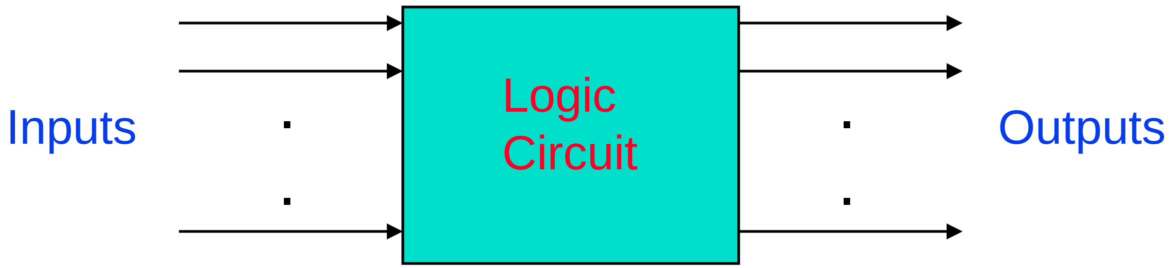
Boolean Algebra

Overview

- **Logic functions with 1's and 0's**
  - **Building digital circuitry**
- **Truth tables**
- **Logic symbols and waveforms**
- **Boolean algebra**
- **Properties of Boolean Algebra**
  - **Reducing functions**
  - **Transforming functions**

Digital Systems

- **Analysis problem:**

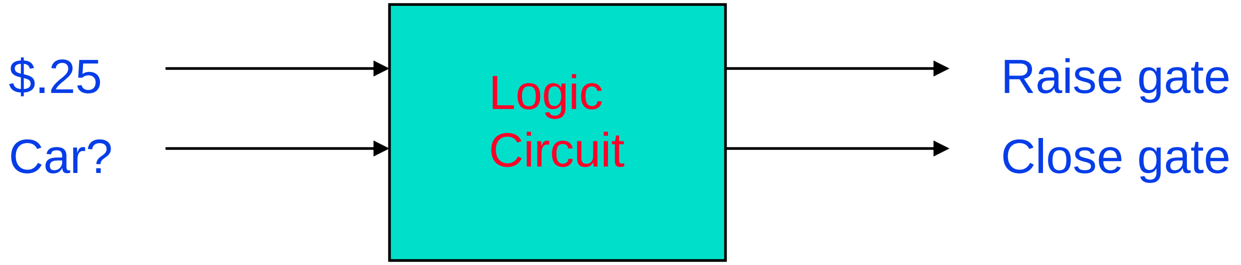


  - **Determine binary outputs for each combination of inputs**

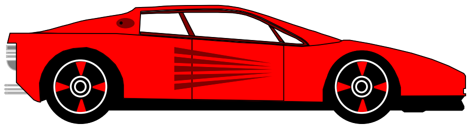
- **Design problem: given a task, develop a circuit that accomplishes the task**
  - **Many possible implementation**
  - **Try to develop “best” circuit based on some criterion (size, power, performance, etc.)**

Toll Booth Controller

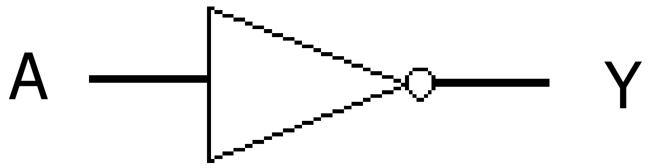
- Consider the design of a toll booth controller
- Inputs: quarter, car sensor
- Outputs: gate lift signal, gate close signal



- If driver pitches in quarter, raise gate.
- When car has cleared gate, close gate.



Describing Circuit Functionality: Inverter



Symbol

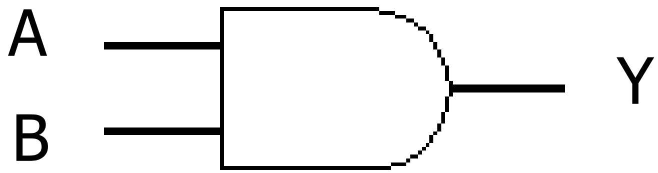
Truth Table

A	Y
0	**1**
1	**0**

Input Output

- **Basic logic functions have symbols.**
- **The same functionality can be represented with truth tables.**
  - **Truth table completely specifies outputs for all input combinations.**
- **The above circuit is an inverter.**
  - **An input of 0 is inverted to a 1.**
  - **An input of 1 is inverted to a 0.**

The AND Gate

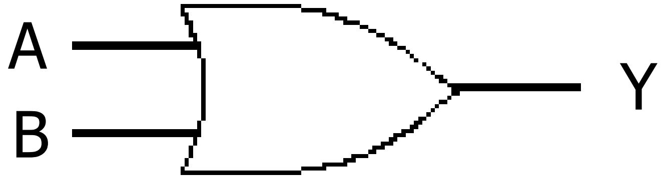


- This is an AND gate.
- So, if the two inputs signals are asserted (high) the output will also be asserted. Otherwise, the output will be deasserted (low).

Truth Table

A	B	Y
0	**0**	**0**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**

The OR Gate



- This is an OR gate.
- So, if either of the two input signals are asserted, or both of them are, the output will be asserted.

A	B	Y
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**1**

Describing Circuit Functionality: Waveforms

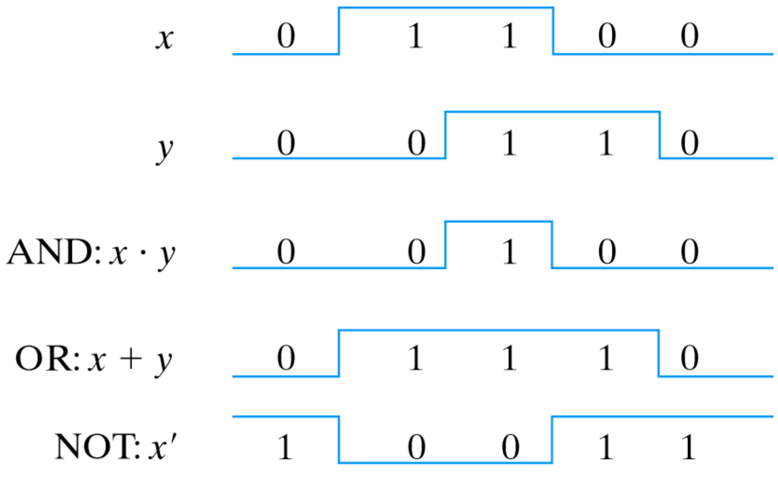


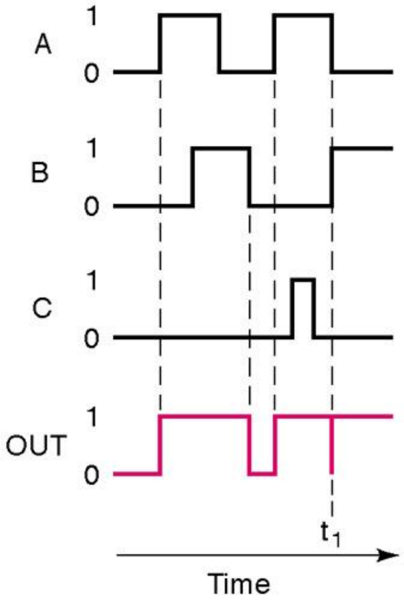
Fig. 1-5 Input-output signals for gates

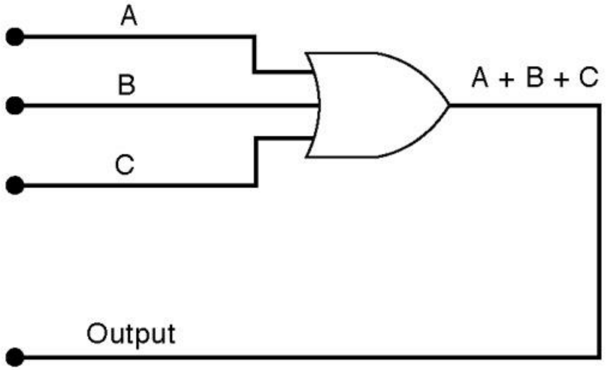
AND Gate

A	B	Y
0	**0**	**0**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**

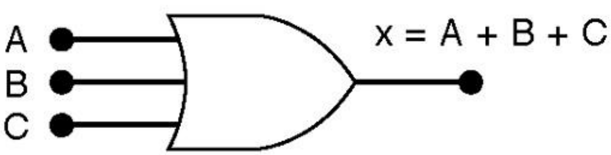
- **Waveforms provide another approach for representing functionality.**
- **Values are either high (logic 1) or low (logic 0).**
- **Can you create a truth table from the waveforms?**

Consider three-input gates





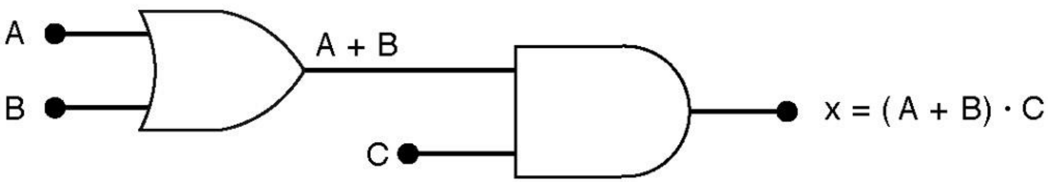
3 Input OR Gate



A	B	C	x = A + B + C
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Ordering Boolean Functions

- How to interpret A•B+C?
  - Is it A•B ORed with C ?
  - Is it A ANDed with B+C ?
- Order of precedence for Boolean algebra: AND before OR.
- Note that parentheses are needed here :



Boolean Algebra

- A *Boolean algebra* is defined as a closed algebraic system containing a set K or two or more elements and the two operators, . and +.
- Useful for identifying and *minimizing* circuit functionality
- Identity elements
  - a + 0 = a
  - a . 1 = a
- 0 is the identity element for the + operation.
- 1 is the identity element for the . operation.

Commutativity and Associativity of the Operators

- **The** Commutative Property:

  For every a and b in K,

  - **a + b = b + a**
  - **a . b = b . a**

- **The** Associative Property:

  For every a, b, and c in K,

  - **a + (b + c) = (a + b) + c**
  - **a . (b . c) = (a . b) . c**

Distributivity of the Operators and Complements

- **The** Distributive Property:

  For every a, b, and c in K,

  - **a + (b . c) = (a + b) . (a + c)**
  - **a . (b + c) = (a . b) + (a . c)**

- **The Existence of the** Complement**:**

  For every a in K there exists a unique element called a' (*complement of a*) such that,

  - **a + a' = 1**
  - **a . a' = 0**

- **To simplify notation, the . operator is frequently omitted. When two elements are written next to each other, the AND (.) operator is implied…**
  - **a + b . c = (a + b) . (a + c)**
  - **a + bc = (a + b)(a + c)**

Duality

- The principle of *duality* is an important concept. This says that if an expression is valid in Boolean algebra, the dual of that expression is also valid.
- To form the dual of an expression, replace all + operators with . operators, all . operators with + operators, all ones with zeros, and all zeros with ones.
- Form the dual of the expression

  $a + (bc) = (a + b)(a + c)$

- Following the replacement rules…

  $a(b + c) = ab + ac$

- Take care not to alter the location of the parentheses if they are present.

Involution

- **This theorem states:**

  $a'' = a$

- **Remember that $aa' = 0$ and $a+a'=1$.**
  - **Therefore, a' is the complement of a and a is also the complement of a'.**
  - **As the complement of a' is unique, it follows that $a''=a$.**
- **Taking the double inverse of a value will give the initial value.**

Absorption

- **This theorem states:**

  $a + ab = a$ $\qquad\qquad$ $a(a+b) = a$

- **To prove the first half of this theorem:**

$$
\begin{aligned}
\mathbf{a + ab} &\mathbf{= a \cdot 1 + ab} \\
&\mathbf{= a\,(1 + b)} \\
&\mathbf{= a\,(b + 1)} \\
&\mathbf{= a\,(1)} \\
\mathbf{a + ab} &\mathbf{= a}
\end{aligned}
$$

DeMorgan's Theorem

- **A key theorem in simplifying Boolean algebra expression is DeMorgan's Theorem. It states:**

  $(a + b)' = a'b'$ $\qquad$ $(ab)' = a' + b'$

- **Complement the expression**

  $a(b + z(x + a'))$ and simplify.

$$
\begin{aligned}
(a(b+z(x + a')))' &= a' + (b + z(x + a'))' \\
&= a' + b'(z(x + a'))' \\
&= a' + b'(z' + (x + a')') \\
&= a' + b'(z' + x'a'') \\
&= a' + b'(z' + x'a)
\end{aligned}
$$

Summary

- Basic logic functions can be made from AND, OR, and NOT (invert) functions
- The behavior of digital circuits can be represented with waveforms, truth tables, or symbols
- Primitive gates can be combined to form larger circuits
- Boolean algebra defines how binary variables can be combined
- Rules for associativity, commutativity, and distribution are similar to algebra
- DeMorgan's rules are important.
  - Will allow us to reduce circuit sizes.