The Digital Abstraction

Reminders:
- No recitation tomorrow
- Lab 1 due today
- Sign up for Lab 1 checkoff meeting
Module 1: Assembly language
- From high-level programming languages to the language of the computer

Module 2: Digital design
- Combinational and sequential circuits

Module 3: Computer architecture
- Simple and pipelined processors
- Caches and the memory hierarchy

Module 4: Computer systems
- Operating system and virtual memory
- Parallelism and synchronization
6.004 Course Outline

- **Module 1: Assembly language**
  - From high-level programming languages to the language of the computer

- **Module 2: Digital design**
  - Combinational and sequential circuits

- **Module 3: Computer architecture**
  - Simple and pipelined processors
  - Caches and the memory hierarchy

- **Module 4: Computer systems**
  - Operating system and virtual memory
  - Parallelism and synchronization
Why Learn Hardware Design?

- Designing cutting-edge systems requires intimate knowledge of hardware
Why Learn Hardware Design?

- Designing cutting-edge systems requires intimate knowledge of hardware

- Chip fabrication technology is reaching its limits → more companies are building custom hardware

Hennessy and Patterson, Turing Award Lecture
CACM 2019, https://dl.acm.org/citation.cfm?id=3282307
Why Learn Hardware Design?

- Designing cutting-edge systems requires intimate knowledge of hardware

- Chip fabrication technology is reaching its limits → more companies are building custom hardware

- You are likely to use and design specialized hardware, not just general-purpose processors!

Hennessy and Patterson, Turing Award Lecture
CACM 2019, https://dl.acm.org/citation.cfm?id=3282307

September 19, 2019
MIT 6.004 Fall 2019
The Digital Abstraction

Building Digital Systems in an Analog World
Analog vs. Digital Systems

- Analog systems represent and process information using continuous signals
  - e.g., voltage, current, temperature, pressure, ...
Analog vs. Digital Systems

- Analog systems represent and process information using **continuous signals**
  - e.g., voltage, current, temperature, pressure, ...

![Graph showing voltage over time](image-url)
Analog vs. Digital Systems

- Analog systems represent and process information using **continuous signals**
  - e.g., voltage, current, temperature, pressure, ...

- Digital systems represent and process information using **discrete symbols**
Analog vs. Digital Systems

- Analog systems represent and process information using continuous signals
  - e.g., voltage, current, temperature, pressure, ...

- Digital systems represent and process information using discrete symbols
  - Typically binary symbols (bits)
Analog vs. Digital Systems

- Analog systems represent and process information using **continuous signals**
  - e.g., voltage, current, temperature, pressure, ...

- Digital systems represent and process information using **discrete symbols**
  - Typically binary symbols (bits)
  - Encoded using ranges of a physical quantity (e.g., voltage)
Analog vs. Digital Systems

- Analog systems represent and process information using **continuous signals**
  - e.g., voltage, current, temperature, pressure, ...

- Digital systems represent and process information using **discrete symbols**
  - Typically binary symbols (bits)
  - Encoded using ranges of a physical quantity (e.g., voltage)
Analog vs. Digital Systems

- Analog systems represent and process information using **continuous signals**
  - e.g., voltage, current, temperature, pressure, ...

- Digital systems represent and process information using **discrete symbols**
  - Typically binary symbols (bits)
  - Encoded using ranges of a physical quantity (e.g., voltage)
Analog vs. Digital Systems

- Analog systems represent and process information using **continuous signals**
  - e.g., voltage, current, temperature, pressure, ...

- Digital systems represent and process information using **discrete symbols**
  - Typically binary symbols (bits)
  - Encoded using ranges of a physical quantity (e.g., voltage)

Digital systems tolerate noise
Example: Analog Audio Equalizer

Input: Voltage signal representing sound pressure

Expected output: Frequency-equalized voltage signal
Example: Analog Audio Equalizer

Input: Voltage signal representing sound pressure

Expected output: Frequency-equalized voltage signal
Example: Analog Audio Equalizer

Input: Voltage signal representing sound pressure

Expected output: Frequency-equalized voltage signal

Does output match expected output?
Example: Analog Audio Equalizer

Input: Voltage signal representing sound pressure

Filters

Bass gain
Mid gain
Treble gain

Expected output: Frequency-equalized voltage signal

Does output match expected output? Not quite!
Example: Analog Audio Equalizer

Input: Voltage signal representing sound pressure

Expected output: Frequency-equalized voltage signal

Does output match expected output?  Not quite!

Why or why not?
Example: Analog Audio Equalizer

Input: Voltage signal representing sound pressure

Expected output: Frequency-equalized voltage signal

Does output match expected output? **Not quite!**

Why or why not? **Noise**
Example: Analog Audio Equalizer

Input: Voltage signal representing sound pressure

Expected output: Frequency-equalized voltage signal

Does output match expected output? Not quite!

Why or why not? Noise
Manufacturing variations
Example: Analog Audio Equalizer

Input: Voltage signal representing sound pressure

Bass gain

Mid gain

Treble gain

Expected output: Frequency-equalized voltage signal

Does output match expected output? Not quite!

Why or why not? Noise

Manufacturing variations

Components degrade over time
Example: Analog Audio Equalizer

Input: Voltage signal representing sound pressure

<table>
<thead>
<tr>
<th>Filters</th>
<th>Bass gain</th>
<th>Mid gain</th>
<th>Treble gain</th>
</tr>
</thead>
</table>

Expected output: Frequency-equalized voltage signal

Does output match expected output? Not quite!

Why or why not? Noise
Manufacturing variations
Components degrade over time

...
The Digital Abstraction

Real World

Volts or Amperes or Lumens

Manufacturing variations

Noise

"Ideal" Abstract World

0/1 Bits

Keep in mind that the world is not digital, we would simply like to engineer it to behave that way. In the end we must use real physical phenomena to implement digital designs!
Using Voltages “Digitally”

- Key idea: Encode two symbols, “0” and “1” (1 bit)
- Use the same convention for every component and wire in our digital system

Attempt #1:

\[ V_{TH} \]
Using Voltages “Digitally”

- Key idea: Encode two symbols, “0” and “1” (1 bit)
- Use the same convention for every component and wire in our digital system

Attempt #1:

\[ V < V_{TH} \]

interpreted as “0”

\[ V_{TH} \]

volts
Using Voltages “Digitally”

- Key idea: Encode two symbols, “0” and “1” (1 bit)
- Use the same convention for every component and wire in our digital system

Attempt #1:

\[ V < V_{TH} \] interpreted as “0”
\[ V_{TH} \]
\[ V \geq V_{TH} \] interpreted as “1”

volts
Using Voltages “Digitally”

• Key idea: Encode two symbols, “0” and “1” (1 bit)
• Use the same convention for every component and wire in our digital system

Attempt #1:

\[ V < V_{TH} \] interpreted as “0”
\[ V_{TH} \]
\[ V \geq V_{TH} \] interpreted as “1”

Not quite correct. Why?
Using Voltages “Digitally”

- Key idea: Encode two symbols, “0” and “1” (1 bit)
- Use the same convention for every component and wire in our digital system

Attempt #1:

\[ V < V_{\text{TH}} \] interpreted as “0”  \[ V_{\text{TH}} \] \[ V \geq V_{\text{TH}} \] interpreted as “1”

Not quite correct. Why? Hard to distinguish \( V_{\text{TH}} - \varepsilon \) from \( V_{\text{TH}} + \varepsilon \)
Using Voltages “Digitally”

- Key idea: Encode two symbols, “0” and “1” (1 bit)
- Use the same convention for every component and wire in our digital system

Attempt #1:

\[ V < V_{TH} \] interpreted as “0”
\[ V \geq V_{TH} \] interpreted as “1”

\[ ✗ \]

Not quite correct. Why? Hard to distinguish \( V_{TH} - \epsilon \) from \( V_{TH} + \epsilon \)

Attempt #2:

\[ V_L \]
\[ V_H \]
Using Voltages “Digitally”

- Key idea: Encode two symbols, “0” and “1” (1 bit)
- Use the same convention for every component and wire in our digital system

Attempt #1:

\[ V < V_{TH} \quad \text{interpreted as “0”} \]
\[ V_{TH} \quad \text{interpreted as “1”} \]
\[ V \geq V_{TH} \]

\[ V \leq V_L \quad \text{interpreted as “0”} \]
\[ V_L \]
\[ V_H \]

Not quite correct. Why? **Hard to distinguish** \( V_{TH} - \epsilon \) from \( V_{TH} + \epsilon \)

Attempt #2:
Using Voltages “Digitally”

- Key idea: Encode two symbols, “0” and “1” (1 bit)
- Use the same convention for every component and wire in our digital system

Attempt #1:

\[ V < V_{TH} \] interpreted as “0” \[ V_{TH} \] \[ V \geq V_{TH} \] interpreted as “1”

Not quite correct. Why? Hard to distinguish \( V_{TH} - \epsilon \) from \( V_{TH} + \epsilon \)

Attempt #2:

\[ V \leq V_L \] interpreted as “0” \[ V_L \] \[ V_H \] \[ V \geq V_H \] interpreted as “1”

volts
Using Voltages “Digitally”

- Key idea: Encode two symbols, “0” and “1” (1 bit)
- Use the same convention for every component and wire in our digital system

**Attempt #1:**

\[ V < V_{TH} \] interpreted as “0”
\[ V_{TH} \]
\[ V \geq V_{TH} \] interpreted as “1”

\( V \leq V_L \) interpreted as “0”
\( V_L < V < V_H \)”Undefined”
\( V_H \)
\( V \geq V_H \) interpreted as “1”

Not quite correct. Why? Hard to distinguish \( V_{TH} - \varepsilon \) from \( V_{TH} + \varepsilon \)
Using Voltages “Digitally”

• Key idea: Encode two symbols, “0” and “1” (1 bit)
• Use the same convention for every component and wire in our digital system

Attempt #1:

\[ V < V_{TH} \quad \text{interpreted as “0”} \quad V_{TH} \quad V \geq V_{TH} \quad \text{interpreted as “1”} \]

\[ \text{Not quite correct. Why? Hard to distinguish } V_{TH}-\epsilon \text{ from } V_{TH}+\epsilon \]

Attempt #2:

\[ V \leq V_L \quad \text{interpreted as “0”} \quad V_L \quad V_L < V < V_H \quad \text{“Undefined”} \quad V_H \quad V \geq V_H \quad \text{interpreted as “1”} \]
Upstream device transmits a signal at $V_{L-\epsilon}$, a valid “0”. Noise on the wire causes the downstream device to receive $V_{L+\epsilon}$, which is undefined.
Will This System Work?

Valid “0”: $V_L - \varepsilon$

$V_L + \varepsilon$: not a valid signal

Upstream device transmits a signal at $V_L - \varepsilon$, a valid “0”. Noise on the wire causes the downstream device to receive $V_L + \varepsilon$, which is undefined.

*How can we address this?*
Will This System Work?

Valid “0”: $V_L - \varepsilon$

Digital device

V L + ε: not a valid signal

Digital device

Upstream device transmits a signal at $V_L - \varepsilon$, a valid “0”. Noise on the wire causes the downstream device to receive $V_L + \varepsilon$, which is undefined.

How can we address this?

Output voltages should use narrower ranges, so that signal will still be valid when it reaches an input even if there is noise.
Noise Margins
Noise Margins

Proposed fix: Different specifications for inputs and outputs
Noise Margins

Proposed fix: Different specifications for inputs and outputs

• Digital output: “0” \( \leq V_{OL} \), “1” \( \geq V_{OH} \)
Noise Margins

Proposed fix: Different specifications for inputs and outputs

• Digital output: “0” ≤ $V_{OL}$, “1” ≥ $V_{OH}$
Noise Margins

Proposed fix: Different specifications for inputs and outputs

- Digital output: “0” ≤ $V_{OL}$, “1” ≥ $V_{OH}$
- Digital input: “0” ≤ $V_{IL}$, “1” ≥ $V_{IH}$
Noise Margins

Proposed fix: Different specifications for inputs and outputs

- Digital output: “0” \( \leq V_{OL} \), “1” \( \geq V_{OH} \)
- Digital input: “0” \( \leq V_{IL} \), “1” \( \geq V_{IH} \)
Proposed fix: Different specifications for inputs and outputs

- Digital output: “0” ≤ \( V_{OL} \), “1” ≥ \( V_{OH} \)
- Digital input: “0” ≤ \( V_{IL} \), “1” ≥ \( V_{IH} \)
- \( V_{OL} < V_{IL} < V_{IH} < V_{OH} \)
Noise Margins

Proposed fix: Different specifications for inputs and outputs

- Digital output: “0” ≤ $V_{OL}$, “1” ≥ $V_{OH}$
- Digital input: “0” ≤ $V_{IL}$, “1” ≥ $V_{IH}$
- $V_{OL} < V_{IL} < V_{IH} < V_{OH}$
Noise Margins

Proposed fix: Different specifications for inputs and outputs

- Digital output: “0” ≤ $V_{OL}$, “1” ≥ $V_{OH}$
- Digital input: “0” ≤ $V_{IL}$, “1” ≥ $V_{IH}$
- $V_{OL} < V_{IL} < V_{IH} < V_{OH}$

A digital device accepts marginal inputs and provides unquestionable outputs (to leave room for noise).
Digital Systems are Restorative

Analog systems: Noise accumulates

$V_I \xrightarrow{\varepsilon_1} f \xrightarrow{\varepsilon_2} g$
Digital Systems are Restorative

Analog systems: Noise accumulates

\[ V_I \xrightarrow{\varepsilon_1} V_I + \varepsilon_1 \xrightarrow{\varepsilon_2} g \]

September 19, 2019

MIT 6.004 Fall 2019
Digital Systems are Restorative

Analog systems: Noise accumulates

\[ V_I + \varepsilon_1 \xrightarrow{\text{f}} f(V_I + \varepsilon_1) \xrightarrow{\varepsilon_2} g \]
Digital Systems are Restorative

Analog systems: Noise accumulates

\[ V_I + \varepsilon_1 \xrightarrow{f} f(V_I + \varepsilon_1) \xrightarrow{\varepsilon_2} f(V_I + \varepsilon_1) + \varepsilon_2 \xrightarrow{g} \]
Digital Systems are Restorative

Analog systems: Noise accumulates

\[ V_I, V_I + \epsilon_1 \xrightarrow{f} f(V_I + \epsilon_1), f(V_I + \epsilon_1) + \epsilon_2 \xrightarrow{g} g(f(V_I + \epsilon_1) + \epsilon_2) \]
Digital Systems are Restorative

Analog systems: Noise accumulates

Digital systems: Noise is canceled at each stage
Digital Systems are Restorative

Analog systems: Noise accumulates

Digital systems: Noise is canceled at each stage
Digital Systems are Restorative

Analog systems: Noise accumulates

\[ V_I \xrightarrow{\varepsilon_1} V_I + \varepsilon_1 \xrightarrow{f} f(V_I + \varepsilon_1) \xrightarrow{\varepsilon_2} f(V_I + \varepsilon_1) + \varepsilon_2 \xrightarrow{g} g(f(V_I + \varepsilon_1) + \varepsilon_2) \]

Digital systems: Noise is canceled at each stage

\[ V_I \xrightarrow{\varepsilon_1} V_I + \varepsilon_1 \xrightarrow{f} f(V_I) \xrightarrow{\varepsilon_2} g(f(V_I)) \]
Digital Systems are Restorative

Analog systems: Noise accumulates

\[ V_I \xrightarrow{\varepsilon_1} V_I + \varepsilon_1 \xrightarrow{f} f(V_I + \varepsilon_1) \xrightarrow{\varepsilon_2} f(V_I + \varepsilon_1) + \varepsilon_2 \xrightarrow{g} g(f(V_I + \varepsilon_1) + \varepsilon_2) \]

Digital systems: Noise is canceled at each stage

\[ V_I \xrightarrow{\varepsilon_1} V_I + \varepsilon_1 \xrightarrow{f} f(V_I) \xrightarrow{\varepsilon_2} f(V_I) + \varepsilon_2 \xrightarrow{g} g(f(V_I) + \varepsilon_2) \]
Digital Systems are Restorative

Analog systems: Noise accumulates

\[ V_{I} + \varepsilon_{1} \xrightarrow{f} f(V_{I} + \varepsilon_{1}) + \varepsilon_{2} \xrightarrow{g} g(f(V_{I} + \varepsilon_{1}) + \varepsilon_{2}) \]

Digital systems: Noise is canceled at each stage

\[ V_{I} + \varepsilon_{1} \xrightarrow{f} f(V_{I}) + \varepsilon_{2} \xrightarrow{g} g(f(V_{I})) \]
Digital Systems are Restorative

Analog systems: Noise accumulates

\[ V_I \xrightarrow{\varepsilon_1} V_I + \varepsilon_1 \xrightarrow{f} f(V_I + \varepsilon_1) \xrightarrow{\varepsilon_2} f(V_I + \varepsilon_1) + \varepsilon_2 \xrightarrow{g} g(f(V_I + \varepsilon_1) + \varepsilon_2) \]

Digital systems: Noise is canceled at each stage

\[ V_I \xrightarrow{\varepsilon_1} V_I + \varepsilon_1 \xrightarrow{f} f(V_I) \xrightarrow{\varepsilon_2} f(V_I) + \varepsilon_2 \xrightarrow{g} g(f(V_I)) \]

Intuitively, canceling noise requires *active components*, i.e., components that inject energy into the system.
Buffer: A simple digital device that copies its input value to its output.

Voltage Transfer Characteristic (VTC): Plot of $V_{out}$ vs. $V_{in}$ where each measurement is taken after any transients have died out.


**Voltage Transfer Characteristic**

*Buffer*: A simple digital device that copies its input value to its output.

**Voltage Transfer Characteristic (VTC):**
Plot of $V_{out}$ vs. $V_{in}$ where each measurement is taken after any transients have died out.
Buffer: A simple digital device that copies its input value to its output.

**Voltage Transfer Characteristic (VTC):** Plot of $V_{out}$ vs. $V_{in}$ where each measurement is taken after any transients have died out.

VTC must avoid the shaded regions (aka “forbidden zones”), which correspond to valid inputs but invalid outputs.
**Buffer**: A simple digital device that copies its input value to its output.

**Voltage Transfer Characteristic (VTC)**: Plot of $V_{out}$ vs. $V_{in}$ where each measurement is taken after any transients have died out.

*Note: VTC does not tell you anything about how fast a device is — it measures static behavior, not dynamic behavior.*

VTC must avoid the shaded regions (aka “forbidden zones”), which correspond to *valid* inputs but *invalid* outputs.
Voltage Transfer Characteristic

$V_{out}$

$V_{OH}$

$V_{OL}$

$V_{IH}$

$V_{IL}$

$V_{in}$
Voltage Transfer Characteristic

1) Note the center white region is taller than it is wide ($V_{OH} - V_{OL} > V_{IH} - V_{IL}$). Net result: device must have $GAIN > 1$ and thus be ACTIVE
1) Note the center white region is taller than it is wide (V_{OH}-V_{OL} > V_{IH}-V_{IL}). Net result: device must have GAIN > 1 and thus be ACTIVE.

2) Note the VTC can do anything when V_{IL} < V_{IN} < V_{IH}
Types of Digital Circuits
Types of Digital Circuits

- Combinational circuits
  - Do not have memory
  - Each output is a function of current input values
Types of Digital Circuits

- Combinational circuits
  - Do not have memory
  - Each output is a function of current input values
  - Examples:
Types of Digital Circuits

- Combinational circuits
  - Do not have memory
  - Each output is a function of current input values
  - Examples: Inverter

![Diagram of digital circuit with input and output connections]
Types of Digital Circuits

- Combinational circuits
  - Do not have memory
  - Each output is a function of current input values
  - Examples: Inverter

\[ 0 \rightarrow 1 \]
Types of Digital Circuits

- Combinational circuits
  - Do not have memory
  - Each output is a function of current input values
  - Examples: Inverter
    
    | Input | Output |
    |-------|--------|
    | 0     | 1      |
    | 1     | 0      |

Digital inputs | Digital outputs
Types of Digital Circuits

- Combinational circuits
  - Do not have memory
  - Each output is a function of current input values
  - Examples:

    | Input | Inverter | AND |
    |-------|----------|-----|
    | 0     | 1        |     |
    | 1     | 0        |     |
Types of Digital Circuits

- Combinational circuits
  - Do not have memory
  - Each output is a function of current input values
  - Examples:

    | Digital inputs | Digital outputs |
    |----------------|-----------------|
    | Inverter       | AND             |
    | 0  →  1        | 1 →  0          |
    | 1  →  0        | Output is 1 if both inputs are 1, 0 otherwise |
Types of Digital Circuits

- Combinational circuits
  - Do not have memory
  - Each output is a function of current input values
  - Examples:
    
    | Inverter | AND |
    |----------|-----|
    | 0 1      | Output is 1 if both inputs are 1, 0 otherwise |
    | 1 0      | 0 0 otherwise |

- Sequential circuits
Types of Digital Circuits

- **Combinational circuits**
  - Do not have memory
  - Each output is a function of current input values
  - Examples:
    - Inverter
      \[
      \begin{array}{c|c}
      0 & 1 \\
      1 & 0 \\
      \end{array}
      \]
    - AND
      \[
      \begin{array}{c|c|c}
      & 0 & 1 \\
      0 & 0 & 0 \\
      1 & 0 & 1 \\
      \end{array}
      \]
      Output is 1 if both inputs are 1, 0 otherwise

- **Sequential circuits**
  - Have memory, i.e., *state*
  - Each output depends on current state + current inputs
Introduction to Combinational Circuits
A combinational device is a circuit element that has
• one or more digital inputs
• one or more digital outputs
• a functional specification that details the value of each output for every possible combination of valid input values
• a timing specification consisting (at a minimum) of a propagation delay ($t_{PD}$): an upper bound on the required time to produce valid, stable output values from an arbitrary set of valid, stable input values

Output a “1” if at least 2 out of 3 of my inputs are a “1”. Otherwise, output “0”.

I will generate a valid output in no more than 2 minutes after seeing valid inputs.
A combinational device is a circuit element that has:

- one or more digital inputs
- one or more digital outputs
- a functional specification that details the value of each output for every possible combination of valid input values
- a timing specification consisting (at a minimum) of a propagation delay ($t_{PD}$): an upper bound on the required time to produce valid, stable output values from an arbitrary set of valid, stable input values

Static discipline

Output a “1” if at least 2 out of 3 of my inputs are a “1”. Otherwise, output “0”.

I will generate a valid output in no more than 2 minutes after seeing valid inputs.
Composing Combinational Devices

A set of interconnected elements is a combinational device if

- each circuit element is combinational
- every input is connected to exactly one output or to a constant (0 or 1)
- the circuit contains no directed cycles
Example: Is This a Combinational Device?

A, B, and C are combinational devices. Is the following circuit a combinational device?
Example: Is This a Combinational Device?

A, B, and C are combinational devices. Is the following circuit a combinational device?

Does it have digital inputs?
Example: Is This a Combinational Device?

A, B, and C are combinational devices. Is the following circuit a combinational device?

Does it have digital inputs? Yes
Example: Is This a Combinational Device?

A, B, and C are combinational devices. Is the following circuit a combinational device?

Does it have digital inputs?  Yes
Does it have digital outputs?
Example: Is This a Combinational Device?

A, B, and C are combinational devices. Is the following circuit a combinational device?

![Circuit Diagram]

Does it have digital inputs? Yes
Does it have digital outputs? Yes
Example: Is This a Combinational Device?

A, B, and C are combinational devices. Is the following circuit a combinational device?

Does it have digital inputs? Yes
Does it have digital outputs? Yes
Can you derive a functional description?
Example: Is This a Combinational Device?

A, B, and C are combinational devices. Is the following circuit a combinational device?

Does it have digital inputs? Yes
Does it have digital outputs? Yes
Can you derive a functional description?
Example: Is This a Combinational Device?

A, B, and C are combinational devices. Is the following circuit a combinational device?

Does it have digital inputs? Yes
Does it have digital outputs? Yes
Can you derive a functional description?

\[ W = f_C(f_A(X, Y), f_B(f_A(X, Y), Z)) \]
Example: Is This a Combinational Device?

A, B, and C are combinational devices. Is the following circuit a combinational device?

Does it have digital inputs? Yes
Does it have digital outputs? Yes
Can you derive a functional description?

\[ W = f_C(f_A(X, Y), f_B(f_A(X, Y), Z)) \]

Can you derive a \( t_{PD} \)?
Example: Is This a Combinational Device?

A, B, and C are combinational devices. Is the following circuit a combinational device?

Does it have digital inputs? Yes
Does it have digital outputs? Yes
Can you derive a functional description?

\[ W = f_C(f_A(X, Y), f_B(f_A(X, Y), Z)) \]

Can you derive a \( t_{PD} \)?
Example: Is This a Combinational Device?

A, B, and C are combinational devices. Is the following circuit a combinational device?

Does it have digital inputs? Yes
Does it have digital outputs? Yes
Can you derive a functional description?

\[ W = f_C(f_A(X, Y), f_B(f_A(X, Y), Z)) \]

Can you derive a \( t_{PD} \)?

\[ t_{PD} = t_{PD,A} + t_{PD,B} + t_{PD,C} \]
Functional Specifications

- There are many ways to specify the function of a combinational device.

\[
\begin{array}{c}
A \\
B \\
C \\
\end{array} \rightarrow Y
\]

If C is 0 then copy A to Y, otherwise copy B to Y.
There are many ways to specify the function of a combinational device.

We will use two systematic approaches:

If C is 0 then copy A to Y, otherwise copy B to Y.
Functional Specifications

- There are many ways to specify the function of a combinational device

- We will use two systematic approaches:
  - Truth tables enumerate the output values for all possible combinations of input values
There are many ways to specify the function of a combinational device.

We will use two systematic approaches:

- **Truth tables** enumerate the output values for all possible combinations of input values.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Y</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>0</td>
</tr>
</tbody>
</table>
There are many ways to specify the function of a combinational device

We will use two systematic approaches:

- **Truth tables** enumerate the output values for all possible combinations of input values
There are many ways to specify the function of a combinational device.

We will use two systematic approaches:

- Truth tables enumerate the output values for all possible combinations of input values.
Functional Specifications

There are many ways to specify the function of a combinational device.

We will use two systematic approaches:

- **Truth tables** enumerate the output values for all possible combinations of input values.

- **Boolean expressions** are equations containing binary (0/1) variables and three operations: AND (\(\cdot\)), OR (\(+\)), and NOT (overbar).

<table>
<thead>
<tr>
<th>C</th>
<th>B</th>
<th>A</th>
<th>Y</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>0</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>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>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
There are many ways to specify the function of a combinational device.

We will use two systematic approaches:

- **Truth tables** enumerate the output values for all possible combinations of input values.

- **Boolean expressions** are equations containing binary (0/1) variables and three operations: AND (\(\cdot\)), OR (+), and NOT (overbar).

\[
Y = \overline{C} \cdot A + C \cdot B
\]
Functional Specifications

- There are many ways to specify the function of a combinational device.

- We will use two systematic approaches:
  - Truth tables enumerate the output values for all possible combinations of input values.
  - Boolean expressions are equations containing binary (0/1) variables and three operations: AND (\(\cdot\)), OR (+), and NOT (overbar)

\[
Y = \overline{C} \cdot A + C \cdot B
\]

Any combinational function can be specified as a truth table or Boolean expression (next lecture)
Timing Specifications

- Propagation delay ($t_{PD}$): An upper bound on the delay from valid inputs to valid outputs
Timing Specifications

- Propagation delay ($t_{PD}$): An upper bound on the delay from valid inputs to valid outputs

![Timing Diagram]
Timing Specifications

- Propagation delay ($t_{PD}$): An upper bound on the delay from valid inputs to valid outputs.
Timing Specifications

- Propagation delay ($t_{PD}$): An upper bound on the delay from valid inputs to valid outputs
Timing Specifications

- Propagation delay ($t_{PD}$): An upper bound on the delay from valid inputs to valid outputs.
Timing Specifications

- Propagation delay ($t_{PD}$): An upper bound on the delay from valid inputs to valid outputs

\[ \text{Propagating delay } (t_{PD}) \leq t_{PD} \leq t_{PD} \]

**Goal:** Minimize $t_{PD}$!
### Timing Specifications

- **Propagation delay (t\(_{PD}\))**: An upper bound on the delay from valid inputs to valid outputs.

- **Contamination delay (t\(_{CD}\))**: A lower bound on the delay from invalid inputs to invalid outputs.
  - Used later (for sequential logic), can ignore for now.

---

![Timing Diagram](image-url)

**Goal**: Minimize \(t_{PD}\)!
The Combinational Contract

A → B

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

t_{PD} propagation delay
t_{CD} contamination delay

No promises during ✴✴✴✴
The Combinational Contract

A \rightarrow B

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

t_{PD} \text{ propagation delay}

t_{CD} \text{ contamination delay}

A

B

Must be ___________

No promises during XXXXX
The Combinational Contract

\[ \begin{array}{c|c|c}
A & B & t_{PD} \\
0 & 1 & t_{CD} \\
1 & 0 &
\end{array} \]

\( t_{PD} \) propagation delay
\( t_{CD} \) contamination delay

\textbf{Must be} \[\leq t_{PD}\]

No promises during \[\text{xxxxxx}\]
The Combinational Contract

A \rightarrow B

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

\( t_{PD} \) propagation delay
\( t_{CD} \) contamination delay

Must be ____________

Must be \( \leq t_{PD} \)

No promises during \( \cdots \)
The Combinational Contract

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

- \( t_{PD} \): propagation delay
- \( t_{CD} \): contamination delay

A ➔ B

Must be \( \geq t_{CD} \)

Must be \( \leq t_{PD} \)

No promises during \( xxxx \)
Summary

- Digital systems tolerate noise
Summary

- Digital systems tolerate noise
- Digital encoding
  - Valid voltage levels for representing "0" and "1"
  - Undefined range avoids mistaking "0" for "1" and vice versa
  - Noise margins require tougher standards for outputs than for inputs
Summary

- Digital systems tolerate noise
- Digital encoding
  - Valid voltage levels for representing “0” and “1”
  - Undefined range avoids mistaking “0” for “1” and vice versa
  - Noise margins require tougher standards for outputs than for inputs
- Combinational devices
  - Have Tinkertoy-set simplicity, modularity
  - Predictable composition: “parts work → whole thing works”
  - Must obey static discipline
    - Digital inputs & outputs: restores marginal input voltages
    - Complete functional specification
    - Valid inputs lead to valid outputs in bounded time
Thank you!

Next lecture: Boolean Algebra