DIGITAL LOGIC: FROM TRANSISTORS TO GATES

CE1 - Hands-on Computer Engineering
LAST WEEK:
Bread Board
RESISTOR COLOR CODE

Black  Brown  Red  Orange  Yellow  Green  Blue  Purple  Grey  White  Gold  Silver

162K Ohm ± 1%
**This week: Logic!**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>F(A,B)</th>
<th>Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>GND</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Vcc</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Vcc</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Vcc</td>
</tr>
</tbody>
</table>
THIS WEEK: CHIPS!

7400
The Transistor

- Transistor: building block of computers
- Microprocessors contain tons of transistors
  - AMD 6-core Opteron (2009): 904 million
  - Intel Core i7 Quad (2008): 731 million
  - Intel Core 2 Duo (2006): 291 million
  - Intel Itanium 2 (2003): 220 million
  - Intel Pentium 4 (2000): 42 million
  - IBM PowerPC 750FX (2002): 38 million
  - Intel 4004 (1971): 2300
THE TRANSISTOR: PAST AND PRESENT
MOORE’S LAW

“The number of active components per chip will double every 18 months.”
GPU PROCESSING COMPARED TO CPU

GFLOPS

G80GL = Quadro 5600 FX
G80 = GeForce 8800 GTX
G71 = GeForce 7900 GTX
G70 = GeForce 7800 GTX
NV40 = GeForce 6800 Ultra
NV35 = GeForce FX 5950 Ultra
NV30 = GeForce FX 5800

3.0 GHz Intel Core2 Duo

WHAT IS A TRANSISTOR?

- A switch, which can close between the source and the drain
- Changing the voltage of the gate lets you change the current flow between the source and drain (closing or opening the switch)
MATERIAL-SEMICONDUCTOR TRANSISTOR
How big is a transistor?

- If a CPU die were as big as this whole classroom…
- A transistor would be…

- Core 2 duo: 291 million transistors
- 143 sq mm
- 3 or 4 transistors would fit on the end of a ball point pen
- So about 1 sq mm
WHAT IS A TRANSISTOR?

- Logically, each transistor is used as a switch
- Combined to implement logic functions
  - AND, OR, NOT
- Combined to build higher-level structures
  - Adder, multiplexer, decoder, register, …
- Combined to build a processor
  - Pentium, Core 2 Duo, etc
**Simple Switch Circuit**

Switch open:
- No current through circuit
- 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 (nMOS)

- 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 in this example.
**DIGITAL VALUES FOR ANALOG SIGNALS**

- Use the switch behavior of MOS transistors to implement logical functions: AND, OR, NOT
- Digital symbols:
  - We assign a range of analog voltages to each digital (logic) symbol
  - Assignment of voltage ranges depends on electrical properties of transistors being used

![Digital Values Diagram]

<table>
<thead>
<tr>
<th>Digital Values</th>
<th>“0”</th>
<th>Illegal</th>
<th>“1”</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

- CMOS is Complementary Metal Oxide Semiconductor
- Uses both n-type and p-type MOS transistors
  - p-type (pMOS)
    - Attached to + voltage
    - Pulls output voltage UP when input is zero
  - n-type (nMOS)
    - Attached to GND
    - Pulls output voltage DOWN when input is one
TRUTH TABLE

- The most basic representation of a logic function
- It is a perfect induction proof - Lists the output for all possible input combinations
- How many rows of the truth table needed?

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A B ...</td>
<td>X Y ...</td>
</tr>
</tbody>
</table>

2^{#inputs}
**Truth Table: Inverter**

- Inverted signals are denoted with an overbar
- Or with a prime symbol
  - $A'$

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$Y = A'$</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
WHAT MAKES GATES GO: TRANSISTORS AS SWITCHES

<table>
<thead>
<tr>
<th>IN</th>
<th>P-type</th>
<th>N-type</th>
<th>OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Closed</td>
<td>Open</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Open</td>
<td>Closed</td>
<td>0</td>
</tr>
</tbody>
</table>
INVERTER (NOT GATE)

<table>
<thead>
<tr>
<th>In</th>
<th>Out</th>
<th>In</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 V</td>
<td>2.9 V</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2.9 V</td>
<td>0 V</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

In = 0 → Out = 1
In = 1 → Out = 0

P-type
N-type
The result of an AND operation is 1 if and only if all inputs are 1.

Depict AND by the multiplication symbol:
- $A \cdot B$

Or by lumping the signals together:
- $AB$

We don’t really build these gates…

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>$Y = A \cdot B$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
WHAT MAKES NAND GATES GO: TRANSISTORS AS SWITCHES

<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>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>
Add an inverter to a NAND.
**TRUTH TABLE: OR GATE**

- The result of an OR operation is 1 if and only if any inputs are 1
- Depict OR by the addition symbol
  - A + B

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A B</td>
<td>Y = A + B</td>
</tr>
</tbody>
</table>
NOR GATE: NOT-OR

Note: Serial structure on top, parallel on bottom.
OR GATE

Add an inverter to a NOR gate.
Simple logic functions can be expressed in a truth table form

- Why would we not want to use truth tables for large number of input functions?

To implement this function in logic we can use the basic gates AND, OR, and NOT

One method to do this is called “Sum of Products” another is “Product of Sums”

There are other methods, including ways to reduce the size of the logic
**SUM OF PRODUCTS**

- **XOR Gate**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>Y</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<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>0</td>
</tr>
</tbody>
</table>

1. Find the rows with the ‘1’ output
2. Write the product-form expression for the inputs in that row
   (0=inverted, 1=normal)
3. Combine the products in step 2 into a sum
   (OR the results of step 2)
PRODUCT OF SUMS

Procedure:
1. Find the rows with the ‘0’ output
2. Write the sum-form expression for the inputs in that row (0=normal, 1=inverted)
3. Combine the sums in step 2 into a product (AND the results of step 2)

Note: we treat 0 and 1 reverse than for SoP
LOGICAL COMPLETENESS

Can implement **ANY** truth table with AND, OR, NOT.

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Sum of Products Method:**

1. AND combinations that yield a "1" in the truth table.
2. OR the results of the AND gates.
3. Is not necessarily a minimal solution.
USING A CHIP:
CONNECTING YOUR CHIP TO POWER AND GROUND
Now you can connect the inputs to one of the gates to either power or ground.
NOW YOU CAN CONNECT THE OUTPUT TO AN LED SO YOU CAN VERIFY THE TRUTH TABLE

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>F</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>
USEFUL CHIP DIAGRAMS

7404 (NOT)

7402 (NOR)
Simple Electronics: Ring Oscillator
74HC04

Power

Ground (Gnd)
4.7MΩ 0.1μF 4.7MΩ

Capacitor (Farads)

Resistor (Ohms)
Light Emitting Diode (LED)