The basics of logic design: Sequential logic

Yipeng Huang

Rutgers University

April 23, 2024
Table of contents

Announcements

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

PA6 Demo code: directMapped read logic

Sequential logic
   SR latch
   SRAM cell
Announcements

Class session plan

- Tuesday, 4/23: 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)

- Thursday, 4/25: Survey of advanced topics in (quantum) computer architecture.

- Tuesday, 5/7: 12:00-15:00, SERC 111, closed book, closed notes, no electronic devices, no calculator final exam.
Table of contents

Announcements

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

PA6 Demo code: directMapped read logic

Sequential logic
SR latch
SRAM cell
The set of logic gates \{\text{NOT, AND, OR}\} is universal.

\[
\begin{align*}
\text{And} & : \quad a \land b \\
\text{Or} & : \quad a \lor b \\
\text{Not} & : \quad \neg a
\end{align*}
\]

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).
- \( D = \overline{A}B\overline{C} + A\overline{B}C \)
- Always only needs NOT, AND, OR gates.
- Supplementary slides example...

### 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>

- AND combinations that yield a "1" in the truth table.
- OR the results of the AND gates.
\[ E = \{0, 2, 6, \beta\} \]

input\(\{3 = 0\}\)
input\(\{2\}\)
input\(\{2\}\)
input\(\{2\}\)
input\(\{2\}\)

<table>
<thead>
<tr>
<th>Logic</th>
<th>input({1})</th>
<th>2({1})</th>
<th>2({2})</th>
<th>2({0})</th>
<th>(E)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ E = (\overline{3} \cdot \overline{2} \cdot \overline{1} \cdot \overline{0}) + (\overline{3} \cdot \overline{2} \cdot \overline{1} \cdot 0) + (\overline{3} \cdot \overline{2} \cdot 1 \cdot 0) + (\overline{3} \cdot 2 \cdot \overline{1} \cdot 0) \]

sum of products, ors of ands
$$E = i_1i_0 + i_1i_0i_2$$
$$= i_0(i_1 + i_1i_2)$$
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).
- \(D = \overline{ABC} + A\overline{B}C\)
- Always only needs NOT, AND, OR gates.
- Supplementary slides example...

Figure: Source: CS:APP
<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>x</th>
<th>y</th>
<th>$o_9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</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>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>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

char $W$, then $V$


$W[6:0]$ $W[6]$ [Comparator]

$W[5:0]$ $W[5]$ [Comparator]

$W[4:0]$ $W[4]$ [Comparator]

$W[3:0]$ $W[3]$ [Comparator]

$W[2:0]$ $W[2]$ [Comparator]

$W[1:0]$ $W[1]$ [Comparator]

cmpb
The NAND gate is universal

**NOT gate as a single NAND gate**

\[ A \xrightarrow{\text{\neg}} \overline{A} = A \xrightarrow{\text{\neg \neg}} \overline{A} \]

<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**

\[ A \quad B \xrightarrow{\text{\neg \neg}} \overline{AB} = A \quad B \xrightarrow{\text{\neg \neg \neg}} \overline{AB} \]

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

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

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

NOT gate as a single NOR gate

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

<table>
<thead>
<tr>
<th>A</th>
<th>( \overline{A} )</th>
<th>( A + A )</th>
<th>( \overline{A} + \overline{A} )</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: \( \overline{A} = A + A \)

OR gate as two NOR gates

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

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

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

$A$  \hspace{1cm} $\overline{A}$  \hspace{1cm} $\overline{A} + \overline{B} = AB$
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

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

Combinational logic
   Decoders
   Multiplexers

PA6 Demo code: directMapped read logic

Sequential logic
   SR latch
   SRAM cell
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

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

Hierarchical design of decoder (2:4 decoder)

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

Note: $A_2$ “selects” whether the 2-to-4 line decoder is active in the top half ($A_2=0$) or the bottom ($A_2=1$)

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

Internal mux organization

Using \( n \)-bit selector input, select among one of \( 2^n \) choices

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
### Table of contents

- **Announcements**

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

- **Combinational logic**
  - Decoders
  - Multiplexers

- **PA6 Demo code: directMapped read logic**

- **Sequential logic**
  - SR latch
  - SRAM cell
directMapped read logic
Table of contents

Announcements

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

Combinational logic
   Decoders
   Multiplexers

PA6 Demo code: directMapped read logic

Sequential logic
   SR latch
   SRAM cell
Sequential logic

Sequential circuit

Outputs

Inputs

Combinational circuit

next state

current state

Storage elements

Sequential circuit

Figure: Source: Mano & Kime
The simplest sequential logic element: The set/reset (SR) latch

SR latch

- Latch constructed of cross-coupled NOR gates

Figure: Source: Mano & Kime
The simplest sequential logic element: The set/reset (SR) latch

SR latch

\[
\begin{array}{ccc}
0 & R & 0 + Q = Q \\
0 & S & 0 + Q = \overline{Q} \\
\end{array}
\]

Hold previous value

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>\overline{Q}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Q</td>
<td>Q</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>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure: Source: Mano & Kime
6 transistor SRAM cell

Figure: Source: Wikimedia
Asynchronous / Synchronous circuits

Figure: Source: Prof. Jeff Ames