Two ways to think about logic signals

- **Fixed logic convention**
  - High voltage always means 1, TRUE, Asserted
  - Low voltage always means 0, FALSE, Negated

- **Mixed Logic convention**
  - Can have High and Low true signals
  - High true signals means that high voltage means 1, True, asserted
  - Low true signals means that low voltage means 1, True, asserted
  - In real world, have both high and low true signals.

---

High True vs. Low True Logic

- Different ways to say that a signal is **high true**
  - Is high if signal is TRUE, is low if signal is FALSE
  - Is high if signal is 1, is low if signal is 0
  - Is high if signal is asserted, is low if signal is negated

- Different ways to say that a signal is **low true**
  - Is low if signal is TRUE, is high if signal is FALSE
  - Is low if signal is 1, is high if signal is 0
  - Is low if signal is asserted, is high if signal is negated

---

Asserted vs. Negated

- **Asserted** ALWAYS means that a signal is TRUE or logic 1.
  - Logic 1 could be represented by a HIGH voltage (high true)
  - Logic 0 could be represented by LOW voltage (low true)

- **Negated** ALWAYS means that a signal is FALSE or logic 0.
  - Logic 0 could be represented by a LOW voltage (high true)
  - Logic 0 could be represented by a HIGH voltage (low true)
### The Physical World

When a button is pressed, it is asserted (true). However, its physical construction may output this as a LOW VOLTAGE (low true!!)

To the person pressing the button, they don’t know and don’t care that a low voltage is output when the button is pressed. They just know they have ASSERTED the button.

---

### THE Problem

Have two buttons, each button outputs a low voltage (L) when pressed.

- **Button A**
- **Button B**

Want a Voltage Gate that outputs a ‘H’ when both buttons are ASSERTED.

The rest of the lecture will be devoted to determining the answer…

---

### Examples of high, low signals

- **High True button (switch)**
  - Switch open (negated), output is L
  - Switch closed (asserted), output is H
Examples of high, low signals

Low True switch

Switch open (negated), output is H

Switch closed (asserted), output is L

7400 Logic Gate

AND gate with high true inputs, low true output

OR gate with low true inputs, high true output

Fixed Logic Polarity vs Mixed Logic Polarity

- In Fixed logic polarity, every signal is considered high true.
- In Mixed logic polarity, can have high, low true signals.
  - Low true signal names followed by ‘(L)’ to indicate low true
Fixed Polarity vs Mixed Polarity

- **NAND, AND**
  - Fixed: \((AB)'\) is read as “A nand B”
  - Mixed: \((AB) (L)\) is read “A and B, low true”.

- **NOR, OR**
  - Fixed: \((A+B)'\) is read as “A nor B”
  - Mixed: \((A+B) (L)\) is read “A or B, low true”.

- **NOT**
  - Fixed: \((A)'\) is read as “NOT A”
  - Mixed: \((A) (L)\) is read as “A, low true”

---

### 7404 Logic Gate

<table>
<thead>
<tr>
<th>A</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>H</td>
<td>L</td>
</tr>
</tbody>
</table>

Buffer that converts high true input to low true output

\[ (A) (L) \]

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

Low true

High true

---

### 7402 Logic Gate

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>L</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>H</td>
<td>H</td>
<td>L</td>
</tr>
</tbody>
</table>

OR gate with high true inputs, low true output

\[ (A+B) (L) \]

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

Low true

High true

---

AND gate with low true inputs, high true output

\[ AB \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
7408 Logic Gate

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>H</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
</tbody>
</table>

AND gate with high true inputs, high true output

\[ (A + B) \text{ (L)} \]

OR gate with low true inputs, low true output

\[ \text{B(L)} \]

OR gate with high true inputs, high true output

\[ \text{Y} = (A + B \text{ (L)}) \]

High True

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

Low True

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

7432 Logic Gate

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>H</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
</tbody>
</table>

OR gate with high true inputs, high true output

\[ \text{A+(B) (L)} \]

AND gate with low true inputs, low true output

\[ \text{B(L)} \]

OR gate with high true inputs, high true output

\[ \text{Y} = (A B \text{ (L)}) \]

High True

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

Low True

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

Problem #1

Two low true switches. When either switch A or switch B is asserted (pressed), want Y to be asserted. Y is high true.
Problem #1 (solution)

Two low true switches. When either switch A or switch B is asserted (pressed), want Y to be asserted. Y is high true.

Problem #2

Two high true switches. When either switch A or switch B is asserted (pressed), want Y to be asserted. Y is low true.

Problem #2 (solution)

Two high true switches. When either switch A or switch B is asserted (pressed), want Y to be asserted. Y is low true.
Problem #3

Two low true switches. When both switch A and switch B is asserted (pressed), want Y to be asserted. Y is low true.

Problem #3 (solution)

Two low true switches. When both switch A and switch B is asserted (pressed), want Y to be asserted. Y is low true.

THE Problem (again)

Have two buttons, each button outputs a low voltage (L) when pressed.

Want a Voltage Gate that outputs a ‘H’ when both buttons are ASSERTED.

Solution?????
**THE Problem (solution)**

Have two buttons, each button outputs a low voltage (L) when pressed.

Button A

Button B

Want a Voltage Gate that outputs a ‘H’ when both buttons are ASSERTED.

When both A and B asserted (low true), output is asserted (high true)

---

**Mixed High True, Low True Inputs**

A(L)  

Y = AB (L)  

B

One low true switch and one high true switch. When both switch A and switch B is asserted (pressed), want Y to be asserted. Y is low true.

---

**Mixed High True, Low True Inputs**

A(L)  

Y = AB (L)  

B

Hmmm… What is this? This does not match any of our gate types. We will have to convert one the gate inputs so that either we have BOTH high true inputs or BOTH low true inputs.
Mixed High True, Low True Inputs

Solution #1: Convert both gate inputs to low true.

\[ Y = AB \quad (L) \]

Now we have a two input gate with both inputs low true. We can now match this to one of our two input gates.

---

Mixed High True, Low True Inputs

Solution #2: Convert both gate inputs to high true.

\[ Y = AB \quad (L) \]

Now we have a two input gate with both inputs high true. We can now match this to one of our two input gates.

---

Gate Summaries
Complete Logic Families

• A set of logic gates is complete if it can implement any boolean function.
  – Must be able to implement AND, OR, NOT function to be complete

The 7400 gate is complete all by itself!!!!

Other Complete Logic Families

The 7402 gate is also complete all by itself.

Any boolean equation can be implemented using either just 7400 gates or just 7402 gates.

Other Complete Logic Families (cont)

The 7408 and 7404 together make a complete family.

The 7432 and 7404 together make a complete family.
**Sum of Products**

- A boolean equation in the form:
  
  \[ f = \text{and\_term} + \text{and\_term} \ldots + \text{and\_term} \]
  
  is called a *Sum of Products* (SOP).

  \[ Y = AB + CD \]

  Implementing this logic in two levels of gating is easy.

**Product of Sums**

- A boolean equation in the form:
  
  \[ f = (\text{or\_term}) (\text{or\_term}) \ldots (\text{or\_term}) \]
  
  is called a *Product of Sums* (POS).

  \[ Y = (A+B) (C+D) \]

  Implementing this logic in two levels of gating is easy.

**What do you have to know?**

- Definitions of Assertion, Negation, High-True, Low-true
- Low, High true switch construction
- Low, High True boolean functions of Voltage gates
- Problems in the form of the switch problems given in these notes
- Complete Logic Families
- NAND-NAND form drawn in mixed logic. NOR-NOR form drawn in mixed logic.