Basic Definitions

A *gate* is a device that accepts one or more input electrical signals, and produces a single output electrical signal.

A *circuit* is a combination of gates.

*Boolean algebra* is an algebra in which variables and functions take on one of two values, 0 ("false") or 1 ("true").

A *logic diagram* is a graphical representation of a circuit, where each type of gate is represented by a specific graphical symbol.

A *truth table* is a table showing all possible input values and the associated output values.
NOT Gate

\[ x \rightarrow \text{NOT } x \]

<table>
<thead>
<tr>
<th>X</th>
<th>NOT X</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\( ! \) is the corresponding Java \textit{unary operator} inverting the value of a boolean variable.

\begin{verbatim}
boolean x=false, y; // boolean is a primitive \textit{data type}.
y=!x; // Assignment statement:
    // The value of y becomes \textit{true}.
\end{verbatim}

\textit{Boolean expression: } \( Y = X' \)
AND Gate

&& is the corresponding Java binary operator. In Java, it is called Conditional-AND, because the second operand is evaluated only if needed there.

```java
int a=3, b=3; // int is another primitive data type
if ((a==b)&&(b==3)) // == means “equal to”, also binary
    System.out.println("a and b are equal to 3.");
```

Boolean expression:

\[ A = X \cdot Y \]
OR Gate

Boolean expression: \( A = X + Y \)

\( \lor \) is the corresponding Java binary operator. In Java, it is called Conditional-OR, because the second operand is evaluated only if needed.

```java
boolean x=false, y=true;
if (x || y)
    System.out.println(“At least one is enough!”);
```
XOR Gate

Exclusive OR. Java has no such operator for variables of type `boolean`, but it offers bitwise exclusive OR, `^`.

```java
int x = 0xAA; // Hexadecimal values are preceded by "0x" in Java!
int y = 0xD4; // preceeded by "0x" in Java!
System.out.println(x ^ y); // Prints 126 (decimal)
```
More Bitwise Operators in Java

~ inverts a bit pattern; i.e., it’s a bitwise NOT;
& denotes a bitwise AND operation;
| is the bitwise OR operator.

The int data type is a 32-bit signed two's complement integer. It has a minimum value of \(-2,147,483,648\) and a maximum value of \(2,147,483,647\) (inclusive).

```java
int x = 0xAA; // 10101010 in Base 2
int y = 0xD4; // 11010100 in Base 2
System.out.println(~x); // Prints -171 (decimal)
System.out.println(x&y); // Prints 128 (decimal)
System.out.println(x|y); // Prints 254 (decimal)
```
NAND Gate

Boolean Expression: $X = (A \cdot B)'$

Logic Diagram Symbol:

<table>
<thead>
<tr>
<th>Truth Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>
NOR Gate

Boolean Expression: \( X = (A + B)' \)

Logic Diagram Symbol:

Truth Table:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>X</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>
Three-Input AND Gate

Boolean Expression: \( X = A \cdot B \cdot C \)

Logic Diagram Symbol:

Truth Table:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>X</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>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>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Constructing Gates

Gates can be constructed using transistors (Section 4.3). You’re not responsible for this section.