Digital logic: Gates, Truth tables, logic equations

Yipeng Huang

Rutgers University

April 20, 2021
Table of contents

Announcements

Transistors: The building block of computers

Combinational logic
  Basic gates
  More-than-2-input gates
Looking ahead

Class plan

1. PA5 due Monday, 4/26.
Table of contents

Announcements

Transistors: The building block of computers

Combinational logic
  Basic gates
  More-than-2-input gates
Computer organization

Layer cake

- Society
- Human beings
- Applications
- Algorithms
- High-level programming languages
  - Java, Python
- Interpreters
- Low-level programming languages
  - C, assembly
- Compilers
- Architectures
- Microarchitectures
- Sequential/combinational logic
- Transistors
- Semiconductors
- Materials science
Why binary

Everything is bits

- Each bit is 0 or 1
- By encoding/interpreting sets of bits in various ways
  - Computers determine what to do (instructions)
  - ... and represent and manipulate numbers, sets, strings, etc...
- Why bits? Electronic Implementation
  - Easy to store with bistable elements
  - Reliably transmitted on noisy and inaccurate wires
To build logic, we need switches

Vacuum tubes a.k.a. valves

Transistors

- The first transistor developed at Bell Labs, Murray Hill, New Jersey
- https://www.bell-labs.com/about/locations/murray-hill-new-jersey-usa/
MOSFETs

MOS: Metal-oxide-semiconductor
  ▶ A sandwich of conductor-insulator-semiconductor.

FET: Field-effect transistor
  ▶ Gate exerts electric field that changes conductivity of semiconductor.
NMOS, PMOS, CMOS

PMOS: P-type MOS
- positive gate voltage, acts as open circuit (insulator)
- negative gate voltage, acts as short circuit (conductor)

NMOS: N-type MOS
- positive gate voltage, acts as short circuit (conductor)
- negative gate voltage, acts as open circuit (insulator)

CMOS: Complementary MOS
- A combination of NMOS and PMOS to build logical gates such as NOT, AND, OR.
- We’ll go to slides posted in supplementary material to see how they work.
Combinational vs. sequential logic

**Combinational logic**
- No internal state nor memory
- Output depends entirely on input
- Examples: NOT, AND, NAND, OR, NOR, XOR, XNOR gates, decoders, multiplexers.

**Sequential logic**
- Has internal state (memory)
- Output depends on the inputs and also internal state
- Examples: latches, flip-flops, Mealy and Moore machines, registers, pipelines, SRAMs.
Table of contents

Announcements

Transistors: The building block of computers

Combinational logic
  Basic gates
  More-than-2-input gates
  All truth tables can be expressed in just NOT, AND and OR gates (sum-of-products form)
  Just either the NAND or the NOR gate are universal to implement all combinational logic
NOT gate

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

**Table:** Truth table for NOT gate
AND gate, NAND gate

We write AND like a product

Table: Truth table for AND gate

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>AB</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>

Table: Truth table for NAND gate

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>AB</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>
OR gate, NOR gate

![Diagram of OR gate and NOR gate]

We write OR like a sum

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$A + B$</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>

Table: Truth table for OR gate

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$\overline{A + B}$</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>

Table: Truth table for NOR gate
**XOR gate, XNOR gate**

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

**Table: Truth table for XOR gate**

\[
\begin{array}{c c c}
A & B & \overline{A} \oplus B \\
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 1 \\
\end{array}
\]

**Table: Truth table for XNOR gate**
More-than-2-input AND gate

![Diagram of a three-input AND gate]

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

Table: Truth table for three-input AND gate
More-than-2-input OR gate

\[ A + B + C \]

\[ A \quad \searrow \quad B \quad \nearrow \quad A + B \]

\[ C \quad \searrow \quad A + B + C \]

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

Table: Truth table for three-input AND gate