Processing Information: The Digital Abstraction

1. Making bits concrete
2. Getting bits under contract
3. Processing bits with transistors

Concrete Encodings of Information

To this point we’ve discussed encoding information using bits. But where do bits come from?

If we’re going to design a machine that manipulates information, how should that information be physically encoded?

What makes a good bit?

- cheap (we want a lot of them)
- stable (reliable, repeatable)
- ease of manipulation
  (access, transform, combine, transmit, store)
A Substrate for Computation

We can build devices for processing and representing bits using almost any physical phenomenon.

Wait! Those last ones might have potential...

- trained elephants
- engraved stone tablets
- sequences of amino acids
- polarization of a photon
- fluid pressure

Representing Information with Voltage

Representation of each point \((x, y)\) on a B&W Picture:

- 0 volts: BLACK
- 1 volt: WHITE
- 0.37 volts: 37% Gray
- etc.

Representation of a picture:
Scan points in some prescribed raster order... generate voltage waveform

How much information at each point?
Information Processing = Computation

First let's introduce some processing blocks:

- **Copy** block: $v \rightarrow \text{Copy} \rightarrow v$
- **INV** block: $v \rightarrow \text{INV} \rightarrow 1-v$

Let's build a system!

input

- Copy → INV → Copy → INV → Copy → INV → Copy → INV → output

(Reality)
Why Did Our System Fail?

Why doesn’t reality match theory?
1. COPY Operator doesn’t work right
2. INVERSION Operator doesn’t work right
3. Theory is imperfect
4. Reality is imperfect
5. Our system architecture stinks

ANSWER: all of the above!
Noise and inaccuracy are inevitable; we can’t reliably reproduce infinite information-- we must design our system to tolerate some amount of error if it is to process information reliably.

The Key to System Design

A system is a structure that is guaranteed to exhibit a specified behavior, assuming all of its components obey their specified behaviors.

How is this achieved? Contracts
Every system component will have clear obligations and responsibilities. If these are maintained we have every right to expect the system to behave as planned. If contracts are violated all bets are off.
The Digital Panacea …

Why DIGITAL?

… because it keeps the contracts simple!

The price we pay for this robustness?

All the information that we transfer between modules is only 1 crummy bit!

But, in exchange, we get a guarantee of reliable processing.

The Digital Abstraction

Real World

"Ideal" Abstract World

Volts or Electrons or Ergs or Gallons

Manufacturing Variations

Noise

Keep in mind, the world is not digital, we simply engineer it to behave that way. Furthermore, we must use real physical phenomena to implement digital designs!
Using Voltages “Digitally”

- Key idea: don’t allow “0” to be mistaken for a “1” or vice versa
- Use the same “uniform representation convention”, for every component and wire in our digital system
- To implement devices with high reliability, we outlaw “close calls” via a representation convention which forbids a range of voltages between “0” and “1”.

![](image)

CONSEQUENCE:

Notion of “VALID” and “INVALID” logic levels

A Digital Processing Element

- 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 $t_{pu}$ on the required time for the device to compute the specified output values from an arbitrary set of stable, valid input values

![](image)
A Combinational Digital System

- A set of interconnected elements is a combinational device if
  - each circuit element is combinational
  - every input is connected to exactly one output or to some vast supply of 0's and 1's
  - the circuit contains no directed cycles

  **No feedback (yet!)**

- But, in order to realize digital processing elements we have one more requirement!

Noise Margins!

Does a wire obey the static discipline?

\[ V_{in} \rightarrow V_{out} \]

(marginally valid) (invalid!)

No! A combinational device must restore marginally valid signals. It must accept marginal inputs and provide unquestionable outputs (to leave room for noise).
Digital Processing Elements

Some digital processing elements occur so frequently that we give them special names and symbols.

- **AND**
  - I will output a '1' if all my inputs are '1'

- **OR**
  - I will output a '1' if any of my inputs are '1'

- **XOR**
  - I will only output a '1' if an odd number of my inputs are '1'

- **Buffer**
  - I will copy and restore my input to my output

- **Inverter**
  - I will output the complement of my input

Static Discipline requires that our devices avoid the forbidden zones, which correspond to valid inputs but invalid outputs. Net result: combinational devices must have GAIN > 1 and be NONLINEAR.
Digital Processing Elements

Some digital processing elements occur so frequently that we give them special names and symbols.

- **Buffer**
  - Input: A
  - Output: Y

- **Inverter**
  - Input: A
  - Output: Y

- **AND**
  - Inputs: A, B
  - Output: Y

- **OR**
  - Inputs: A, B
  - Output: Y

- **XOR**
  - Inputs: A, B
  - Output: Y

Building Gates in Silicon

Semiconductors can be used to achieve both requirements of a digital system, gain and nonlinearity. Semiconductors are special elements that neither tend to “give up” or “accept” electrons in order to stabilize their outer orbitals. Pure crystalline samples of these elements tend to be nonconducting insulators. However, with the introduction of small impurities, they can be coerced into conducting an electric current. Moreover, the amount of conductance can be controlled with an “externally provided” electric field.

This property has been used to create electrically controlled devices for conducting current. Some of the first devices were made from Germanium, but over time, Silicon, has proven to be the most useful.
**PN Junctions as Insulation**

There are two kinds of impurities that we can introduce, those with mobile electrons in their outer orbitals (N-type), and those lacking mobile electrons (P-type). Both materials conduct, but their behavior is even more interesting when they come in contact. If two materials, with opposite impurities are in placed contact a junction is formed. The mobile carriers about each side of the junction move:

- **Diffusion** of holes from P to N and electrons from N to P ⇒ depletion of majority carriers in boundary region.
- **Drift** of majority carriers due to E field formed by fixed ions ⇒ acts in opposite direction of diffusion.

At equilibrium, the sum of the drift currents = sum of the diffusion currents. A depletion region is formed with a voltage across it due to induced field. At room temp, with doping concentrations of $10^{15}/\text{cm}^3$, this voltage is 0.6v. The net result is a diode:

If $V_{pn} \leq 0$, the two regions are electrically isolated.

---

**FET = Field-Effect Transistor**

We can layout junctions in various ways to take advantage of, and control their conducting and insulating properties. This creates a device called a transistor.

The four terminals of a FET (gate, source, drain and bulk) connect to conducting surfaces 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.

**Conduction:**
If a channel exists, a horizontal field will cause a drift current from the drain to the source.
Qualitative NFET Model

Process ing Information 21

Terminal with higher voltage is labeled D, the other is labeled S so \( I_{DS} \geq 0 \).

The substrate (bulk) terminal for nfets is almost always connected to ground to keep \( V_{BN} \leq 0 \).

"Linear" Operating Region

Larger \( V_{GS} \) increases drift current but also reduces vertical field component which in turn makes channel less deep. At some point, electrons are traveling as fast as possible through the channel ("velocity saturation") and the current stops growing linearly.

\[ I_{DS} \text{ proportional to } \mu_c(W/L) \]

Increasing \( V_{GS} \)

\[ \text{Increasing } V_{GS} \]

\[ V_{DS} \]

\[ I_{DS} \]

Compi20 - Spring 2005 1/20/05 01:25
Saturated Operating Region

When $V_{DS} = V_{GS} - V_{TH}$, the vertical field component is reduced and the channel is pinched-off. Electrons just keep traveling across depletion region.

$V_{DSat} = V_{GS} - V_{TH}$

$\delta L = L' - \delta L$

IDS

For your information only (not exam material)

NFET Review

Operating regions:
- cut-off: $V_{GS} < V_{TH}$
- linear: $V_{GS} \geq V_{TH}$, $V_{DS} < V_{DSat}$
- saturation: $V_{GS} \geq V_{TH}$, $V_{DS} \geq V_{DSat}$

For your information only (not exam material)
P-channel MOSFETS

Threshold voltage is negative since we need attract holes to form inversion layer.

PFET is built inside its own "substrate": a n-type well diffused into p-type bulk substrate.

The substrate (bulk) terminal for PFETs is almost always connected to power supply to keep \( V_{NB} \leq 0 \).

For your information only (not exam material)

PFET Review

Operating regions:

- **cut-off:** \( V_{GS} > V_{TH} \)
- **linear:** \( V_{GS} \leq V_{TH} \), \( V_{DS} > V_{DSAT} \)
- **saturation:** \( V_{GS} \leq V_{TH} \), \( V_{DS} \leq V_{DSAT} \)

For your information only (not exam material)
Finally… Using Transistors to Build Logic Gates!

Inverter recipe:

- **pullup**: make this connection when $V_{IN}$ near 0 so that $V_{OUT} = V_{DD}$
- **pulldown**: make this connection when $V_{IN}$ near $V_{DD}$ so that $V_{OUT} = 0$

### CMOS Inverter

- $S = power$ supply
- $V_{IN}$ vs $V_{OUT}$ for PULLUP
  - $V_{IN} = 0v$
  - $V_{IN} = 1v$
  - $V_{IN} = 2v$
  - $V_{IN} = 3v$
  - $V_{IN} = 4v$
- $S = 0v$
- $I_{PU}$ vs $V_{OUT}$ for PULLUP
- $I_{PD}$ vs $V_{OUT}$ for PULLDOWN
CMOS Inverter VTC

When both FETs are saturated, small changes in $V_{in}$ produce large changes in $V_{out}$.

Can We Build a CMOS Buffer?

Not with just two transistors!
Here’s why:

Nfets turn off when $V_{GS}$ falls below the threshold voltage $V_{th}$. So, even if the input voltage is, say, 5V, then the pullup will turn off when $V_{out}$ reaches $5V - V_{th} = 4.2V$. The pulldown will also turn off before $V_{out}$ reaches 0V.
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 “1”</td>
</tr>
<tr>
<td>off</td>
<td>on</td>
<td>driven “0”</td>
</tr>
<tr>
<td>on</td>
<td>on</td>
<td>driven “X”</td>
</tr>
<tr>
<td>off</td>
<td>off</td>
<td>no connection</td>
</tr>
</tbody>
</table>

Now you know what the “C” in CMOS stands for!

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

What a nice \( V_{OH} \) you have...

Thanks. It runs in the family...

```
conducts when \( V_{GS} \) is high
A
B
conducts when \( V_{GS} \) is low
A
B
conducts when \( A \) is high and \( B \) is high: \( A \cdot B \)
A
B
conducts when \( A \) is low or \( B \) is low: \( A + B \)
```

```
conducts when \( A \) is low or \( B \) is low: \( \overline{A} \cdot \overline{B} = \overline{A+B} \)
A
B
```

```
conducts when \( A \) is high or \( B \) is high: \( A+B \)
A
B
```

```
conducts when \( A \) is low and \( B \) is low: \( \overline{A} \cdot \overline{B} = \overline{A+B} \)
A
B
```
A Two Input Logic Gate

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></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Here's Another...

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></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
General CMOS Gate Recipe

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

Step 2. Walk the hierarchy replacing nfets with pfets, series subnets with parallel subnets, and parallel subnets with series subnets

Step 3. Combine pfet pullup network from Step 2 with nfet pulldown network from Step 1 to form fully-complementary CMOS gate.

One Last Exercise

Let's construct a gate to compute:

\[ F = \overline{A+BC} = \overline{\text{NOT(OR(A,AND(B,C)))}} \]

Step 1: The pull-down network

Step 2: The complementary pull-up network
One Last Exercise

Let's construct a gate to compute:

\[ F = \overline{A+BC} = \text{NOT}(\text{OR}(A,\text{AND}(B,C))) \]

**Step 1:** The pull-down network

**Step 2:** The complementary pull-up network

**Step 3:** Combine and Verify

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

Review

The Digital Contract and Compromise:

1) In order to design processing elements that are robust to outside influences (like noise) we choose to use only a narrow range of possible voltages to encode valid information. Moreover, valid voltage ranges are separated by regions of invalid ranges.

2) The range of valid output voltages must more demanding than the range of valid input voltages to provide a noise margin.

3) Gain and nonlinearity are required to provide noise margins.

Transistors, in particular, field-effect transistors can provide both the required gain and nonlinearity

1) Gain is used to move rapidly through the invalid voltage range
2) Nonlinearities of cut-off and saturation provide "switch-like" behavior
3) NFET transistors are good at pulling down nodes, PFETs are good at pulling nodes up
4) Complementary switch networks are used to build useful digital logic gates.