CMOS Technology

1. Qualitative MOSFET model
2. CMOS logic gates
3. CMOS design issues

Combinational Device Wish List

- Design our system to tolerate some amount of error
  ⇒ Add positive noise margins
  ⇒ VTC: gain > 1 & nonlinearity
- Lots of gain ⇒ big noise margin
- Cheap, small
- Changing voltages will require us to dissipate power, but if no voltages are changing, we'd like zero power dissipation
- Want to build devices with useful functionality (what sort of operations do we want to perform?)

MOSFETs: Gain & non-linearity

MOSFETs (metal-oxide-semiconductor field-effect transistors) are four-terminal voltage-controlled switches. Current flows between the diffusion terminals if the voltage on the gate terminal is large enough to create a conducting "channel", otherwise the mosfet is off and the diffusion terminals are not connected.

FETs as switches

The four terminals of a Field Effect Transistor (gate, source, drain and bulk) connect to conductors that generate a complicated set of electric fields in the channel region which depend on the relative voltages of each terminal.

INVERSION: A sufficiently strong vertical field will attract enough electrons to the surface to create a conducting n-type channel between the source and drain. The gate voltage when the channel first forms is called the threshold voltage -- the mosfet switch goes from "off" to "on".

CONDUCTION: If a channel exists, a horizontal field will cause a drift current from the drain to the source.
FETs come in two flavors

NFET: n-type source/drain diffusions in a p-type substrate. Positive threshold voltage; inversion forms n-type channel.

PFET: p-type source/drain diffusions in a n-type substrate. Negative threshold voltage; inversion forms p-type channel.

The use of both NFETs and PFETs – complimentary transistor types – is a key to CMOS (complementary MOS) logic families.

Connect B to GND to keep PN reverse-biased (Vp < Vn); keeps D and S insulated from B.

Connect B to VDD to keep PN reverse-biased.

NFET: n-type source/drain diffusions in a p-type substrate. Positive threshold voltage; inversion forms n-type channel.

PFET: p-type source/drain diffusions in a n-type substrate. Negative threshold voltage; inversion forms p-type channel.

CMOS Recipe

If we follow two rules when constructing CMOS circuits then we can model the behavior of the mosfets as simple switches:

Rule #1: only use NFETs in pull-down circuits (paths from output node to GND)
Rule #2: only use PFETs in pull-up circuits (paths from output node to VDD)

NFET Operating regions:
- "off": \( V_G < V_{TH, NFET} \)
- "on": \( V_G > V_{TH, NFET} \)

PFET Operating regions:
- "off": \( V_G > V_{DD} + V_{TH, PFET} \)
- "on": \( V_G < V_{DD} + V_{TH, PFET} \)

CMOS Inverter VTC

Steady state reached when \( V_{out} \) reaches value where \( I_{pu} = I_{pd} \).

When \( V_{in} \) is low, the nfet is off and the pfet is on, so current flows into the output node and \( V_{out} \) eventually reaches \( V_{DD} = V_{OH} \) at which point no more current will flow.

When \( V_{in} \) is high, the pfet is off and the nfet is on, so current flows out of the output node and \( V_{out} \) eventually reaches GND (= \( V_{OL} \)) at which point no more current will flow.

Beyond Inverters: Complementary pullups and pulldowns

We want complementary pullup and pulldown logic, i.e., the pulldown should be "on" when the pullup is "off" and vice versa.

<table>
<thead>
<tr>
<th>pullup</th>
<th>pulldown</th>
<th>( F(A_1, \ldots, A_n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>on</td>
<td>off</td>
<td>driven &quot;1&quot;</td>
</tr>
<tr>
<td>off</td>
<td>on</td>
<td>driven &quot;0&quot;</td>
</tr>
<tr>
<td>on</td>
<td>on</td>
<td>driven &quot;X&quot;</td>
</tr>
<tr>
<td>off</td>
<td>off</td>
<td>no connection</td>
</tr>
</tbody>
</table>

Since there's plenty of capacitance on the output node, when the output becomes disconnected it "remembers" its previous voltage -- at least for a while. The "memory" is the load capacitor's charge. Leakage currents will cause eventual decay of the charge (that's why DRAMs need to be refreshed).
**CMOS complements**

- Conducts when $V_{GS}$ is high
- Conducts when $V_{GS}$ is low
- Conducts when $A$ is high and $B$ is high: $A \cdot B$
- Conducts when $A$ is low or $B$ is low: $A + B = A \cdot B$
- Conducts when $A$ is low and $B$ is low: $A \cdot B = A + B$

**A pop quiz!**

What function does this gate compute?

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

**Cost:**
- $3500 per 300mm wafer
- 300mm round wafer = $\pi(150e^{-3})^2 = .07m^2$
- NAND gate = $(82)(16)(45e^{-9})^2 = 2.66e^{-12}m^2$
- $2.6e^{10}$ NAND gates/wafer ($= 100$ billion FETS!)
- Marginal cost of NAND gate: $132n$

**General CMOS gate recipe**

1. **Step 1.** Figure out pulldown network that does what you want, e.g. $F = A \cdot (B + C)$

2. **Step 2.** Walk the hierarchy replacing nFETs with pFETs, series subnets with parallel subnets, and parallel subnets with series subnets

3. **Step 3.** Combine pFET pullup network from Step 2 with nFET pulldown network from Step 1 to form fully-complementary CMOS gate.

So, what's the big deal?
A Quick Review

- A combinational device is a circuit element that has
  - one or more digital inputs
  - one or more digital outputs
  - a functional specification that details the value of each output for every possible combination of valid input values
  - a timing specification consisting (at minimum) of an upper bound \( t_{PD} \) on the required time for the device to compute the specified output values from an arbitrary set of stable, valid input values

Static discipline

Big Issue 1: Wires

Today (i.e., 100nm):
\[ \tau_{RC} = 50\text{ps/mm} \]
Implies > 1 ns to traverse a 20mm x 20mm chip
This is a long time in a 2GHz processor

Due to unavoidable delays…

Propagation delay \( (t_{PD}) \): An UPPER BOUND on the delay from valid inputs to valid outputs.

\[
\begin{align*}
\text{GOAL:} & \quad \text{minimize propagation delay!} \\
\text{ISSUE:} & \quad \text{keep Capacitances low and transistors fast}
\end{align*}
\]

\[
\begin{align*}
\text{time constant} \quad \tau = R_P D C_L \\
\text{time constant} \quad \tau = R_P U C_L
\end{align*}
\]

Contamination Delay
an optional, additional timing spec

INVALID inputs take time to propagate, too…

\[
\begin{align*}
\text{CONTAMINATION DELAY, } t_{CD} \\
\text{A LOWER BOUND on the delay from any invalid input to an invalid output}
\end{align*}
\]

Do we really need \( t_{CD} \)?
Usually not… it’ll be important when we design circuits with registers (coming soon!)
If \( t_{CD} \) is not specified, safe to assume it’s 0.
The Combinational Contract

$$A \rightarrow B$$

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>$t_{PD}$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>$t_{CD}$</td>
</tr>
</tbody>
</table>

$t_{PD}$ propagation delay  
$t_{CD}$ contamination delay

Must be $>$ $t_{CD}$  
Must be $<$ $t_{PD}$

Note:
1. No Promises during $\cdots$ 
2. Default (conservative) spec: $t_{CD} = 0$

Acyclic Combinational Circuits

If NAND gates have a $t_{PD} = 4$ nS and $t_{CD} = 1$ nS

$$t_{PD} = 12$$ nS  
$$t_{CD} = 2$$ nS

$t_{PD}$ is the minimum cumulative contamination delay over all paths from inputs to outputs

What happens in this case?

Input A alone is sufficient to determine the output

CMOS NOR:

LENIENT Combinational Device:
Output guaranteed to be valid when any combination of inputs sufficient to determine output value has been valid for at least $t_{PD}$.
Tolerates transitions -- and invalid levels -- on irrelevant inputs!

Oh yeah… one last issue

NOR:

Recall the rules for combinational devices:

Output guaranteed to be valid when all inputs have been valid for at least $t_{PD}$, and, outputs may become invalid no earlier than $t_{CD}$ after an input changes!

Many gate implementations—e.g., CMOS—adhere to even tighter restrictions.
Big Issue 2: Power

Energy dissipated = \( C V_{DD}^2 \) per cycle

Power consumed = \( f \cdot n \cdot C V_{DD}^2 \) per chip

where
- \( f \) = frequency of charge/discharge
- \( n \) = number of gates / chip

Vin moves from L to H to L

Vout moves from H to L to H

\( V_{IN} \) discharges and then recharges

\( C \)

Unfortunately...

Modern chips (UltraSparc III, Power4, Itanium 2) dissipate from 80W to 150W with a \( V_{dd} \approx 1.2V \) (Power supply current is \( \approx 100 \) Amps)

Hey: could we somehow recycle the charge?

Worse yet...
- Little room left to reduce \( V_{dd} \)
- \( nC \) and \( f \) continue to grow

Cooling challenge is like making the filament of a 100W incandescent lamp cool to the touch!

MUST computation consume energy?
(a tiny digression…)

How energy-efficient can we make a gate? It seems that switching the input to a NAND gate will always dissipate some energy…

Landauer’s Principle (1961): discarding information is what costs energy!

Bennett (1973): Use reversible logic gates, not NAND, and there’s no lower bound to energy use!

Summary

- **CMOS**
  - Only use NFETs in pull-downs, PFETs in pull-ups → mosfets behave as voltage-controlled switches
  - Series/parallel pullup and pulldown switch circuits are complementary
  - CMOS gates are naturally inverting (rising input transition can only cause falling output transition, and vice versa).
  - “Perfect” VTC (high gain, \( V_{OH} = V_{DD}, V_{OL} = GND \)) means large noise margins and no static power dissipation.

- **Timing specs**
  - \( t_{PD} \): upper bound on time from valid inputs to valid outputs
  - \( t_{CD} \): lower bound on time from invalid inputs to invalid outputs
  - If not specified, assume \( t_{CD} = 0 \)
  - Lenient gates: output unaffected by some input transitions

- Next time: logic simplification, other canonical forms