Transistor: Digital Building Blocks

- Microprocessors contain lots of transistors
  - Intel 8086 (1978): 29 thousand
  - Intel 80186 (1982): 55 thousand
  - Intel 80386 (1985): 275 thousand
  - Intel 80486 (1989): 1.1 million
  - Intel Pentium (1993): 3.1 million
  - Intel Pentium II (1998): 7.5 million
  - Intel Pentium III (2001): 45 million
  - Intel Core 2 Duo (2006): 291 million
  - Intel Quad Core i7 (2011): 1.1 billion
  - Intel 8-core Xeon (2012): 2.3 billion
**Transistor: Digital Building Blocks**

- Logically, each transistor acts as a switch
- Combined to implement logic functions (gates)
  - AND, OR, NOT
- Combined to build higher-level structures
  - Multiplexer, decoder, register, memory ...
  - Adder, multiplier ...
- Combined to build simple processor
  - LC-3

**Simple Switch Circuit**

- **Switch open:**
  - Open circuit, no current
  - Light is off
  - \( V_{out} \) is +2.9V
- **Switch closed:**
  - Short circuit across switch, current flows
  - Light is on
  - \( V_{out} \) is 0V

Switch-based circuits can easily represent two states: on/off, open/closed, voltage/no voltage.

---

**n-type MOS Transistor**

- MOS = Metal Oxide Semiconductor
- two types: n-type and p-type
- **n-type**
  - when Gate has **positive** voltage, short circuit between #1 and #2 (switch **closed**)
  - when Gate has **zero** voltage, open circuit between #1 and #2 (switch **open**)

Terminal #2 must be connected to GND (0V).

---

**p-type MOS Transistor**

- p-type is complementary to n-type
- when Gate has **positive** voltage, open circuit between #1 and #2 (switch **open**)
- when Gate has **zero** voltage, short circuit between #1 and #2 (switch **closed**)

Terminal #1 must be connected to +2.9V.
Physical Transistor

http://en.wikipedia.org/wiki/CMOS

Transistor Output (Ideal)

Logic analyzer view of waveforms

Transistor Output (Actual)

Actual waveform is not ideal!

Propagation Delay

Each gate has a propagation delay, typically a fraction of a nanosecond ($10^{-9}$ sec).

- Delays accumulate depending on the chain of gates the signals have to go through.
- Clock frequency of a processor is determined by the delay of the longest combinational path between storage elements, i.e. cycle time.
Logic Gates

- Use switch behavior of MOS transistors to implement logical functions: AND, OR, NOT.
- Digital symbols:
  - recall that we assign a range of analog voltages to each digital (logic) symbol
  - assignment of voltage ranges depends on electrical properties of transistors being used
  - typical values for "1": +5V, +3.3V, +2.9V
  - from now on we’ll use +2.9V

<table>
<thead>
<tr>
<th>Digital Values</th>
<th>&quot;0&quot;</th>
<th>Illegal</th>
<th>&quot;1&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analog Values</td>
<td>0</td>
<td>0.5</td>
<td>2.4</td>
</tr>
</tbody>
</table>

CMOS Circuit

- Complementary MOS
- Uses both n-type and p-type MOS transistors
  - p-type
    - Attached to + voltage
    - Pulls output voltage UP when input is zero
  - n-type
    - Attached to GND
    - Pulls output voltage DOWN when input is one
- For all inputs, make sure that output is either connected to GND or to +, but not both!

Inverter (NOT Gate)

<table>
<thead>
<tr>
<th>In</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 V</td>
<td>2.9 V</td>
</tr>
<tr>
<td>2.9 V</td>
<td>0 V</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>In</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Truth table:

<table>
<thead>
<tr>
<th>P-type</th>
<th>Out=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>In=0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>N-type</th>
<th>Out=0</th>
</tr>
</thead>
<tbody>
<tr>
<td>In=1</td>
<td></td>
</tr>
</tbody>
</table>

Logical Operation: OR and NOR

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

Inputs: 2 or more

Output=A+B

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>NOR</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>
AND and NAND

Inputs: 2 or more
Output = \( A \cdot B \)

NAND Gate (AND-NOT)

Note: Parallel structure on top, serial on bottom.

NOR Gate (Not of OR)

Note: Serial structure on top, parallel on bottom.

OR Gate

Add inverter to NOR.
AND Gate

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

Truth table

Add inverter to NAND.

Basic Logic Gates

- **NOT**
- **OR**
- **NOR**
- **AND**
- **NAND**

Boolean Algebra

- \( x \cdot 0 = 0 \)
- \( x \cdot 1 = x \)
- \( x \cdot \bar{x} = 0 \)
- \( x = x \)
- \( x + 0 = x \)
- \( x + 1 = 1 \)
- \( x + x = 1 \)

Remember Identify, Domination, Negation Laws from Logic!

Boolean Algebra Laws

- **Commutative**
  - \( A + B = B + A \)
  - \( A \cdot B = B \cdot A \)
- **Associative**
  - \( A + (B + C) = (A + B) + C = A + B + C \)
  - \( A \cdot (B \cdot C) = (A \cdot B) \cdot C = A \cdot B \cdot C \)
- **Distributive**
  - \( A \cdot (B + C) = A \cdot B + A \cdot C \)
  - \( A + (B \cdot C) = (A + B) \cdot (A + C) \)
Useful Simplification Identities

1. $AB + AB = A$
   - Proof: $AB + AB = A(B+B) = A(T) = A$ (Identity Law)

2. $A + AB = A$
   - Proof: $A + AB = A(1+B) = A(1) = A$ (Identity Law)

DeMorgan’s Law

- Converting AND to OR (with some help from NOT)
- Consider the following gate:

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$A + B$</th>
<th>$\overline{A + B}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

To convert AND to OR (or vice versa), invert inputs and output.

Same as $A \text{ OR } B$!

More than 2 Inputs?

- AND/OR can take any number of inputs.
  - AND = 1 if all inputs are 1.
  - OR = 1 if any input is 1.
  - Similar for NAND/NOR.
- Can implement with multiple two-input gates, or with single CMOS circuit.

Summary

- MOS transistors are used as switches to implement logic functions.
  - n-type: connect to GND, turn on (1) to pull down to 0
  - p-type: connect to +2.9V, turn on (0) to pull up to 1
- Basic gates: NOT, NOR, NAND
  - Logic functions are usually expressed with AND, OR, and NOT
- DeMorgan’s Law
  - Convert AND to OR (and vice versa) by inverting inputs and output
Building Functions from Logic Gates

- **Combinational Logic Circuit**
  - output depends only on the current inputs
  - stateless

- **Sequential Logic Circuit**
  - output depends on the sequence of inputs (past and present)
  - stores information (state) from past inputs

We’ll first look at some useful combinational circuits, then show how to use sequential circuits to store information.

Building Complex Functions

Start with a truth table. Two approaches:

- **Use gates as the building block**
  - Systematically derive the circuit
    - one row = one gate
    - minimize the gates (e.g., K-maps, QMcC)

- **Use transistors directly as building blocks**
  - Translate the truth table into a circuit for the pull-up circuit
  - Also translate it into a different circuit for the pull-down circuit
  - Both are very closely related — so transform the pull-up circuit into the pull-down circuit

Series Parallel Circuits (SPC)

- Simple (recursive) rules to define an important family of circuits
- Useful to design combinational logic
- Expose/reinforce recursive thinking
- Alternative view of basic concepts
  - Universality
  - Transformations & Equivalence

SPC Recursive Rules

- A single transistor by itself is a series-parallel circuit (with source and drain defining the “direction” of current flow, i.e., input and output)

  - **SPCs in series make an SPC.** If X and Y are SPC’s
    - Connect the output of X to input of Y
    - Input of X is the input of the new circuit
    - Output of Y is the output of the new SPC

  - **SPCs in parallel make an SPC.** If X and Y are SPC’s
    - Connect inputs of X and Y, and outputs of X and Y
    - Input of X (or Y) is the input of the new circuit
    - Output of X (or Y) is the output of the new SPC
Two more rules

- Nothing else is an SPC (the previous rules are “complete”)

- All transistors in an SPC must be of the same type
  - either all p-type
  - or all n-type
  - this is the type of the SPC

SPCs to design Boolean functions

Describe the desired circuit behavior as a truth table, and design p-type SPC as the “pull up” and an n-type SPC as the “pull down”

- For every input combination where the output is 1
  - The pull up circuit must provide a path between \( V_{DD} \) and the output
- For every input combination where the output is 0
  - The pull down circuit must provide a path between GND and the output

Pull-up Circuit

- One (parallel) branch for each row where the output is 1: each row is an “or-alternative” so branches are in parallel
- Each such branch has one transistor in series for each term in the row
  - Each term must have the specified value (AND = series)
    - If term is zero, the gate is the input signal
    - If term is 1, gate is the negation of input
- Example on doc-cam

Pull-down Circuit

- For every truth table row with a 0 output
  - Provide a path from the output to GND
  - Connect all the paths in parallel
  - Either design it from scratch like the pull-up circuit
- Or equivalently, simply construct the complement of the pull-up circuit
- Follows from DeMorgan’s Laws
Recursive Rules for Complement

- The complement of an SPC with a single transistor is the complement of the transistor.

- If an SPC is the series composition of two (or more) SPC's, X and Y:
  - Its complement is the parallel connection of the individual complements of X and Y.

- If SPC is the parallel composition of two (or more) SPC's, X and Y:
  - Its complement is the serial connection of the individual complements of X and Y.