Chapter 3
Digital Logic Structures
- Part 1

Transistor: Building Block of Computers
Microprocessors contain millions of transistors
• Intel Core 2 Duo: 291 million
• AMD Barcelona: 463 million
• IBM PowerPC: 790 million

Logically, each transistor acts as a switch
Combined to implement logic functions
• AND, OR, NOT
Combined to build higher-level structures
• Adder, multiplexer, decoder, register, ...
Combined to build processor
• LC-3

Simple Switch Circuit
Switch open:
• No current through circuit
• Light is off
• \( V_{\text{out}} = +2.9 \text{V} \)

Switch closed:
• Short circuit across switch
• Current flows
• Light is on
• \( V_{\text{out}} = 0 \text{V} \)

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.
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

<table>
<thead>
<tr>
<th>Analog Values</th>
<th>0.5V</th>
<th>Digital Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 V</td>
<td>2.9 V</td>
<td>'0' illegal</td>
</tr>
<tr>
<td>2.4 V</td>
<td>2.9 V</td>
<td>'1'</td>
</tr>
</tbody>
</table>

- assignment of voltage ranges depends on electrical properties of transistors being used
  - typical values for "1": +5V, +3.3V, +2.9V, +1.1V
  - for purposes of illustration, we'll use +2.9V

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>

NOR 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>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>

Note: Serial structure on top, parallel on bottom.

OR 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>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>

NAND Gate (AND-NOT)

<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>

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

\[
\begin{array}{c|c|c}
A & B & C \\
0 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 1 \\
\end{array}
\]

Add inverter to NAND.

Basic Logic Gates

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.

Logical Completeness
Can implement ANY truth table with AND, OR, NOT.

DeMorgan's Law
Converting AND to OR (with some help from NOT)
Consider the following gate:

\[
\begin{array}{c|c|c|c|c}
A & B & A \cdot B & A + B \\
0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 \\
\end{array}
\]

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

Building Functions from Logic Gates
We've already seen how to implement truth tables using AND, OR, and NOT -- an example of combinational logic.

Combinational Logic Circuit
- output depends only on the current inputs
- Stateless

View the online lecture to see examples of some useful combinational circuits.
Summary
MOS transistors used as switches to implement logic functions.
• N-type: connect to GND, turn on (with 1) to pull down to 0
• P-type: connect to +2.9V, turn on (with 0) to pull up to 1

Basic gates: NOT, NOR, NAND
• Logic functions are usually expressed with AND, OR, and NOT

Properties of logic gates
• Completeness: can implement any truth table with AND, OR, NOT
• DeMorgan’s Law: convert AND to OR by inverting inputs and output

Building logic functions from a truth table