverter
First-Order DC Analysis*

*First-order means we will ignore Capacitance.
Switch logic
Static Logic Gates (most common and straightforward type of gate)

- At every point in time (except during the switching transients) each gate output is connected to either $V_{DD}$ or $V_{GND}$ via a low resistive path.

- The output of the gate assumes at all times the value of the Boolean function implemented by the circuit (ignoring, once again, the transient effects during switching periods).

Example: CMOS Inverter
Building logic from switches (NMOS)

Series

\[ Y = X \text{ if } A \text{ AND } B \]

Parallel

\[ Y = X \text{ if } A \text{ OR } B \]

(output undefined if condition not true)
Logic using inverting switches (PMOS)

Series

Parallel

NOR
\[ Y = X \text{ if } \overline{A} \text{ AND } \overline{B} = \overline{A + B} \]

NAND
\[ Y = X \text{ if } \overline{A} \text{ OR } \overline{B} = \overline{AB} \]

(output undefined if condition not true)
Example Gate: NAND

- pull-down network: \( G = AB \Rightarrow \text{Conduction to GND} \)
- pull-up network: \( F = \overline{A} + \overline{B} = \overline{AB} \Rightarrow \text{Conduction to } V_{\text{DD}} \)
Static Complementary CMOS

PUN and PDN are dual logic networks:
- Series connections in the PUN are parallel connections in the PDN
- Parallel connections in the PUN are series connections in the PDN

PUN and PDN functions are complements:
- Guarantees they are mutually exclusive, under all input values, one or the other is conductive, but never both!

Inverting switches (pmos) F(In₁, In₂, ..., Inₙ)

Non-Inverting switches (nmos)
Example Gate: NOR

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

Truth Table of a 2 input NOR gate

\[ \text{OUT} = \overline{A + B} \]
Complex CMOS Gate

\[ \text{OUT} = D + A \cdot (B + C) \]

\[ \text{OUT} = D \cdot A + B \cdot C \]
Graph Models for Duals

Pull-down circuit

<table>
<thead>
<tr>
<th>a</th>
<th>d</th>
<th>I_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>I_1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>c</td>
<td>I_2</td>
</tr>
</tbody>
</table>

Gnd

Pull-up circuit

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>V_{DD}</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>c</td>
</tr>
</tbody>
</table>

V_{DD}

Pull-down
Pull-up

<table>
<thead>
<tr>
<th>a</th>
<th>I_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>d</td>
</tr>
</tbody>
</table>

I_1

I_2

Gnd

Non-inverting logic

Non-Inverting switches (nmos)

Inverting switches (pmos)

Why is this a bad idea?

PUN and PDN are dual logic networks
PUN and PDN functions are complementary
Switch Limitations

- **p-type pullup**
  - Good 1
  - \( V_{DD} \rightarrow V_{DD} \)

- **n-type pullup**
  - Bad 1
  - \( V_{DD} \rightarrow V_{DD} - V_{Tn} \)

- **n-type pulldown**
  - Good 0
  - \( V_{DD} \rightarrow 0 \)

- **p-type pulldown**
  - Bad 0
  - \( V_{DD} \rightarrow |V_{Tp}| \)

Tough luck …
“Static” CMOS gates

Inverting switches (PMOS transistors)

Non-Inverting switches (NMOS transistors)

- Static CMOS gates are always inverting

AND = NAND, INV
Transmission Gate

- Transmission gates are the way to build ideal "switches" in CMOS.
- In general, for an ideal switch, both transistor types are needed:
  - nFET to pass zeros.
  - pFET to pass ones.
- The transmission gate is bi-directional (unlike logic gates).

Does not directly connect to Vdd and GND, but can be combined with logic gates or inverters to simplify many logic functions.

If \( en == 1 \) then \( A \) connects to \( B \).
Transmission-gate Multiplexor

2-to-1 multiplexor:
\[ c = sa + s'b \]

Switches simplify the implementation:

Compare the cost to logic gate implementation.

Care must be taken to not string together many pass-transistor stages. Occasionally, need to “rebuffer” with static gate.
4-to-1 Transmission-gate Mux

- The series connection of pass-transistors in each branch effectively forms the AND of s1 and s0 (or their complement).

- Compare cost to logic gate implementation

Any alternate solutions?
Alternative 4-to-1 Multiplexor

- This version has less delay from in to out.

- In both versions, care must be taken to avoid turning on multiple paths simultaneously (shorting together the inputs).
Tri-state Buffers

Tri-state Buffer:

<table>
<thead>
<tr>
<th>CE</th>
<th>IN</th>
<th>OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

“high impedance” (output disconnected)

Variations:

Inverting buffer

<table>
<thead>
<tr>
<th>CE</th>
<th>IN</th>
<th>OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Inverted enable

transmission gate provide the isolation: usually designed this way

CMOS Implementation
Tri-state Buffers

Tri-state buffers are used when multiple circuits all connect to a common node or wire. Only one circuit at a time is allowed to drive the bus. All others “disconnect” their outputs, but can “listen”.

Tri-state buffers enable “bidirectional” connections.
Tri-state Based Multiplexor

Multiplexor:

If $s=1$ then $c=a$ else $c=b$
Latches and Flip-flops

Positive Level-sensitive latch:

Latch Implementation:

When the clock is high, the latch is “transparent”, when clock is low, it holds the last seen value
Positive edge-triggered flip-flop

A flip-flop “samples” on the positive clock-edge, and then “holds” value.

Sampling latch

Holding latch

Delay in Flip-flops

- **Setup time** results from delay through the first latch.
- **Clock to Q delay** results from delay through the second latch.
Sensing: When clock is low

A flip-flop “samples” on the positive clock-edge, and then “holds” value.

Sampling latch

Holding latch

\[ \text{clk} = 0 \quad \text{clk}' = 1 \]

Ready to capture new value on clock edge

Is outputing last previous captured value.
Capture: When clock goes high

A flip-flop “samples” on the positive clock-edge, and then “holds” value.

Sampling latch

Holding latch

\[ \text{clk} = 1 \]
\[ \text{clk}' = 0 \]

Remembers value just captured.

Outputs value just captured.
Return to sensing: When clock is low

A flip-flop “samples” just before the edge, and then “holds” value.

Sampling latch

Holding latch

\[ clk = 0 \quad \text{clk}' = 1 \]

Ready to capture new value on clock edge

Continue to output captured value.
Tri-state-Inverter Latch implementation

- Commonly used in standard cell flip-flops.
- More transistors than pass-transistor version, but more robust.
- Lays out well with modern layout rules.

Negative Level-sensitive latch: