Digital Logic: From Transistors to Gates

Textbook Chapter 3
Announcements

• Auxiliary Website is up
  – https://classes.soe.ucsc.edu/cmpe012/Winter16/

• Office Hours
  – Tuesdays 2:30PM-4:30PM
  – Fridays 1PM-3PM

- uno
The Transistor

- Transistor: building block of computers
- Microprocessors contain tons of transistors
  - Intel Core i7-5960X (2015): 2.6 Billion
  - Qualcomm Snapdragon 810 (2015): multi billions
  - AMD 6-core Opteron (2009): 904 million
  - Intel Core i7 Quad (2008): 731 million
  - Intel Itanium 2 (2003): 220 million
  - Intel Pentium 4 (2000): 42 million
  - Intel 4004 (1971): 2300
The Transistor: Past and Present

- 1947 first point-contact transistor
Moore’s Law

Microprocessor Transistor Counts 1971-2011 & Moore’s Law

The curve shows transistor count doubling every two years.

Number of active transistors per chip will double every 18 months.”
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)
- Think of a light switch, the gate is the switch that allows electricity to flow from the source to the drain
Metal-Oxide-Semiconductor transistor

NMOS Transistor (n-channel MOSFET)

Silicon Dioxide (insulator)

source

gate

drain

gate electrode

p-type silicon

n-type silicon

n-channel

Silicon Substrate
FinFET

- Higher performance at lower voltages
- Less power
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
  - LC-3, Core i7, A9, etc
Simple switch circuit

Switch open:
- No current through circuit
- Light is off
- $V_{out}$ is $+2.9\text{V}$

Switch closed:
- Short circuit across switch
- Current flows
- Light is on
- $V_{out}$ is $0\text{V}$

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

n-type MOS transistor

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 ➔ “0” Illegal “1”

Analog Values ➔ 0 0.5 2.4 2.9 Volts
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
- Faster than using just one type
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?

\[ 2^n \]
Truth Table: Inverter

- Inverted signals are denoted with an overbar
- Or with a prime symbol
- Or with a bubble in a circuit diagram

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Y = A'</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Inverter (NOT gate)

\[
\begin{array}{c|c|c|c|c}
\text{In} & \text{Out} & \text{In} & \text{Out} \\
0 V & 2.9 V & 0 & 1 \\
2.9 V & 0 V & 1 & 0 \\
\end{array}
\]
**Truth Table: AND Gate**

- 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>$A \ B$</td>
<td>$Y = A \cdot B$</td>
</tr>
<tr>
<td>0 0</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td>0</td>
</tr>
<tr>
<td>1 0</td>
<td>0</td>
</tr>
<tr>
<td>1 1</td>
<td>1</td>
</tr>
</tbody>
</table>
NAND gate (NOT-AND)

Note: Parallel structure on top, serial on bottom.
AND gate

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<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>

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

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A B</td>
<td>Y = A + B</td>
</tr>
<tr>
<td>0 0</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td>1</td>
</tr>
<tr>
<td>1 0</td>
<td>1</td>
</tr>
<tr>
<td>1 1</td>
<td>1</td>
</tr>
</tbody>
</table>
NOR Gate: NOT-OR

Note: Serial structure on top, parallel on bottom.

<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>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>
OR gate

Add an inverter to a NOR gate.
Truth table to transistors

- So giving some arbitrary truth table, how do you go about creating a transistor-based circuit for it?
- Typically this is only done for a handful of gate types.
- Recall:
  - PMOS (with the bubbles) on top
  - NMOS (no bubbles) on bottom
  - Series structure makes AND
  - Parallel structure makes OR
Transistor based designing

• How do you get from a truth table a transistor based circuit?

• Procedure:
  1. Find the rows with the ‘1’ output
  2. Use these to form the “pull-up” part of the circuit, remember p-type are active low
  3. Find the rows with the ‘0’ output
  4. Use these to form the “pull-down” part of the circuit, remember n-type are active high

  Note: This is not optimal
Simple example

- XOR Gate – one or the other, but not both

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</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>0</td>
</tr>
</tbody>
</table>
Synthesis of an AOI Gate

- AOI means AND-OR-Invert

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

3 sets of \( p \)
5 sets of \( n \)
5 sets of \( 3 \)
Synthesis of AOI Gate

<table>
<thead>
<tr>
<th>INPUT</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>0 0 0</td>
<td>1</td>
</tr>
<tr>
<td>0 0 1</td>
<td>1</td>
</tr>
<tr>
<td>0 1 0</td>
<td>1</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0</td>
</tr>
<tr>
<td>1 0 0</td>
<td>0</td>
</tr>
<tr>
<td>1 0 1</td>
<td>0</td>
</tr>
<tr>
<td>1 1 0</td>
<td>0</td>
</tr>
<tr>
<td>1 1 1</td>
<td>0</td>
</tr>
</tbody>
</table>
Why are our circuits so big?

- The circuits for the NAND and NOR were a lot smaller than the one we just did. Why is that?
- We just used the brute force method.
- To do correctly you need the Function and the dual of the function. E.g. for NAND:
  - \( F = (AB)' = A' + B' \)
  - \( F' = ((AB)')' = AB \)