• 3-way light switch:

- want either switch to be able to turn light ON/OFF independent of position of other switch

\[
X = 1, \text{ light ON}, \quad A = 1, \text{ switch UP}, \quad B = 1, \text{ UP} \\
0, \text{ OFF} \quad 0, \text{ switch DOWN} \quad 0, \text{ DOWN}
\]

- assume when first installed \( X = 0, A = 0, \) and \( B = 0 \)

- want:

\[
\begin{align*}
A = 0 \text{ and } B = 0 & \quad \Rightarrow \\
A = 0 \rightarrow 1 \text{ and } B = 0 & \quad \Rightarrow \\
A = 1 \text{ and } B = 0 \rightarrow 1 & \quad \Rightarrow \\
A = 1 \rightarrow 0 \text{ and } B = 1 & \quad \Rightarrow \\
A = 0 \text{ and } B = 1 \rightarrow 0 & \quad \Rightarrow \\
\end{align*}
\]

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

\( \Rightarrow \text{ circuit (6)} \)

- used double pole switches to realize (6) : \( X = A \cdot B + A \cdot \overline{B} \)

- (6) termed exclusive-OR (XOR) since \( A \) or \( B = 1 \) but not both

- (7) termed inclusive-OR (or just OR) since \( A \) or \( B \) or both = 1

• Seat-belt buzzer:

- want:

\[
\begin{align*}
I = \begin{cases} 
1, \text{ ignition ON} , \\
0, \text{ OFF} 
\end{cases} \quad S = \begin{cases} 
1, \text{ fastened} , \\
0, \text{ unfastened} 
\end{cases} \quad B = \begin{cases} 
1, \text{ sound} , \\
0, \text{ quiet} 
\end{cases}
\end{align*}
\]
\[ \begin{array}{c|c|c}
I & S & B \\
0 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
\end{array} \Rightarrow \text{circuit } 2 \Rightarrow B = I \cdot \overline{S} \Rightarrow \text{seat-belt needs to be n.c. switch} \]

- physically, S best as n.o., but want it to be n.c. in control circuit
  \[ \Rightarrow \text{can use relay:} \]

- both B and relay coil are loads
- when S actuated, current flows through relay coil, which becomes a magnet that actuates relay contact

- **Relays:**
  - up to now, all inputs have been mechanical switches that directly made or broke the circuit controlling the load
  - relays can be used to indirectly control higher power circuit:

\[ \text{120 V AC} \]

\[ \text{12 V DC} \]

\[ \text{high-power circuit} \]

\[ \text{low-power control circuit} \]
- relays can be normally open (n.o.) or normally closed (n.c.)
- use of relays eliminates need for complex multiple pole switches and wiring

- **Logic Gates:** (see handout)
  - in seat-belt control circuit, relay used to convert n.o. mechanical switch to n.c. switch
    ⇒ performing Logical NOT ⇒ inverting signal ⇒ inverter
  - **NOT gate:**
    ![Not Gate Diagram](image)
  
  - **NOR gate:** when two (or more) inputs connected in parallel to single relay of NOT gate, get NOR gate
    ![NOR Gate Diagram](image)
    
    - if n.o. relay contact used in NOR gate ⇒ **OR gate** = A
    ![OR Gate Diagram](image)
    
    - but, unlike OR gate, NOR gate (and NAND gate) are universal gates
      ⇒ any Boolean logic circuit can be realized using just NOR gates (or just NAND gates)
- NOT gate via NOR gate:

\[ A \quad B \quad X \Rightarrow \begin{array}{ccc} 0 & 0 & 1 \\ 1 & 0 & 0 \end{array} \]

\[ \Rightarrow A \quad \overline{X} = \overline{A} \]

- OR gate via NOR gates:

\[ A \quad A + B \quad X = A + B \quad \iff \quad B \quad 0 \]

- AND gate via NOR gates: not immediately obvious how to construct

\[ \Rightarrow \text{use Theorems of Boolean Algebra} \]

- Theorems of Boolean Algebra (see handout)
  - used to manipulate Boolean expressions
  - developed by George Boole in 1850s (Laws of Thought)
  - in 1938, Claude Shannon saw one-to-one correspondence between Boolean expressions and switching circuits
    \[ \Rightarrow \text{Boolean algebra can be used to simplify logic control circuits} \]
  - AND gate via NOR gates + Boolean algebra:
    Want: \[ X = A \cdot B \]
    \[ = \overline{A \cdot B} \quad \text{(by Negation Th.)} \]

    \[ \text{Let } A' = \overline{A} \text{ and } B' = \overline{B} \]
    \[ X = \overline{A' \cdot B'} \]
    \[ = \overline{A' + B'} \quad \text{(by DeMorgan 2 — left side of DeMorgan 2 is NOR)} \]
    \[ = A + B \]
Theorems also give NOT and OR gates via NOR gates:

NOT: \( X = \overline{A} \)

\[
\begin{align*}
X &= \overline{A} \cdot 1 \quad \text{(by Char. 2)} \\
&= \overline{A} \cdot 0 \quad \text{(1=0 & 0=1)} \\
&= \overline{A} + 0 \quad \text{(by DeMorgan 2)}
\end{align*}
\]

OR: \( X = A + B \)

\[
\begin{align*}
X &= \overline{A} + \overline{B} \quad \text{(by Neg.)} \\
&= \overline{A} \cdot \overline{B} \quad \text{(by DeMorgan 1)} \\
&= \overline{A} \cdot \overline{B} \cdot 1 \quad \text{(by Char. 2)} \\
&= \overline{A} \cdot \overline{B} \cdot 0 \quad \text{(1=0)} \\
&= \overline{A} \cdot \overline{B} + 0 \quad \text{(by DeMorgan 2)} \\
&= (A + B) + 0 \quad \text{(by DeMorgan 2)}
\end{align*}
\]

- **Transistor Logic:**
  - Why are 3 NOR gates used instead of 1 AND gate?
  - NOR gate easy to make using transistor
  - transistors used instead of relays in all control applications except to switch high power circuits

- for control system design, can think in terms of relays
• **Multi-input gates:** logic gates can have more than two inputs

\[ X = A + B + \ldots \]  
\[ \text{can all be realized using multi-input NOR or NAND gates} \]

• **Input negation:**

\[ A \quad \Rightarrow \quad A \quad \Leftrightarrow \quad X = A + B \]

• **Nesting:** when output from gate used as input to next, equivalent to parenthesis in Boolean expression (AND higher precedence than OR)

• **Logic gate network →** Boolean expression:

\[ X = (A \cdot B + C) + C \]

\[ X = [(A \cdot \bar{B}) + B + (B + C)] \cdot \bar{D} \]

\[ = (A \cdot \bar{B} + B + B + C) \cdot \bar{D} \]
### Theorems and Laws of Boolean Algebra

**CHARACTERISTIC THEOREMS**

1. \( X \cdot 0 = 0 \)
2. \( X \cdot 1 = X \)
3. \( X + 0 = X \)
4. \( X + 1 = 1 \)

**NEGATION THEOREM**

\( \overline{X} = X \)

**NEGATION THEOREM**

\( \overline{X} = X \)

**INCLUSION THEOREMS**

1. \( X \cdot \overline{X} = 0 \)
2. \( X + \overline{X} = 1 \)

**COMmutative LAW**

1. \( X + Y = Y + X \)
2. \( X \cdot Y = Y \cdot X \)

**ABSORPTIVE LAWS**

1. \( X + XY = X \)
2. \( X(X + Y) = X \)

**ASSOCIATIVE LAW**

1. \( X + Y + Z = X + (Y + Z) \)
2. \( X \cdot Y \cdot Z = X \cdot (Y \cdot Z) \)

**REFLECTIVE THEOREMS**

1. \( X + \overline{XY} = X + Y \)
2. \( X(\overline{X} + Y) = XY \)
3. \( XY + \overline{XYZ} = XY + YZ \)

**DISTRIBUTIVE LAW**

1. \( X \cdot Y + X \cdot Z = X(Y + Z) \)
2. \( (X + Y)(W + Z) = XW + XZ + YW + YZ \)

**CONSISTENCY THEOREM**

1. \( XY + X\overline{Y} = X \)
2. \( (X + Y)(X + \overline{Y}) = X \)

**IDEMPOTENT THEOREMS**

1. \( X \cdot X = X \)
2. \( X + X = X \)

**DEMORGAN’S LAWS**

1. \( X \overline{Y} = X + \overline{Y} \)
2. \( \overline{X + Y} = \overline{X} \cdot \overline{Y} \)
Logic: $\land$ A  B  $\lor$  $\downarrow$  $\leftrightarrow$  $\bar{B}$  $\leftarrow$  $\bar{A}$  $\rightarrow$  $\uparrow$

Boolean: $\bullet$  $\oplus$  $+$  $\odot$

<table>
<thead>
<tr>
<th>Name</th>
<th>AND</th>
<th>XOR</th>
<th>OR</th>
<th>NOR</th>
<th>XNOR</th>
<th>NOT</th>
<th>NOT</th>
<th>NAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<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>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

short circuit

$A \cdot B$, $\bar{A} \cdot B$, $A$, $\bar{A}$, $\bar{A} \cdot B$, $A + B$, $\bar{A} \cdot B + A \cdot B$, $A + B$, $\bar{A}$, $\bar{A} + B$, $\bar{A} + B$, $A \cdot B$, no control
# Digital Logic Gates and Associated Logical Operations for Binary Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Symbol</th>
<th>Logical Operation</th>
<th>Truth Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td><img src="image1" alt="AND gate" /></td>
<td>( Z = A \cdot B )</td>
<td>( \begin{array}{ccc} A &amp; B &amp; Z \ 0 &amp; 0 &amp; 0 \ 0 &amp; 1 &amp; 0 \ 1 &amp; 0 &amp; 0 \ 1 &amp; 1 &amp; 1 \end{array} )</td>
</tr>
<tr>
<td>OR</td>
<td><img src="image2" alt="OR gate" /></td>
<td>( Z = A + B )</td>
<td>( \begin{array}{ccc} A &amp; B &amp; Z \ 0 &amp; 0 &amp; 0 \ 0 &amp; 1 &amp; 1 \ 1 &amp; 0 &amp; 1 \ 1 &amp; 1 &amp; 1 \end{array} )</td>
</tr>
<tr>
<td>NOT</td>
<td><img src="image3" alt="NOT gate" /></td>
<td>( Z = \overline{A} )</td>
<td>( \begin{array}{c} A \mid Z \ 0 &amp; 1 \ 1 &amp; 0 \end{array} )</td>
</tr>
<tr>
<td>NAND</td>
<td><img src="image4" alt="NAND gate" /></td>
<td>( Z = A \cdot \overline{B} )</td>
<td>( \begin{array}{ccc} A &amp; B &amp; Z \ 0 &amp; 0 &amp; 1 \ 0 &amp; 1 &amp; 1 \ 1 &amp; 0 &amp; 1 \ 1 &amp; 1 &amp; 0 \end{array} )</td>
</tr>
<tr>
<td>NOR</td>
<td><img src="image5" alt="NOR gate" /></td>
<td>( Z = \overline{(A + B)} )</td>
<td>( \begin{array}{ccc} A &amp; B &amp; Z \ 0 &amp; 0 &amp; 1 \ 0 &amp; 1 &amp; 0 \ 1 &amp; 0 &amp; 0 \ 1 &amp; 1 &amp; 0 \end{array} )</td>
</tr>
<tr>
<td>XOR</td>
<td><img src="image6" alt="XOR gate" /></td>
<td>( Z = A \oplus B )</td>
<td>( \begin{array}{ccc} A &amp; B &amp; Z \ 0 &amp; 0 &amp; 0 \ 0 &amp; 1 &amp; 1 \ 1 &amp; 0 &amp; 1 \ 1 &amp; 1 &amp; 0 \end{array} )</td>
</tr>
<tr>
<td>XNOR</td>
<td><img src="image7" alt="XNOR gate" /></td>
<td>( Z = A \oplus B )</td>
<td>( \begin{array}{ccc} A &amp; B &amp; Z \ 0 &amp; 0 &amp; 1 \ 0 &amp; 1 &amp; 0 \ 1 &amp; 0 &amp; 0 \ 1 &amp; 1 &amp; 1 \end{array} )</td>
</tr>
</tbody>
</table>
- **Simplifying Boolean expression** \( \Leftrightarrow \) **reducing number of logic gates**

\[
X = (A \cdot B + C) + C
\]

\[
= (A \cdot B \cdot C) + C \quad \text{(by DeMorgan 2)}
\]

\[
= A \cdot B + C \quad \text{(by Reflective 1)}
\]

\[\downarrow \text{from 3 to 2 gates}\]

\[
X = (A \cdot B + B + B + C) \cdot D
\]

\[
= (A \cdot B + B + C) \cdot D \quad \text{(by Idempotent 2)}
\]

\[
= (A + B + C) \cdot D \quad \text{(by Reflective 1)}
\]

\[\downarrow \text{from 4 to 2 }\]

- **Canonical Sum-of-Products Form**
  - Used to realize any logic control circuit from its truth table

1. For each circuit, construct its truth table to relate all possible inputs to desired output of circuit:

<table>
<thead>
<tr>
<th>Row</th>
<th>A</th>
<th>B</th>
<th>X</th>
<th>desired output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

2. For each row where \(X = 1\), AND together inputs, where 0 inputs are NOTed

- row 0: \(X \neq 1\)
- row 1:
  - \(A \cdot \bar{B} \cdot B\)
- row 2:
  - \(A \cdot \bar{B} \cdot \bar{B} \cdot B\)
- row 3:
  - \(A \cdot \bar{B} \cdot B\)
3. OR together AND gates from step 2
   \[ X = \overline{A} \cdot B + A \cdot \overline{B} + A \cdot B \]

4. Use Theorems of Boolean Algebra to simplify if possible
   \[ = \overline{A} \cdot B + A \cdot (\overline{B} + B) \quad \text{(by Distrib. 1)} \]
   \[ = \overline{A} \cdot B + A \cdot (1) \quad \text{(by Inclusion 2)} \]
   \[ = A \cdot B + A \quad \text{(by Char. 2)} \]
   \[ = B + A \quad \text{(by Reflect. 1)} \]
   \[ = A + B \Rightarrow \text{OR} \quad \text{(by Commut. 1)} \]

- termed “Sum-of-Products” since OR-ing together (summing) AND-ed inputs (products)

- Example: XOR

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

   \[ \Rightarrow \quad X = \overline{A} \cdot B + A \cdot \overline{B}, \quad \text{can’t simplify} \]

- Example: Seat belt and Door-Open Buzzer

   \[ I = \begin{cases} 
   1, \text{ignition ON} \\
   0, \text{OFF} 
   \end{cases}, \quad S = \begin{cases} 
   1, \text{seat belt fastened} \\
   0, \text{unfastened} 
   \end{cases} \]

   \[ D = \begin{cases} 
   1, \text{door closed} \\
   0, \text{door open} 
   \end{cases}, \quad B = \begin{cases} 
   1, \text{sound} \\
   0, \text{quiet} 
   \end{cases} \]
\[
B = I \cdot S \cdot D + I \cdot S \cdot D + I \cdot S \cdot D
\]

\[
= I \cdot (S \cdot D + S \cdot D + S \cdot D) \quad \text{(by Distrib. 1)}
\]

\[
= I \cdot (S + S \cdot D) \quad \text{(by Consist. 1)}
\]

\[
= I \cdot (S + D) \quad \text{(by Reflect. 1)}
\]

<table>
<thead>
<tr>
<th>I</th>
<th>S</th>
<th>D</th>
<th>B</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>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>0</td>
</tr>
</tbody>
</table>

- Example: Automatic Door and Lock

\[ D = \begin{cases} 1, \text{door opening} \\ 0, \text{door closing} \end{cases} \quad \text{O = } \begin{cases} 1, \text{door not shut} \\ 0, \text{door shut} \end{cases} \]

\[ M = 1, \text{someone on mat} \quad \text{L = } \begin{cases} 1, \text{unlocked} \\ 0, \text{locked} \end{cases} \]

\[ \text{Operation: Want door to open if someone on mat and door unlocked.} \]

\[ \text{If locked, want it to stay open or shut independent of mat} \]