The basics of logic design: Combinational logic

Yipeng Huang

Rutgers University

April 21, 2022
Table of contents

Announcements

Logic gates
  Basic gates
  More-than-2-input gates

Functional completeness
  The set of logic gates \{\text{NOT, AND, OR}\} is universal
  The NAND gate is universal
  The NOR gate is universal

Combinational logic
  Decoders
  Multiplexers
  Putting all combinational logic together: Seven-segment display
Announcements

Class session plan

- 4/21, 4/26: Diving deeper: Digital logic. (CS:APP Chapter 4.2) (Recommended reading: Patterson & Hennessy, Computer organization and design, appendix on "The Basics of Logic Design." Available online via Rutgers Libraries)
- 4/28: Survey of advanced topics in computer architecture.
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

Logic gates
  Basic gates
  More-than-2-input gates

Functional completeness
  The set of logic gates \{\text{NOT, AND, OR}\} is universal
  The NAND gate is universal
  The NOR gate is universal

Combinational logic
  Decoders
  Multiplexers
  Putting all combinational logic together: Seven-segment display
NOT gate

\[ A \rightarrow \overline{A} \]

<table>
<thead>
<tr>
<th>( A )</th>
<th>( \overline{A} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

\[ \begin{array}{ccc}
A & B & AB \\
0 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 1 \\
\end{array} \]

Table: Truth table for AND gate

\[ \begin{array}{ccc}
A & B & \overline{AB} \\
0 & 0 & 1 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
\end{array} \]

Table: Truth table for NAND gate
OR gate, NOR gate

Table: Truth table for OR gate

<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 NOR 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>
XOR gate, XNOR gate

Table: Truth table for XOR gate

<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>0</td>
</tr>
</tbody>
</table>

Table: Truth table for XNOR gate

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>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>1</td>
</tr>
</tbody>
</table>
More-than-2-input AND gate

Table: Truth table for three-input AND gate

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

Table: Truth table for three-input OR gate

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

Logic gates
  Basic gates
  More-than-2-input gates

Functional completeness
  The set of logic gates \{\text{NOT, AND, OR}\} is universal
  The NAND gate is universal
  The NOR gate is universal

Combinational logic
  Decoders
  Multiplexers
  Putting all combinational logic together: Seven-segment display
The set of logic gates \{\text{NOT, AND, OR}\} is universal

\begin{align*}
\text{And} & : \quad a \land b \quad \text{out} = a \land b \\
\text{Or} & : \quad a \lor b \quad \text{out} = a \lor b \\
\text{Not} & : \quad \overline{a} \quad \text{out} = \overline{a}
\end{align*}

\textbf{Figure:} Source: CS:APP
The set of logic gates \{\text{NOT, AND, OR}\} is universal

- Any truth table can be expressed as sum of products form.
- Write each row with output 1 as a product (minterm).
- Sum the products (minterm).
- Forms a disjunctive normal form (DNF).
- \( D = \overline{A}BC + A\overline{B}C \)
- Always only needs NOT, AND, OR gates.

### Logical Completeness

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

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

1. AND combinations that yield a "1" in the truth table.
2. OR the results of the AND gates.

Sum of products
OR of AND clauses
The set of logic gates \{\text{NOT, AND, OR}\} is universal

- Any truth table can be expressed as sum of products form.
- Write each row with output 1 as a product (minterm).
- Sum the products (minterm).
- Forms a disjunctive normal form (DNF).
- Always only needs NOT, AND, OR gates.

**Figure:** Source: CS:APP
The NAND gate is universal

NOT gate as a single NAND gate

\[ A \rightleftharpoons \overline{A} \]

<table>
<thead>
<tr>
<th>A</th>
<th>( \overline{A} )</th>
<th>( AA )</th>
<th>( \overline{AA} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table: \( \overline{A} = \overline{AA} \)

AND gate as two NAND gates

\[ \begin{align*}
A & \quad \overline{AB} \\
B & \quad AB
\end{align*} \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>AB</th>
<th>( \overline{AB} )</th>
<th>( \overline{A}B )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table: \( AB = \overline{AB} \)
The NAND gate is universal

De Morgan’s Law

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

Table: $\overline{A} \overline{B} = \overline{A + B}$

OR gate as three NAND gates

$A$

$\overline{A}$

$B$

$\overline{B}$

$A \overline{B} = A + B$
The NOR gate is universal

NOT gate as a single NOR gate

\[ A \rightarrow \overline{A} \quad \overline{A} = \overline{A} \]

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

Table: \( \overline{A} = A + A \)

OR gate as two NOR gates

\[ A \]
\[ B \]

\[ AB = A + B \]

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

Table: \( A + B = \overline{A + B} \)
The NOR gate is universal

De Morgan’s Law

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

Table: $\overline{A} + \overline{B} = \overline{AB}$

AND gate as three NOR gates

$A$

$B$

$\overline{A} + \overline{B} = AB$
Announcements

Logic gates
   Basic gates
   More-than-2-input gates

Functional completeness
   The set of logic gates \{\text{NOT, AND, OR}\} is universal
   The NAND gate is universal
   The NOR gate is universal

Combinational logic
   Decoders
   Multiplexers
   Putting all combinational logic together: Seven-segment display
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.
Decoders

Takes n-bit input, uses it as an index to enable exactly one of $2^n$ outputs

Internal design of 1:2 decoder

Figure: Source: Mano & Kime
Decoders

Hierarchical design of decoder (2:4 decoder)

Takes n-bit input, uses it as an index to enable exactly one of $2^n$ outputs

Figure: Source: Mano & Kime
Decoders

Decoder (3:8)

Hierarchical design: use small decoders to build bigger decoder

Takes n-bit input, uses it as an index to enable exactly one of $2^n$ outputs

Figure: Source: Mano & Kime
Multiplexers

Using n-bit selector input, select among one of $2^n$ choices

Figure: Source: CS:APP
Multiplexers

Using n-bit selector input, select among one of $2^n$ choices

Figure: Source: CS:APP
Multiplexers

Using n-bit selector input, select among one of $2^n$ choices

Internal mux organization

**Selector Logic** (selects which input “flows through”)

**Enabler logic** (takes inputs)

Only 1 AND gate passes “1” through

AND gates “zero out” unselected $I_i$

Or gate “passes through” the non-zeroed out $I_i$

Figure: Source: Mano & Kime
Putting all combinational logic together: Seven-segment display