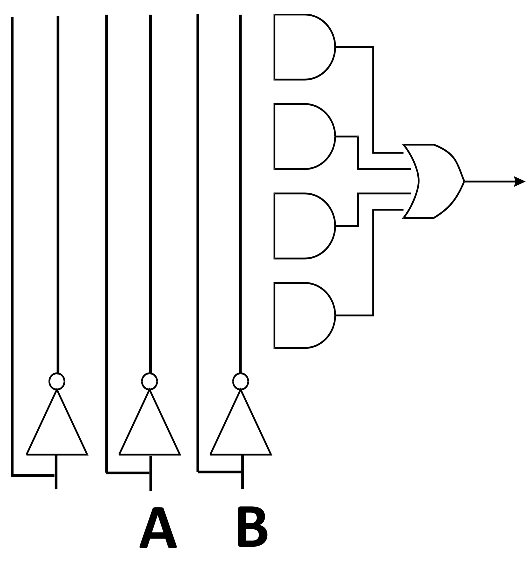
More Boolean Algebra



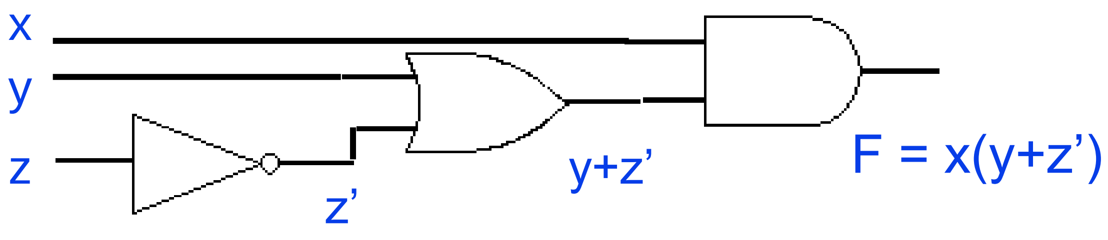
Overview

- Expressing Boolean functions
- Relationships between algebraic equations, symbols, and truth tables
- Simplification of Boolean expressions
- Minterms and Maxterms
- AND-OR representations
    - Product of sums
    - Sum of products

Boolean Functions

- Boolean algebra deals with binary variables and logic operations.
- Function results in binary 0 or 1

x	y	z	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

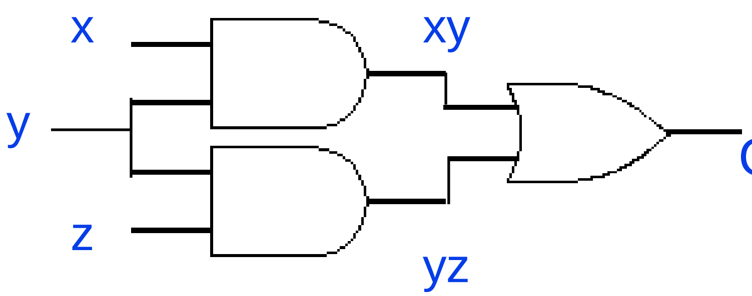


F = x(y+z')

Boolean Functions

- **Boolean algebra deals with binary variables and logic operations.**
- **Function results in binary 0 or 1**

x	y	z	xy	yz	G
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	1	1
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	1	0	1
1	1	1	1	1	1

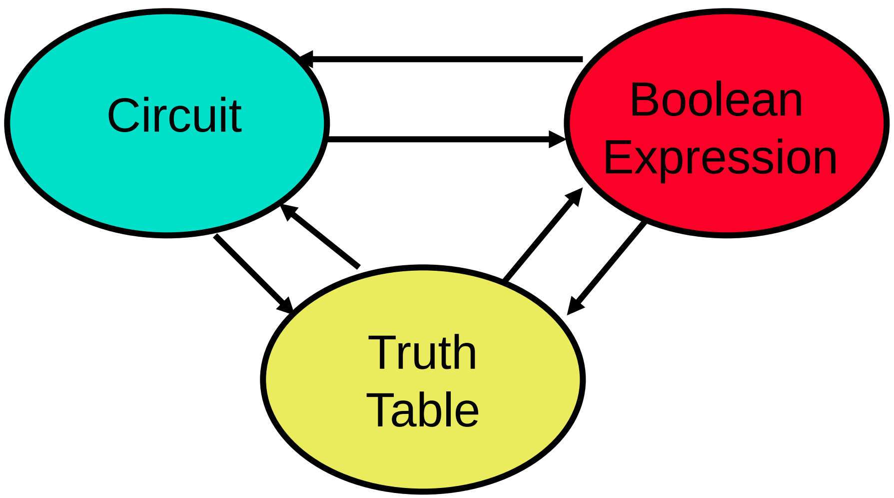


$G = xy + yz$

We will learn how to transition between equation, symbols, and truth table.

Representation Conversion

- Need to transition between boolean expression, truth table, and circuit (symbols).
- Converting between truth table and expression is easy.
- Converting between expression and circuit is easy.
- More difficult to convert to truth table.



Truth Table to Expression

- Converting a truth table to an expression
  - Each row with output of 1 becomes a product term
  - Sum product terms together.

x	y	z	G
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

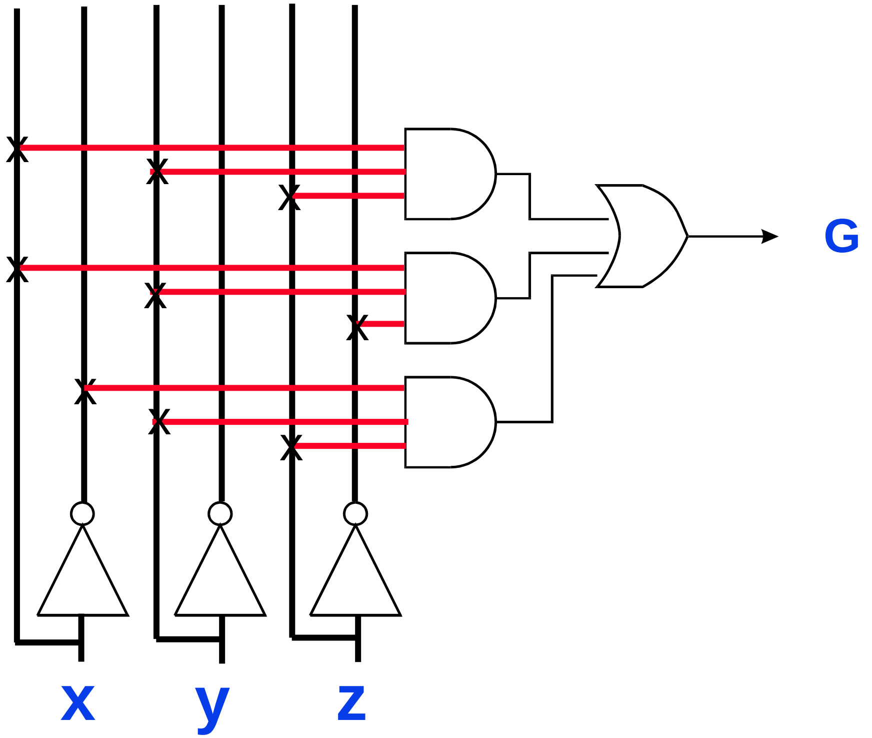
Any Boolean Expression can be represented in sum of products form!

$xyz + xyz' + x'yz$

Equivalent Representations of Circuits

- All three formats are equivalent
- Number of 1's in truth table output column equals AND terms for Sum-of-Products (SOP)

x	y	z	G
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



$G = xyz + xyz' + x'yz$

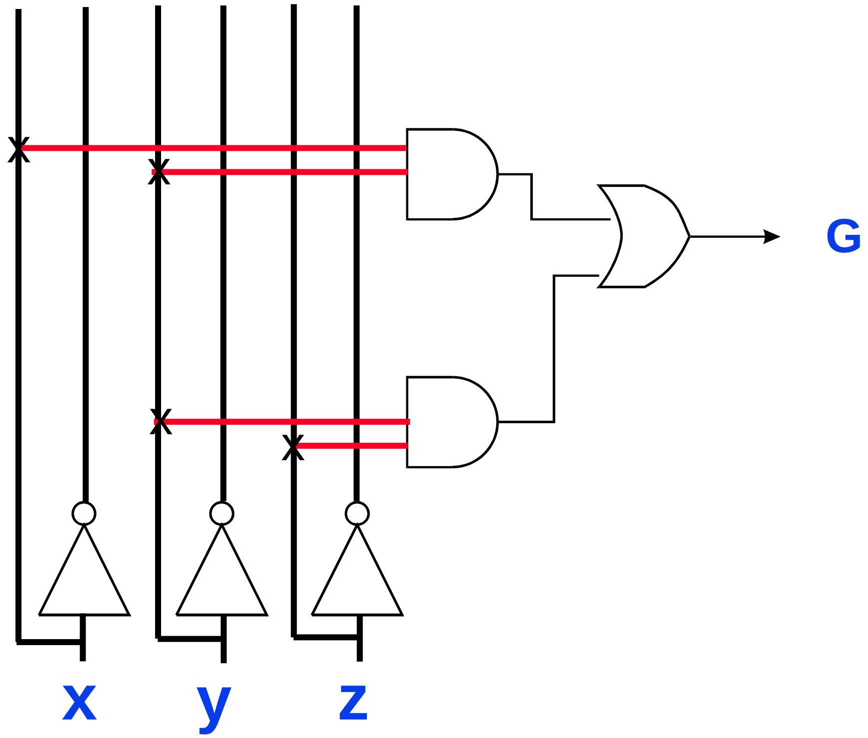
Reducing Boolean Expressions

- **Is this the smallest possible implementation of this expression? No!** $G = xyz + xyz' + x'yz$
- **Use Boolean Algebra rules to reduce complexity while preserving functionality.**
- **Step 1: Use Theorum 1 ($a + a = a$)**
  - **So** $xyz + xyz' + x'yz = xyz + xyz + xyz' + x'yz$
- **Step 2: Use distributive rule $a(b + c) = ab + ac$**
  - **So** $xyz + xyz + xyz' + x'yz = xy(z + z') + yz(x + x')$
- **Step 3: Use Postulate 3 ($a + a' = 1$)**
  - **So** $xy(z + z') + yz(x + x') = xy.1 + yz.1$
- **Step 4: Use Postulate 2 ($a . 1 = a$)**
  - **So** $xy.1 + yz.1 = xy + yz = xyz + xyz' + x'yz$

Reduced Hardware Implementation

- Reduced equation requires less hardware!
- Same function implemented!

x	y	z	G
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



$G = xyz + xyz' + x'yz = xy + yz$

Minterms and Maxterms

- **Each variable in a Boolean expression is a literal**
- **Boolean variables can appear in normal (x) or complement form (x')**
- **Each AND combination of terms is a <u>minterm</u>**
- **Each OR combination of terms is a <u>maxterm</u>**

For example:
Minterms

x	y	z	Minterm	
0	0	0	x'y'z'	m_0
0	0	1	x'y'z	m_1
…				
1	0	0	xy'z'	m_4
…				
1	1	1	xyz	m_7

For example:
Maxterms

x	y	z	Maxterm	
0	0	0	x+y+z	M_0
0	0	1	x+y+z'	M_1
…				
1	0	0	x'+y+z	M_4
…				
1	1	1	x'+y'+z'	M_7

Representing Functions with Minterms

- Minterm number same as row position in truth table (starting from top from 0)
- Shorthand way to represent functions

x	y	z	G
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

$$G = xyz + xyz' + x'yz$$

$$\downarrow$$

$$G = m_7 + m_6 + m_3 = \Sigma(3, 6, 7)$$

Complementing Functions

- **Minterm number same as row position in truth table (starting from top from 0)**
- **Shorthand way to represent functions**

x	y	z	G	G'
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	1
1	1	0	1	0
1	1	1	1	0

$G = xyz + xyz' + x'yz$

$G' = (xyz + xyz' + x'yz)' =$

Can we find a simpler representation?

Complementing Functions

- **Step 1: assign temporary names**
  - b + c -> z
  - (a + z)’ = G’
- **Step 2: Use DeMorgans’ Law**
  - (a + z)’ = a’ . z’
- **Step 3: Resubstitute (b+c) for z**
  - a’ . z’ = a’ . (b + c)’
- **Step 4: Use DeMorgans’ Law**
  - a’ . (b + c)’ = a’ . (b’. c’)
- **Step 5: Associative rule**
  - a’ . (b’. c’) = a’ . b’ . c’

G = a + b + c

G’ = (a + b + c)’

G = a + b + c

G’ = a’ . b’ . c’ = a’b’c’

Complementation Example

- **Find complement of $F = x'z + yz$**
  - $F' = (x'z + yz)'$
- **DeMorgan's**
  - $F' = (x'z)' (yz)'$
- **DeMorgan's**
  - $F' = (x''+z')(y'+z')$
- **Reduction -> eliminate double negation on x**
  - $F' = (x+z')(y'+z')$

This format is called product of sums

Conversion Between Canonical Forms

- **Easy to convert between minterm and maxterm representations**
- **For maxterm representation, select rows with 0's**

x	y	z	G
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

$$G = xyz + xyz' + x'yz$$

$$\downarrow$$

$$G = m_7 + m_6 + m_3 = \Sigma(3, 6, 7)$$

$$\downarrow$$

$$G = M_0 M_1 M_2 M_4 M_5 = \Pi(0,1,2,4,5)$$

$$\downarrow$$

$$G = (x+y+z)(x+y+z')(x+y'+z)(x'+y+z)(x'+y+z')$$

Representation of Circuits

- All logic expressions can be represented in 2-level format
- Circuits can be reduced to minimal 2-level representation
- Sum of products representation most common in industry.

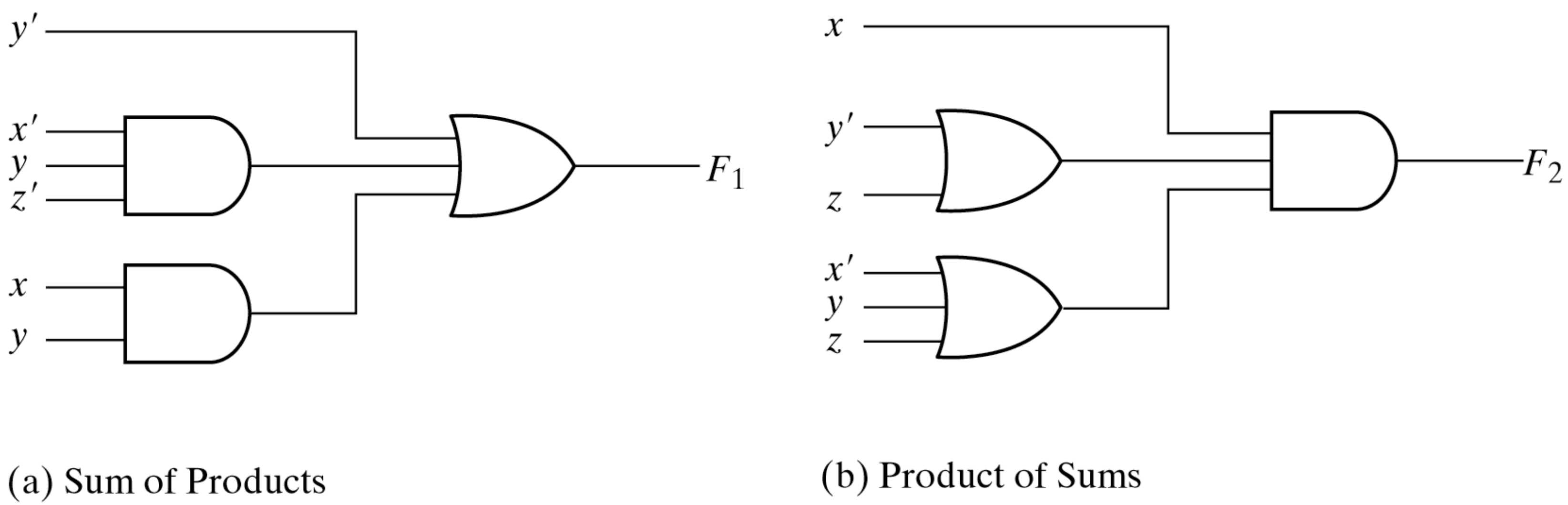


Fig. 2-3 Two-level implementation

Summary

- Truth table, circuit, and boolean expression formats are equivalent
- Easy to translate truth table to SOP and POS representation
- Boolean algebra rules can be used to reduce circuit size while maintaining function
- All logic functions can be made from AND, OR, and NOT
- Easiest way to understand: Do examples!
- Next time: More logic gates!