Revised: 5/9/2024

Digital Logic I

Michael C. Hackett

Assistant Professor, Computer Science

COMMUNITY COLLEGE OF PHILADELPHIA

Lecture Topics

- Boolean Functions
- Boolean Expressions
- Simplifying Boolean Expressions
    - Karnaugh Maps
- Logic Gates and Circuits
- Buses
- Sign Extends

Boolean Functions

- We’ve become familiar with logical operations (and, or, not…) while working with and assembly language (and high-level languages in other courses)
- With more complex logical expressions, we use special notations to represent Boolean functions.
  - Some of these notations were seen in a previous lecture

Boolean Functions

$$x\ AND\ y$$

may be expressed as either:

$$x \cdot y$$

$$xy$$

x	y	$x\ AND\ y$
0	0	0
0	1	0
1	0	0
1	1	1

Boolean Functions

$$x\ OR\ y$$

is expressed as:

$$x + y$$

x	y	$x\ OR\ y$
0	0	0
0	1	1
1	0	1
1	1	1

Boolean Functions

$$NOT\ x$$

may be expressed as either:

x' *("x prime")*

$\overline{x}$ *("x bar" or "not x")*

x	$NOT\ x$
0	1
1	0

Boolean Functions

$$x\ XOR\ y$$

is expressed as:

$$\bar{x}y + x\bar{y}$$
$$x \oplus y$$

x	y	$x\ XOR\ y$
0	0	0
0	1	1
1	0	1
1	1	0

Boolean Functions

- An **NOR** operation results in true (1) only when **both** of the operands are false (0)
    - It is the negation of the **or** operator

x	y	$x\ NOR\ y$
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**0**

$x\ NOR\ y$

is expressed as:

$$\overline{x + y}$$

Boolean Functions

- A **NAND** operation results in true (1) only when **both** of the operands are *not* true (1)
  - It is the negation of the **and** operator

$x\ NAND\ y$

may be expressed as either:

$\overline{x \cdot y}$

$\overline{xy}$

x	y	$x\ NAND\ y$
0	0	**1**
0	1	**1**
1	0	**1**
1	1	**0**

Boolean Functions

- An **XNOR** (**ex**clusive **nor**, “*x nor*”; “*znor*”) operation results in true (1) only when the operands are either both true (1) or both false (0)
  - It is the negation of the **xor** operator

$x\ XNOR\ y$

is expressed as:

$\overline{x}\,\overline{y} + xy$

$x \odot y$

x	y	$x\ XNOR\ y$
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**1**

Boolean Functions

- The precedence for Boolean operators is
  1. *Expressions in parentheses*
  2. NOT
  3. AND
  4. OR

- $x\ NOR\ y = NOT(x\ OR\ y)$
- $x\ NAND\ y = NOT(x\ AND\ y)$
- $x\ XNOR\ y = NOT(x\ XOR\ y)$
- $x\ XOR\ y = (NOT(x)\ AND\ y)\ OR\ (x\ AND\ NOT(y))$

Boolean Expressions

- Boolean expressions can be evaluated using truth tables
  - Be sure to follow the order of operations

$$\overline{x} + y$$

		1	2
x	y	$\bar{x}$	$\overline{\boldsymbol{x}} + \boldsymbol{y}$
0	0	1	**1**
0	1	1	**1**
1	0	0	**0**
1	1	0	**1**

Boolean Expressions

- Boolean expressions can be evaluated using truth tables

$$x + xy$$

		1	2
x	y	xy	$\boldsymbol{x + xy}$
0	0	0	**0**
0	1	0	**0**
1	0	0	**1**
1	1	1	**1**

Boolean Expressions

- Boolean expressions can be evaluated using truth tables

$$\overline{x} + xy + \overline{y}$$

		1	2	3	4	5
x	y	$\bar{x}$	$\bar{y}$	xy	$\bar{x} + xy$	$\boldsymbol{\bar{x} + xy + \bar{y}}$
0	0	1	1	0	1	**1**
0	1	1	0	0	1	**1**
1	0	0	1	0	0	**1**
1	1	0	0	1	1	**1**

- This truth table tells us that, regardless of the value of x or y, the result will always be 1

Boolean Expressions

- Boolean expressions can be evaluated using truth tables

$$x + y\bar{z}$$

			1	2	3
x	y	z	$\bar{z}$	$y\bar{z}$	$\boldsymbol{x + y\bar{z}}$
0	0	0	1	0	**0**
0	0	1	0	0	**0**
0	1	0	1	1	**1**
0	1	1	0	0	**0**
1	0	0	1	0	**1**
1	0	1	0	0	**1**
1	1	0	1	1	**1**
1	1	1	0	0	**1**

Boolean Expressions

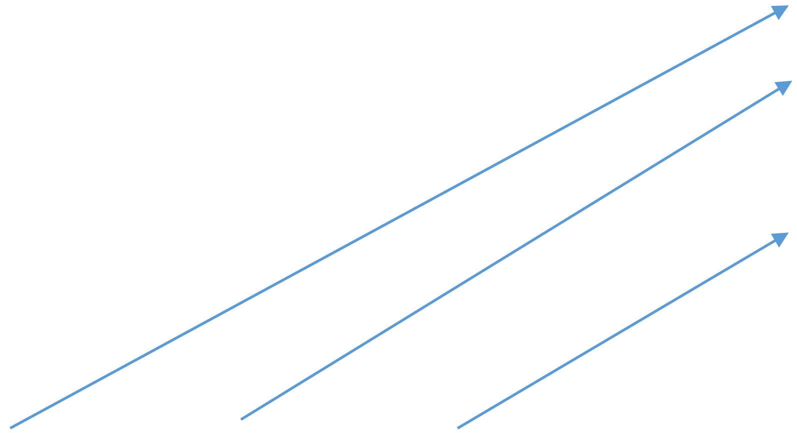
- We'll take the previous expressions and convert them to **sum-of-products form** (or, *disjunctive normal form*)
- This form can be obtained with a truth table
- We are interested in the scenarios where the output of the function is 1 (true)

Boolean Expressions

- Converting to Sum-of-Products

$$\overline{x} + y$$

x	y	$\bar{x}$	$\bar{x} + y$
0	**0**	1	**1**
0	**1**	1	**1**
1	0	0	0
1	**1**	0	**1**



$$\bar{x}\bar{y} + \bar{x}y + xy$$

Boolean Expressions

- Proving their identity

$$\overline{x} + y = \bar{x}\bar{y} + \bar{x}y + xy$$

				Original				SOP Form
x	y	$\bar{x}$	$\bar{y}$	$\bar{x} + y$	$\bar{x}\bar{y}$	$\bar{x}y$	xy	$\bar{x}\bar{y} + \bar{x}y + xy$
0	0	1	1	**1**	1	0	0	**1**
0	1	1	0	**1**	0	1	0	**1**
1	0	0	1	**0**	0	0	0	**0**
1	1	0	0	**1**	0	0	1	**1**

Boolean Expressions

- Converting to Sum-of-Products

$$x + xy$$

x	y	xy	$x + xy$
0	0	0	0
0	1	0	0
1	**0**	0	**1**
1	**1**	1	**1**

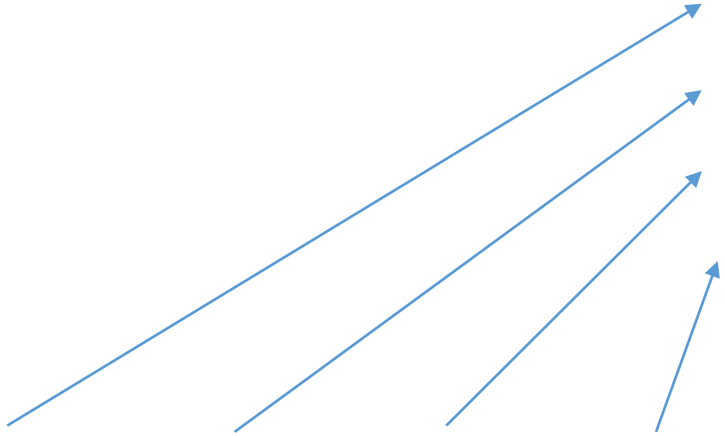
$$x\bar{y} + xy$$

Boolean Expressions

- Converting to Sum-of-Products

$$\overline{x} + xy + \overline{y}$$

x	y	$\bar{x}$	$\bar{y}$	xy	$\bar{x} + xy$	$\bar{x} + xy + \bar{y}$
0	**0**	1	1	0	1	**1**
0	**1**	1	0	0	1	**1**
1	**0**	0	1	0	0	**1**
1	**1**	0	0	1	1	**1**



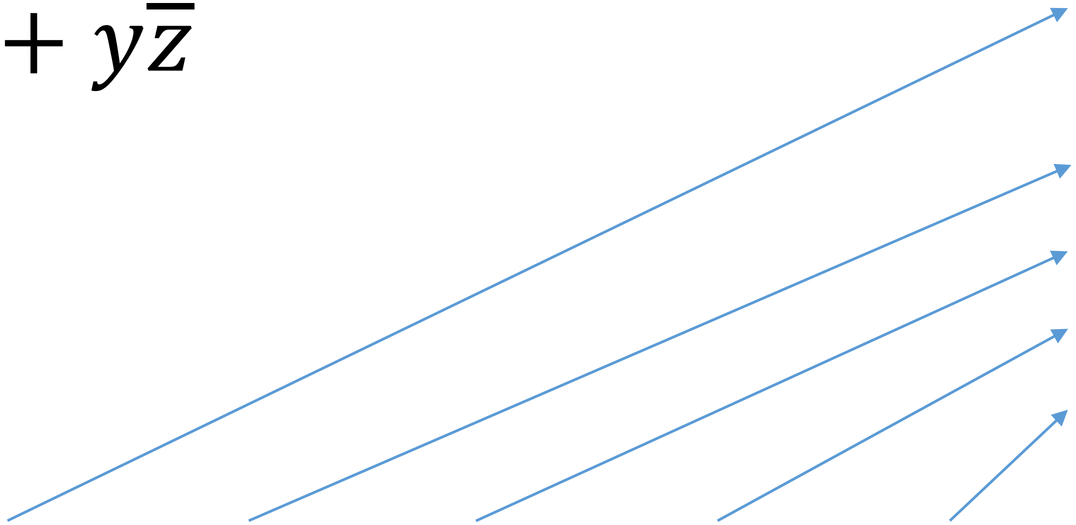
$$\bar{x}\bar{y} + \bar{x}y + x\bar{y} + xy$$

Boolean Expressions

- Converting to Sum-of-Products

$$x + y\overline{z}$$

x	y	z	$\bar{z}$	$y\bar{z}$	$x + y\bar{z}$
0	0	0	1	0	0
0	0	1	0	0	0
0	**1**	**0**	1	1	**1**
0	1	1	0	0	0
1	**0**	**0**	1	0	**1**
1	**0**	**1**	0	0	**1**
1	**1**	**0**	1	1	**1**
1	**1**	**1**	0	0	**1**



$$\bar{x}y\bar{z} + x\bar{y}\bar{z} + x\bar{y}z + xy\bar{z} + xyz$$

Boolean Expressions

- We are now ready to look at some laws of Boolean algebra that will let us simplify Boolean expressions to the fewest terms and operations needed.

$$\overline{x} + y = \bar{x}\overline{y} + \bar{x}y + xy$$

- Though these expressions are logically equivalent, the first expression uses fewer terms and operations than the second.

Boolean Expressions

- Identity Laws

$$x \cdot 1 = x$$

$$x + 0 = x$$

x	$x \cdot 1$	$x + 0$
0	**0**	**0**
1	**1**	**1**

Boolean Expressions

- Constant Laws

$$x \cdot 0 = 0$$

$$x + 1 = 1$$

x	$x \cdot 0$	$x + 1$
0	**0**	**1**
1	**0**	**1**

Boolean Expressions

- Negation Laws

$$x \cdot \overline{x} = 0$$

$$x + \overline{x} = 1$$

x	$\overline{x}$	$x \cdot \overline{x}$	$x + \overline{x}$
0	1	**0**	**1**
1	0	**0**	**1**

Boolean Expressions

- Double Negation Law

$$\bar{\bar{x}} = x$$

x	$\bar{x}$	$\bar{\bar{x}}$
0	1	**0**
1	0	**1**

Boolean Expressions

- Idempotent Laws

$$x \cdot x = x$$

$$x + x = x$$

x	$x \cdot x$	$x + x$
0	0	0
1	1	1

Boolean Expressions

- Commutative Laws

$$x \cdot y = y \cdot x$$

$$x + y = y + x$$

x	y	$x \cdot y$	$y \cdot x$
0	0	**0**	**0**
0	1	**0**	**0**
1	0	**0**	**0**
1	1	**1**	**1**

x	y	$x + y$	$y + x$
0	0	**0**	**0**
0	1	**1**	**1**
1	0	**1**	**1**
1	1	**1**	**1**

Boolean Expressions

- Distributive Laws

$$x \cdot (y + z) = xy + xz$$

x	y	z	$(y + z)$	$\boldsymbol{x \cdot (y + z)}$
0	0	0	0	**0**
0	0	1	1	**0**
0	1	0	1	**0**
0	1	1	1	**0**
1	0	0	0	**0**
1	0	1	1	**1**
1	1	0	1	**1**
1	1	1	1	**1**

x	y	z	xy	xz	$\boldsymbol{xy + xz}$
0	0	0	0	0	**0**
0	0	1	0	0	**0**
0	1	0	0	0	**0**
0	1	1	0	0	**0**
1	0	0	0	0	**0**
1	0	1	0	1	**1**
1	1	0	1	0	**1**
1	1	1	1	1	**1**

Boolean Expressions

- Distributive Laws

$$x + (yz) = (x + y) \cdot (x + z)$$

x	y	z	(yz)	$\boldsymbol{x + (yz)}$
0	0	0	0	**0**
0	0	1	0	**0**
0	1	0	0	**0**
0	1	1	1	**1**
1	0	0	0	**1**
1	0	1	0	**1**
1	1	0	0	**1**
1	1	1	2	**1**

x	y	z	$(x + y)$	$(x + z)$	$\boldsymbol{(x + y) \cdot (x + z)}$
0	0	0	0	0	**0**
0	0	1	0	1	**0**
0	1	0	1	0	**0**
0	1	1	1	1	**1**
1	0	0	1	1	**1**
1	0	1	1	1	**1**
1	1	0	1	1	**1**
1	1	1	1	1	**1**

Boolean Expressions

- Associative Laws

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z$$

x	y	z	$(y \cdot z)$	$\boldsymbol{x \cdot (y \cdot z)}$
0	0	0	0	**0**
0	0	1	0	**0**
0	1	0	0	**0**
0	1	1	1	**0**
1	0	0	0	**0**
1	0	1	0	**0**
1	1	0	0	**0**
1	1	1	1	**1**

x	y	z	$(x \cdot y)$	$\boldsymbol{(x \cdot y) \cdot z}$
0	0	0	0	**0**
0	0	1	0	**0**
0	1	0	0	**0**
0	1	1	0	**0**
1	0	0	0	**0**
1	0	1	0	**0**
1	1	0	1	**0**
1	1	1	1	**1**

Boolean Expressions

- Associative Laws

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

x	y	z	$(y + z)$	$\boldsymbol{x + (y + z)}$
0	0	0	0	**0**
0	0	1	1	**1**
0	1	0	1	**1**
0	1	1	1	**1**
1	0	0	0	**1**
1	0	1	1	**1**
1	1	0	1	**1**
1	1	1	1	**1**

x	y	z	$(x + y)$	$\boldsymbol{(x + y) + z}$
0	0	0	0	**0**
0	0	1	0	**1**
0	1	0	1	**1**
0	1	1	1	**1**
1	0	0	1	**1**
1	0	1	1	**1**
1	1	0	1	**1**
1	1	1	1	**1**

Boolean Expressions

- DeMorgan's Laws

$$\overline{x + y} = \bar{x} \cdot \bar{y}$$

x	y	$x + y$	$\overline{\boldsymbol{x} + \boldsymbol{y}}$
0	0	0	**1**
0	1	1	**0**
1	0	1	**0**
1	1	1	**0**

x	y	$\bar{x}$	$\bar{y}$	$\bar{\boldsymbol{x}} \cdot \bar{\boldsymbol{y}}$
0	0	1	1	**1**
0	1	1	0	**0**
1	0	0	1	**0**
1	1	0	0	**0**

Boolean Expressions

- DeMorgan's Laws

$$\overline{x \cdot y} = \bar{x} + \bar{y}$$

x	y	$x \cdot y$	$\overline{x \cdot y}$
0	0	0	**1**
0	1	0	**1**
1	0	0	**1**
1	1	1	**0**

x	y	$\bar{x}$	$\bar{y}$	$\bar{x} + \bar{y}$
0	0	1	1	**1**
0	1	1	0	**1**
1	0	0	1	**1**
1	1	0	0	**0**

Boolean Expressions

- Absorption Laws

$$x + (x \cdot y) = x$$

$$x \cdot (x + y) = x$$

$\boldsymbol{x}$	y	$x \cdot y$	$\boldsymbol{x} + (\boldsymbol{x} \cdot \boldsymbol{y})$
0	0	0	**0**
0	1	0	**0**
1	0	0	**1**
1	1	1	**1**

$\boldsymbol{x}$	y	$x + y$	$\boldsymbol{x} \cdot (\boldsymbol{x} + \boldsymbol{y})$
0	0	0	**0**
0	1	1	**0**
1	0	1	**1**
1	1	1	**1**

Boolean Expressions

- Absorption Laws

$$x + (\bar{x} \cdot y) = x + y$$

$$x \cdot (\bar{x} + y) = x \cdot y$$

x	y	$\bar{x}$	$\boldsymbol{x + y}$	$\bar{x} \cdot y$	$\boldsymbol{x + (\bar{x} \cdot y)}$
0	0	1	**0**	0	**0**
0	1	1	**1**	1	**1**
1	0	0	**1**	0	**1**
1	1	0	**1**	0	**1**

x	y	$\bar{x}$	$\boldsymbol{x \cdot y}$	$\bar{x} + y$	$\boldsymbol{x \cdot (\bar{x} + y)}$
0	0	1	**0**	1	**0**
0	1	1	**0**	1	**0**
1	0	0	**0**	0	**0**
1	1	0	**1**	1	**1**

Boolean Expressions

- Consensus Laws

$$xy + \bar{x}z + yz = xy + \bar{x}z$$

x	y	z	$\bar{x}$	xy	$\bar{x}z$	yz	$\boldsymbol{xy + \bar{x}z}$	$\boldsymbol{xy + \bar{x}z + yz}$
0	0	0	1	0	0	0	**0**	**0**
0	0	1	1	0	1	0	**1**	**1**
0	1	0	1	0	0	0	**0**	**0**
0	1	1	1	0	1	1	**1**	**1**
1	0	0	0	0	0	0	**0**	**0**
1	0	1	0	0	0	0	**0**	**0**
1	1	0	0	1	0	0	**1**	**1**
1	1	1	0	1	0	1	**1**	**1**

Boolean Expressions

- Consensus Laws

$$(x+y)\cdot(\bar{x}+z)\cdot(y+z)=(x+y)\cdot(\bar{x}+z)$$

x	y	z	$\bar{x}$	$x+y$	$\bar{x}+z$	$y+z$	$(x+y)\cdot(\bar{x}+z)$	$(x+y)\cdot(\bar{x}+z)\cdot(y+z)$
0	0	0	1	0	1	0	**0**	**0**
0	0	1	1	0	1	1	**0**	**0**
0	1	0	1	1	1	1	**1**	**1**
0	1	1	1	1	1	1	**1**	**1**
1	0	0	0	1	0	0	**0**	**0**
1	0	1	0	1	1	1	**1**	**1**
1	1	0	0	1	0	1	**0**	**0**
1	1	1	0	1	1	1	**1**	**1**

Boolean Expressions

- An earlier example:

$$\overline{x} + y = \bar{x}\overline{y} + \bar{x}y + xy$$

- Let’s use any applicable laws to the SOP expression:

$$\bar{x}\overline{y} + \bar{x}y + xy$$

- To arrive at its equivalent:

$$\overline{x} + y$$

Boolean Expressions

$$\bar{x}\bar{y} + \bar{x}y + xy$$

- Distributive Law:
  $x \cdot (y + z) = \boldsymbol{xy} + \boldsymbol{xz}$

$$\boldsymbol{\bar{x}\bar{y}} + \boldsymbol{\bar{x}y} + xy$$

$$\bar{x}(\bar{y} + y) + xy$$

- Negation Law:
  $\boldsymbol{x} + \boldsymbol{\bar{x}} = 1$

$$\bar{x}(\boldsymbol{\bar{y}} + \boldsymbol{y}) + xy$$

$$\bar{x}(1) + xy$$

- Identity Law:
  $\boldsymbol{x} \cdot \boldsymbol{1} = x$

$$\boldsymbol{\bar{x}}(\boldsymbol{1}) + xy$$

$$\bar{x} + xy$$

- Absorption Law:
  $\boldsymbol{x} + (\boldsymbol{\bar{x}} \cdot \boldsymbol{y}) = x + y$

$$\boldsymbol{\bar{x}} + \boldsymbol{xy}$$

$$\bar{x} + y$$

Boolean Expressions

- An earlier example:

$$x + xy = x\bar{y} + xy$$

was proven to be true with a truth table

- Let’s use any applicable laws to this SOP expression to see if the Boolean expression can be simplified further:

$$x\bar{y} + xy$$

Boolean Expressions

Distributive Law:

$x \cdot (y + z) = \boldsymbol{xy} + \boldsymbol{xz}$

- Negation Law:

  $\boldsymbol{x} + \overline{\boldsymbol{x}} = 1$

- Identity Law:

  $\boldsymbol{x} \cdot \boldsymbol{1} = x$

$x\bar{y} + xy$

$\boldsymbol{x}\overline{\boldsymbol{y}} + \boldsymbol{xy}$

$x(\bar{y} + y)$

$x(\overline{\boldsymbol{y}} + \boldsymbol{y})$

$x(1)$

$\boldsymbol{x}(\boldsymbol{1})$

x

Boolean Expressions

$$\boldsymbol{x + xy = \ x\bar{y} + xy = x}$$

$\boldsymbol{x}$	y	$\bar{y}$	xy	$x\bar{y}$	$\boldsymbol{x + xy}$	$\boldsymbol{x\bar{y} + xy}$
0	0	1	0	0	**0**	**0**
0	1	0	0	0	**0**	**0**
1	0	1	0	1	**1**	**1**
1	1	0	1	1	**1**	**1**

Boolean Expressions

- An earlier example:

$$\overline{x} + xy + \overline{y} = \bar{x}\bar{y} + \bar{x}y + x\bar{y} + xy$$

- We not only proved this to be true with truth tables, but we also saw the result is always 1

x	y	$\bar{x}$	$\bar{y}$	xy	$\bar{x} + xy$	$\bar{x} + xy + \bar{y}$
0	0	1	1	0	1	**1**
0	1	1	0	0	1	**1**
1	0	0	1	0	0	**1**
1	1	0	0	1	1	**1**

- Let’s also prove it true using any applicable laws

Boolean Expressions

$\bar{x}\bar{y} + \bar{x}y + x\bar{y} + xy$

- Distributive Law:
  $x \cdot (y + z) = \boldsymbol{xy} + \boldsymbol{xz}$

$\boldsymbol{\bar{x}}\,\boldsymbol{\bar{y}} + \boldsymbol{\bar{x}y} + x\bar{y} + xy$

$\bar{x}(\bar{y} + y) + x\bar{y} + xy$

- Negation Law:
  $\boldsymbol{x} + \boldsymbol{\bar{x}} = 1$

$\bar{x}(\boldsymbol{\bar{y}} + \boldsymbol{y}) + x\bar{y} + xy$

$\bar{x}(1) + x\bar{y} + xy$

- Identity Law:
  $\boldsymbol{x} \cdot \boldsymbol{1} = x$

$\boldsymbol{\bar{x}}(\boldsymbol{1}) + x\bar{y} + xy$

$\bar{x} + x\bar{y} + xy$

$\bar{x} + x\bar{y} + xy$

- Distributive Law:
  $x \cdot (y + z) = \boldsymbol{xy} + \boldsymbol{xz}$

$\bar{x} + \boldsymbol{x\bar{y}} + \boldsymbol{xy}$

$\bar{x} + x(\bar{y} + y)$

- Negation Law:
  $\boldsymbol{x} + \boldsymbol{\bar{x}} = 1$

$\bar{x} + x(\boldsymbol{\bar{y}} + \boldsymbol{y})$

$\bar{x} + x(1)$

- Identity Law:
  $\boldsymbol{x} \cdot \boldsymbol{1} = x$

$\bar{x} + \boldsymbol{x}(\boldsymbol{1})$

$\bar{x} + x$

- Negation Law:
  $\boldsymbol{x} + \boldsymbol{\bar{x}} = 1$

$\boldsymbol{\bar{x}} + \boldsymbol{x}$

$\boldsymbol{1}$

Boolean Expressions

- An earlier example:

$$x + y\overline{z} = \bar{x}y\bar{z} + x\bar{y}\bar{z} + x\bar{y}z + xy\bar{z} + xyz$$

was proven to be true with a truth table

- Let’s use any applicable laws to the SOP expression:

$$\bar{x}y\bar{z} + x\bar{y}\bar{z} + x\bar{y}z + xy\bar{z} + xyz$$

- To arrive at its equivalent:

$$x + y\overline{z}$$

Boolean Expressions

$\bar{x}y\bar{z} + \boldsymbol{x\bar{y}\bar{z}} + \boldsymbol{x\bar{y}z} + \boldsymbol{xy\bar{z}} + \boldsymbol{xyz}$

- Distributive Law: $\boldsymbol{x}(\boldsymbol{\bar{y}\bar{z}} + \boldsymbol{\bar{y}z} + \boldsymbol{y\bar{z}} + \boldsymbol{yz}) + \bar{x}y\bar{z}$

$x(\boldsymbol{\bar{y}\bar{z}} + \boldsymbol{\bar{y}z} + \boldsymbol{y\bar{z}} + \boldsymbol{yz}) + \bar{x}y\bar{z}$

- Distributive Law (x2): $x(\boldsymbol{\bar{y}}(\boldsymbol{\bar{z}} + \boldsymbol{z}) + \boldsymbol{y}(\boldsymbol{\bar{z}} + \boldsymbol{z})) + \bar{x}y\bar{z}$

$x(\bar{y}(\boldsymbol{\bar{z}} + \boldsymbol{z}) + y(\boldsymbol{\bar{z}} + \boldsymbol{z})) + \bar{x}y\bar{z}$

- Negation Law (x2): $x(\bar{y}(\boldsymbol{1}) + y(\boldsymbol{1})) + \bar{x}y\bar{z}$
- Identity Law (x2): $x(\bar{y} + y) + \bar{x}y\bar{z}$

$x(\boldsymbol{\bar{y}} + \boldsymbol{y}) + \bar{x}y\bar{z}$

- Negation Law: $x(\boldsymbol{1}) + \bar{x}y\bar{z}$
- Identity Law: $x + \bar{x}y\bar{z}$

$\boldsymbol{x} + \boldsymbol{\bar{x}y\bar{z}}$

- Absorption Law: $x + y\bar{z}$
  $x + (\bar{x} \cdot y) = x + y$

Simplifying Boolean Expressions

- Those laws are useful for reducing expressions, but it can be cumbersome to reduce very complex expression this way

- A tool used to help with the simplification process is a Karnaugh Map
    - K-Map, for short

- K-Maps are not a total substitute for Boolean algebra, but they are faster

Simplifying Boolean Expressions

- Below is a two-variable K-Map
  - Notice that only one bit changes between each row and column

		$\bar{y}$	y
	$\boldsymbol{x} \backslash \boldsymbol{y}$	0	1
$\bar{x}$	0	00	01
x	1	10	11

Simplifying Boolean Expressions

- To demonstrate, we'll use the following expression:

$$x + \overline{y}$$

- Which yields the following truth table:

x	y	$\bar{y}$	$x + \overline{y}$
0	0	1	**1**
0	1	0	**0**
1	0	1	**1**
1	1	0	**1**

Simplifying Boolean Expressions

x	y	$\bar{y}$	$x + \bar{y}$
0	**0**	1	**1**
0	1	0	0
1	0	1	1
1	1	0	1

$\boldsymbol{x}$ \ $\boldsymbol{y}$	$\bar{y}$ **0**	y 1
$\bar{x}$ **0**	**1** **00**	01
x 1	10	11

Simplifying Boolean Expressions

x	y	$\bar{y}$	$x+\bar{y}$
0	0	1	1
0	**1**	0	**0**
1	0	1	1
1	1	0	1

		$\bar{y}$	y
	$\boldsymbol{x} \backslash \boldsymbol{y}$	0	**1**
$\bar{x}$	**0**	1 00	**0** **01**
x	1	10	11

Simplifying Boolean Expressions

x	y	$\bar{y}$	$x + \bar{y}$
0	0	1	1
0	1	0	0
1	**0**	1	**1**
1	1	0	1

	$\boldsymbol{y}$ $\boldsymbol{x}$	$\bar{y}$ **0**	y 1
$\bar{x}$	0	1 00	0 01
x	**1**	**1** **10**	11

Simplifying Boolean Expressions

x	y	$\bar{y}$	$x+\bar{y}$
0	0	1	1
0	1	0	0
1	0	1	1
1	**1**	0	**1**

$\boldsymbol{x}$ \ $\boldsymbol{y}$		$\bar{y}$	y
		0	**1**
$\bar{x}$	0	1 00	0 01
x	**1**	1 10	**1** **11**

Simplifying Boolean Expressions

- K-Map for $x + \overline{y}$

$\boldsymbol{y}$ $\boldsymbol{x}$		$\bar{y}$ 0	y 1
$\bar{x}$	0	1 00	0 01
x	1	1 10	1 11

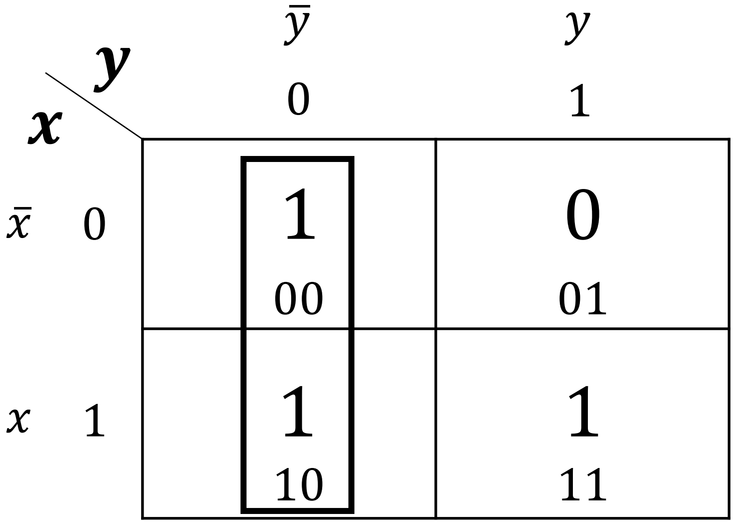
Simplifying Boolean Expressions

- Resulting SOP Expression: $\overline{x}\,\overline{y} + x\overline{y} + xy$

		$\bar{y}$	y
$\boldsymbol{x}$	$\boldsymbol{y}$	0	1
$\bar{x}$	0	**1** 00	**0** 01
x	1	**1** 10	**1** 11

Simplifying Boolean Expressions

- When using K-Maps, we can group adjacent cells together in order to simply the function



x \ y		$\bar{y}$ 0	y 1
$\bar{x}$	0	1 00	0 01
x	1	1 10	1 11

- This illustrates the output is 1 when y = 0 (it does not depend on x)

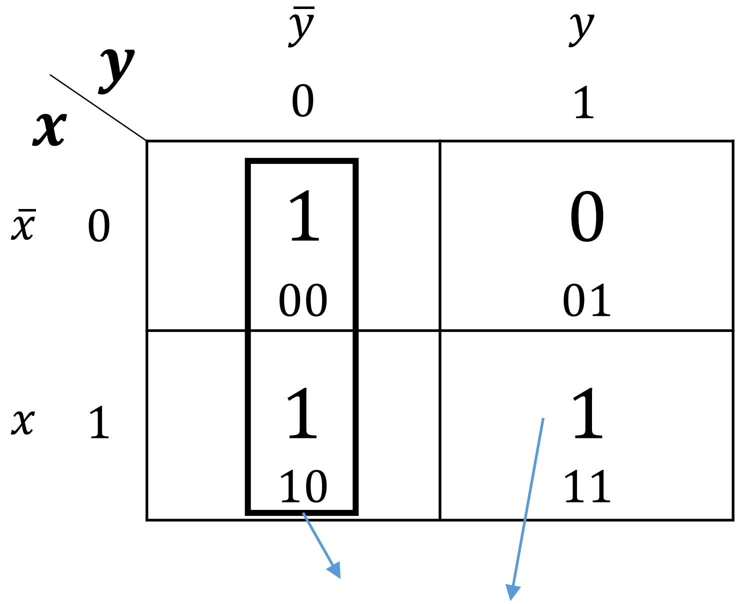
Simplifying Boolean Expressions

- When grouping adjacent cells, you must group them in powers of 2
  - A group of 2, 4, 8, etc.

- The grouped cells must all contain 1s

- Cells cannot be grouped diagonally

Simplifying Boolean Expressions

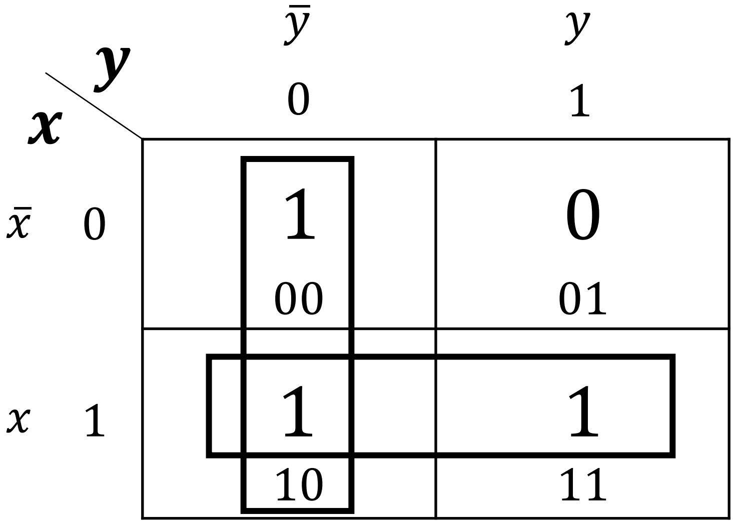


$\boldsymbol{x}$ $\boldsymbol{y}$		$\bar{y}$	y
		0	1
$\bar{x}$	0	1 00	0 01
x	1	1 10	1 11

$$\overline{y} + xy$$

Simplifying Boolean Expressions

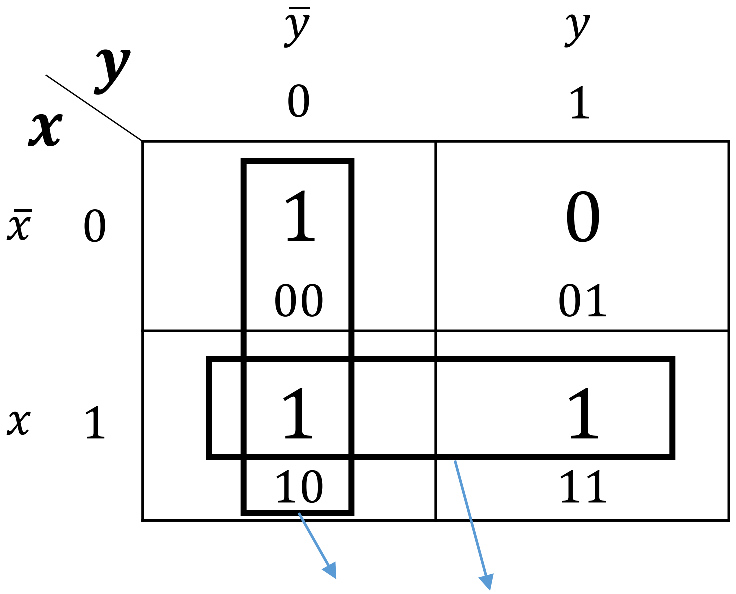
- We can reuse 1’s that were grouped with other adjacent cells



$x \backslash y$	$\bar{y}$ 0	y 1
$\bar{x}$ 0	1 00	0 01
x 1	1 10	1 11

- This second grouping illustrates the output is 1 when x = 1 (it does not depend on y)

Simplifying Boolean Expressions



$x \backslash y$	$\bar{y}$ 0	y 1
$\bar{x}$ 0	1 00	0 01
x 1	1 10	1 11

$$\overline{y} + x$$

- Our original expression was $x + \overline{y}$
  - Already reduced to fewest terms

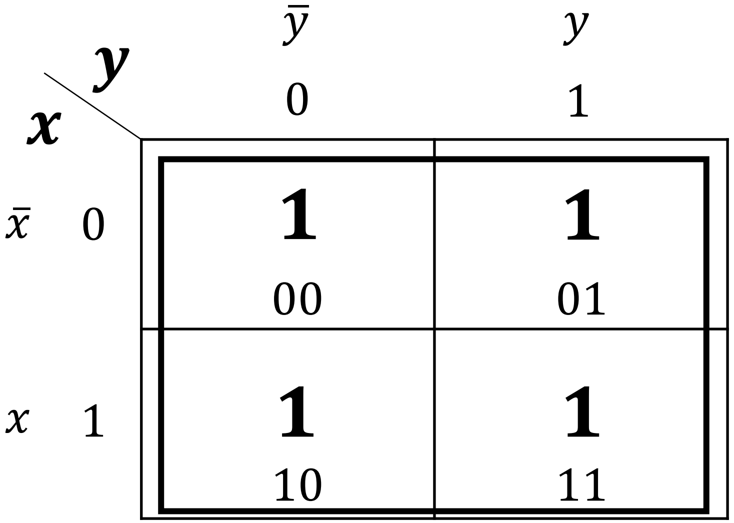
Simplifying Boolean Expressions

- Here's an SOP Expression from earlier: $\bar{x}\bar{y} + \bar{x}y + x\bar{y} + xy$
  - The one that always results in 1

$\boldsymbol{x}$ \ $\boldsymbol{y}$	$\bar{y}$ 0	y 1
$\bar{x}$ 0	**1** 00	**1** 01
x 1	**1** 10	**1** 11

Simplifying Boolean Expressions

- Here, we’ve grouped all 4 (2^2) cells together
  - Just like we saw earlier, it doesn’t matter what the values of x and y are- the function always results in 1



$\boldsymbol{y}$ $\boldsymbol{x}$		$\bar{y}$ 0	y 1
$\bar{x}$	0	**1** 00	**1** 01
x	1	**1** 10	**1** 11

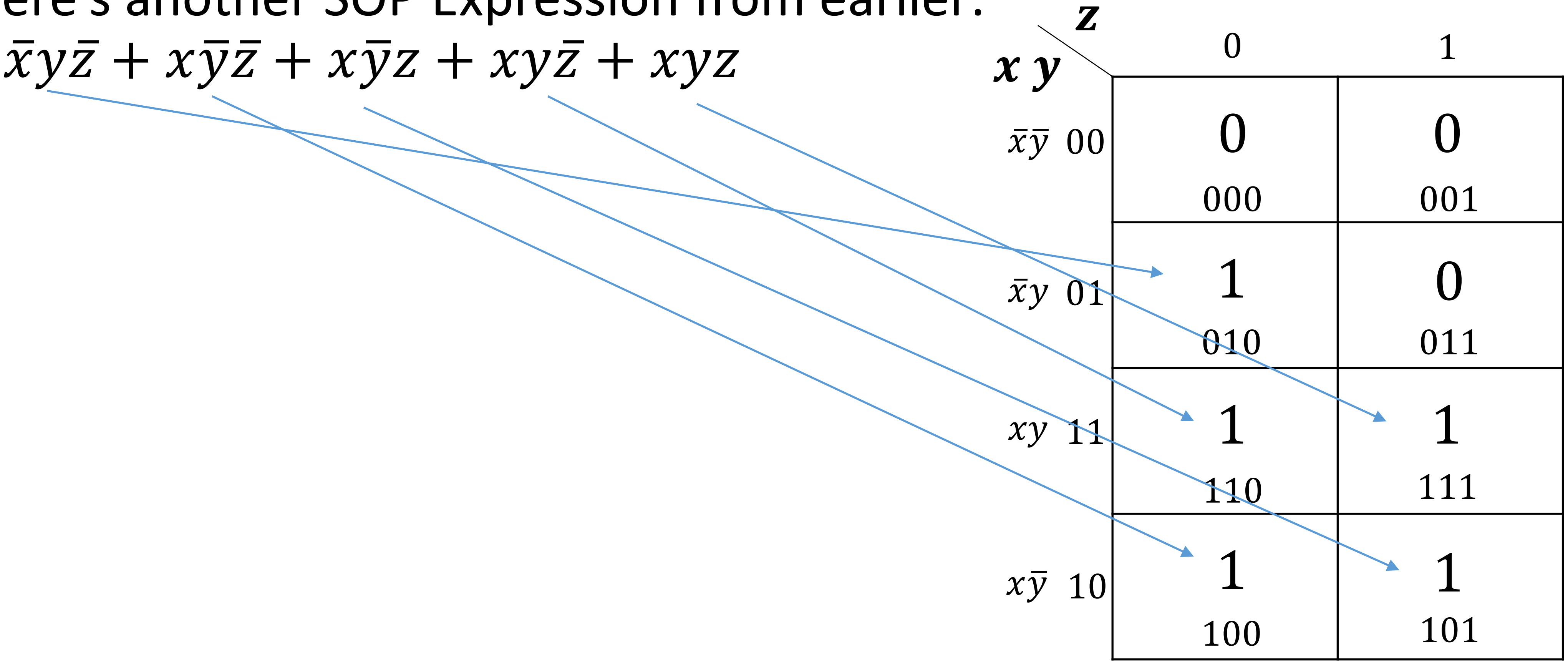
Simplifying Boolean Expressions

- This is a three-variable K-Map
  - Notice that only one bit changes between each row and column

$\boldsymbol{x\ y}$ \ $\boldsymbol{z}$	$\bar{z}$ 0	z 1
$\bar{x}\bar{y}$ 00	0 000	0 001
$\bar{x}y$ 01	0 010	0 011
xy 11	0 110	0 111
$x\bar{y}$ 10	0 100	0 101

Simplifying Boolean Expressions

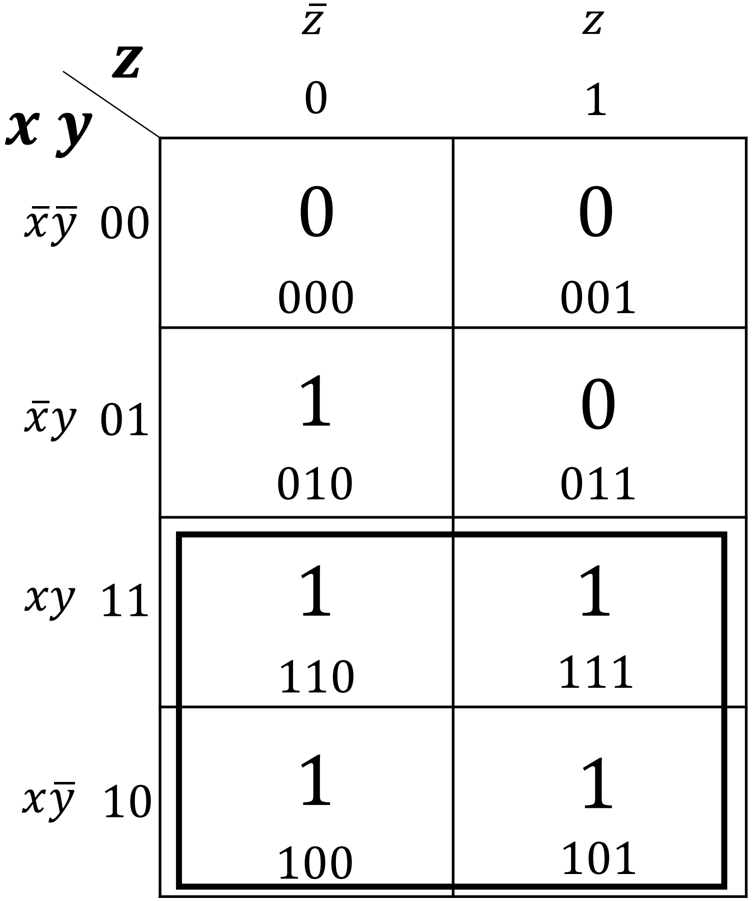
- Here’s another SOP Expression from earlier:
  $\bar{x}y\bar{z} + x\bar{y}\bar{z} + x\bar{y}z + xy\bar{z} + xyz$



$\boldsymbol{z}$ / $\boldsymbol{x\ y}$	$\bar{z}$ 0	z 1
$\bar{x}\bar{y}$ 00	0 000	0 001
$\bar{x}y$ 01	1 010	0 011
xy 11	1 110	1 111
$x\bar{y}$ 10	1 100	1 101

Simplifying Boolean Expressions

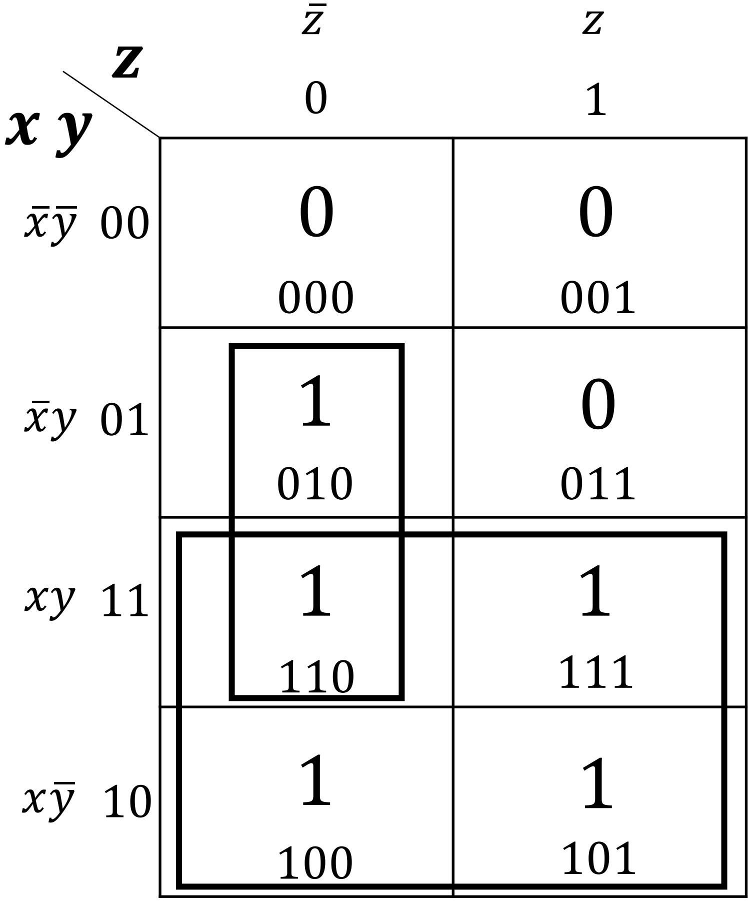
- First grouping (4 cells)
- Result is 1 when x is 1
  - y and z are irrelevant



$\boldsymbol{x\,y}$ \ $\boldsymbol{z}$	$\bar{z}$ 0	z 1
$\bar{x}\bar{y}$ 00	0 000	0 001
$\bar{x}y$ 01	1 010	0 011
xy 11	1 110	1 111
$x\bar{y}$ 10	1 100	1 101

Simplifying Boolean Expressions

- Second grouping (2 cells)
- Result is 1 when y is 1 and z is 0
  - x is irrelevant



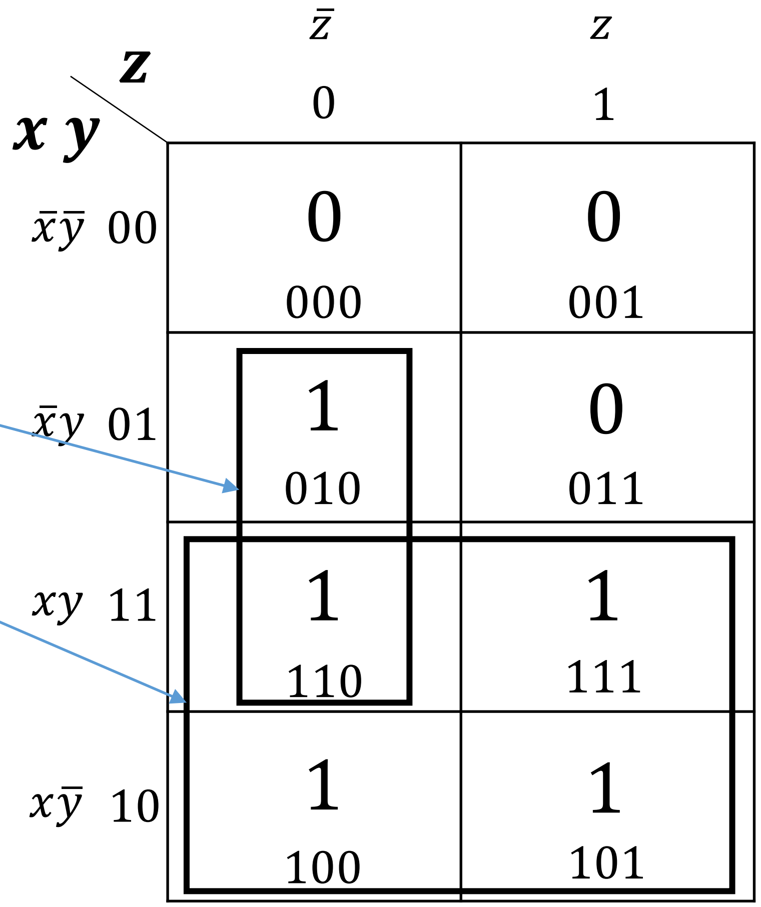
x y \ ***z***	$\bar{z}$ 0	z 1
$\bar{x}\bar{y}$ 00	0 000	0 001
$\bar{x}y$ 01	1 010	0 011
xy 11	1 110	1 111
$x\bar{y}$ 10	1 100	1 101

Simplifying Boolean Expressions

- Result:

$$x + y\bar{z}$$

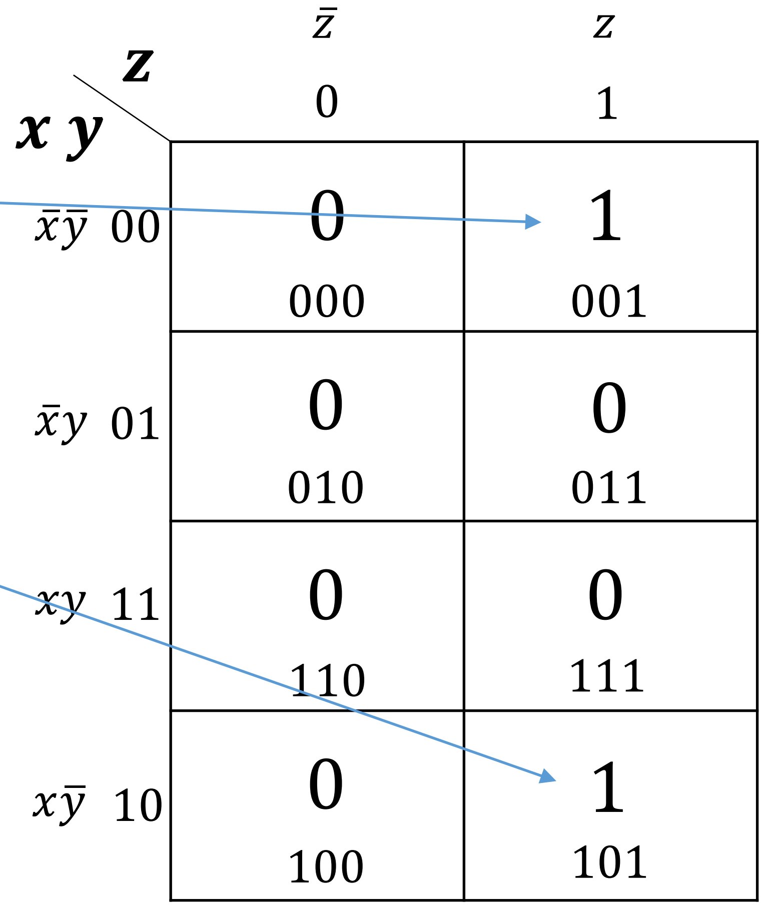
(Same as was derived with Boolean algebra)



$\boldsymbol{x\,y}$ \ $\boldsymbol{z}$	$\bar{z}$ 0	z 1
$\bar{x}\bar{y}$ 00	0 000	0 001
$\bar{x}y$ 01	1 010	0 011
xy 11	1 110	1 111
$x\bar{y}$ 10	1 100	1 101

Simplifying Boolean Expressions

- Another (new) SOP Expression:
  $x\bar{y}z + \bar{x}\bar{y}z$



$\boldsymbol{x\,y}$ \ $\boldsymbol{z}$	$\bar{z}$ 0	z 1
$\bar{x}\bar{y}$ 00	0 000	1 001
$\bar{x}y$ 01	0 010	0 011
xy 11	0 110	0 111
$x\bar{y}$ 10	0 100	1 101

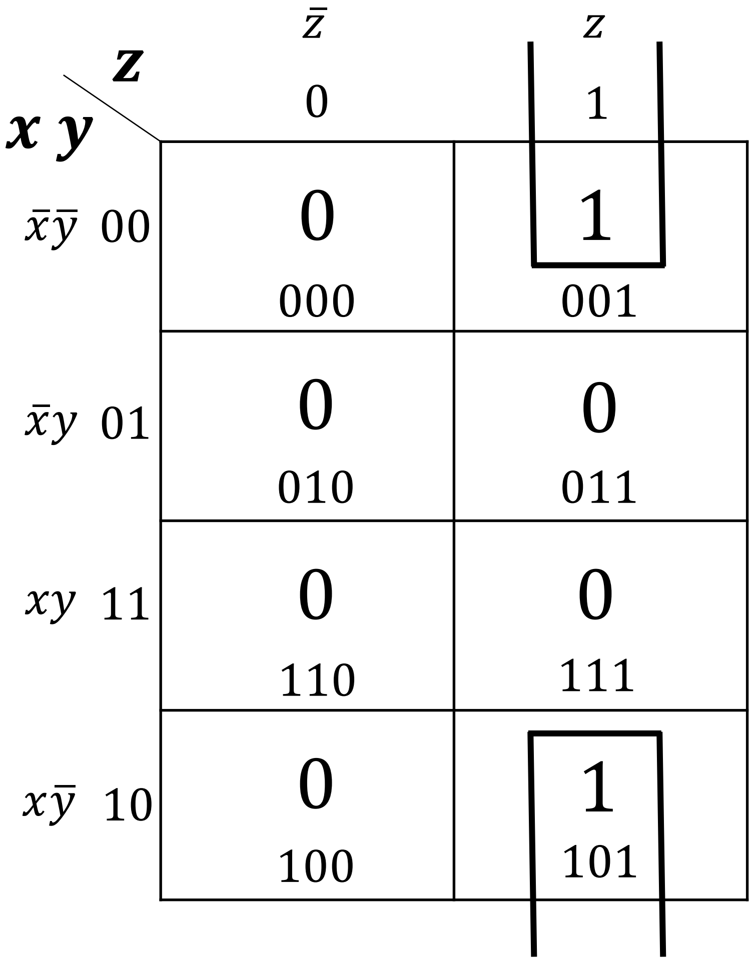
Simplifying Boolean Expressions

- Cell groups can wrap around the map

- Result is 1 when z is 1 and y is 0
  - x is irrelevant

- Result:

$$x\bar{y}z + \bar{x}\bar{y}z = \boldsymbol{\bar{y}z}$$

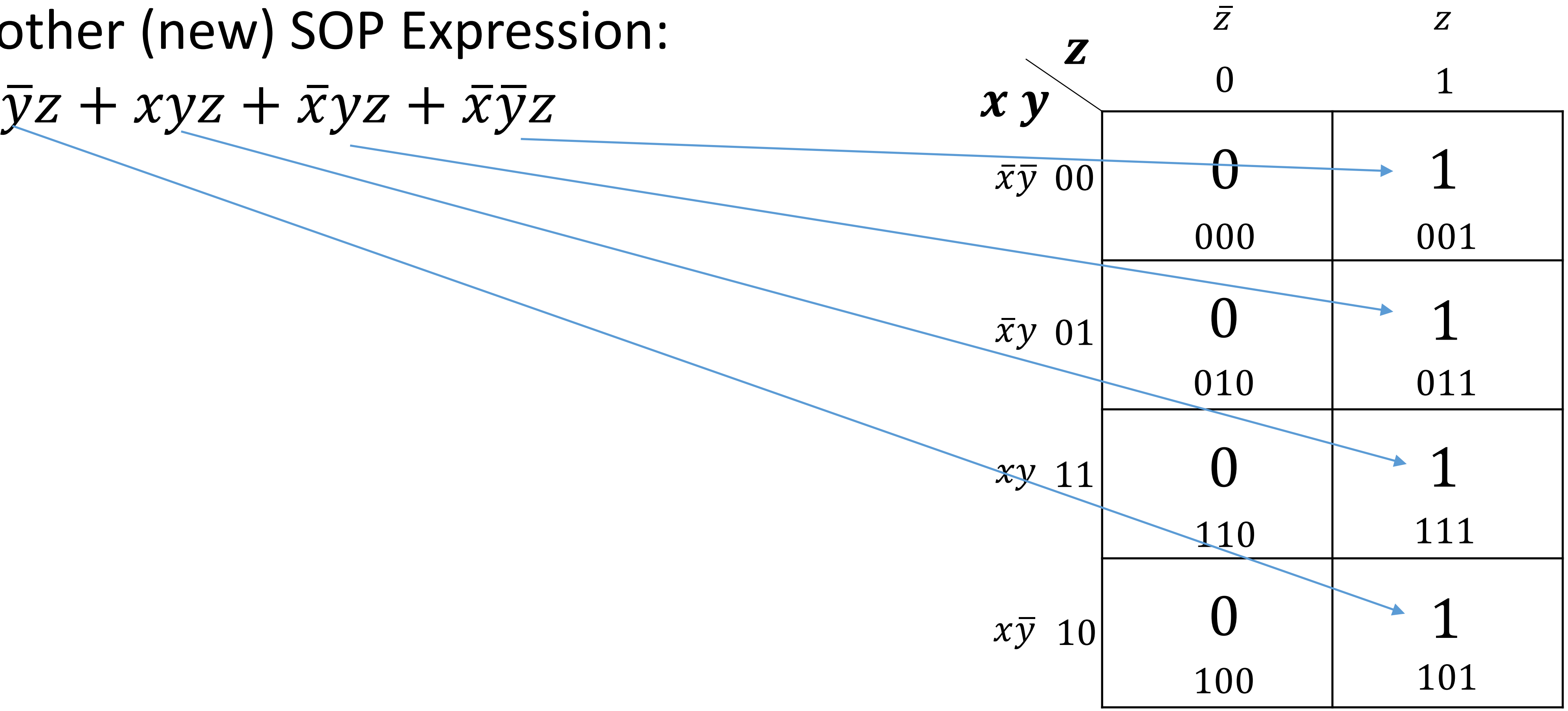


$\boldsymbol{x\,y}$ \ $\boldsymbol{z}$	$\bar{z}$ 0	z 1
$\bar{x}\bar{y}$ 00	0 000	1 001
$\bar{x}y$ 01	0 010	0 011
xy 11	0 110	0 111
$x\bar{y}$ 10	0 100	1 101

Simplifying Boolean Expressions

- Another (new) SOP Expression:

  $x\bar{y}z + xyz + \bar{x}yz + \bar{x}\bar{y}z$



$\boldsymbol{x\,y} \backslash \boldsymbol{z}$	$\bar{z}$ 0	z 1
$\bar{x}\bar{y}$ 00	0 000	1 001
$\bar{x}y$ 01	0 010	1 011
xy 11	0 110	1 111
$x\bar{y}$ 10	0 100	1 101

Simplifying Boolean Expressions

- Cell grouping of 4

- Result is 1 when z is 1
    - x and y are irrelevant

- Result:

$$x\bar{y}z + xyz + \bar{x}yz + \bar{x}\bar{y}z = \boldsymbol{z}$$

$\boldsymbol{x\ y}$ \ $\boldsymbol{z}$	$\bar{z}$ 0	z 1
$\bar{x}\bar{y}$ 00	0 000	1 001
$\bar{x}y$ 01	0 010	1 011
xy 11	0 110	1 111
$x\bar{y}$ 10	0 100	1 101

Simplifying Boolean Expressions

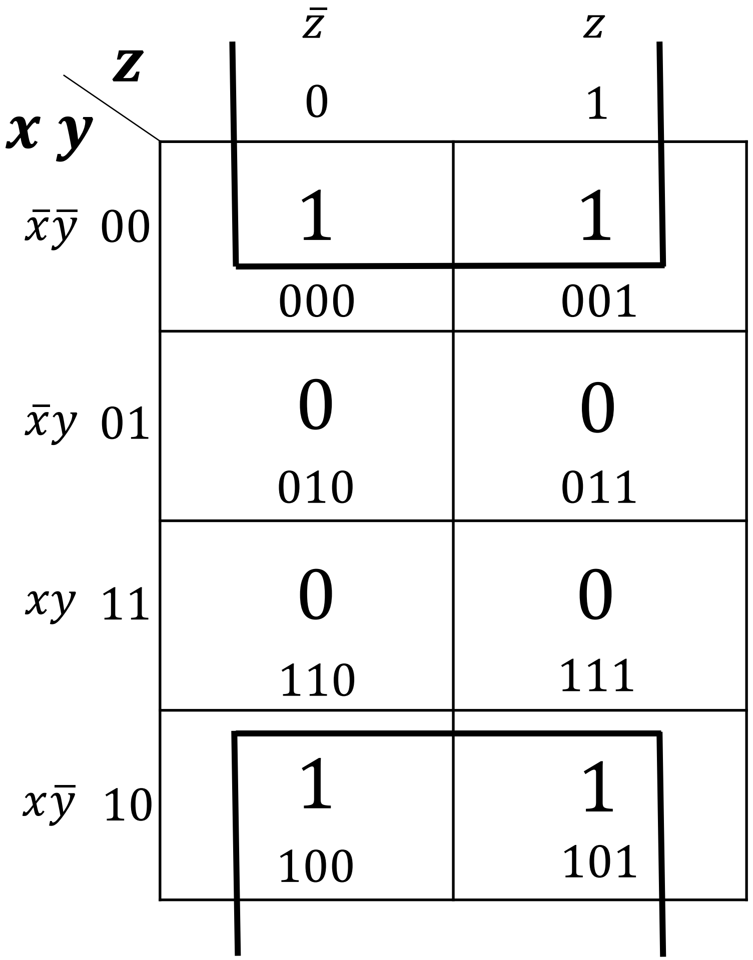
- Another (new) SOP Expression:
  $x\bar{y}\bar{z} + x\bar{y}z + \bar{x}\bar{y}\bar{z} + \bar{x}\bar{y}z$

$\boldsymbol{z}$ $\boldsymbol{x\ y}$	$\bar{z}$ 0	z 1
$\bar{x}\bar{y}$ 00	1 000	1 001
$\bar{x}y$ 01	0 010	0 011
xy 11	0 110	0 111
$x\bar{y}$ 10	1 100	1 101

Simplifying Boolean Expressions

- Cell groups can wrap around the map
- Result is 1 when y is 0
  - x and z are irrelevant
- Result:

$$x\bar{y}\bar{z} + x\bar{y}z + \bar{x}\bar{y}\bar{z} + \bar{x}\bar{y}z = \boldsymbol{\bar{y}}$$



$\boldsymbol{x\ y}$ \ $\boldsymbol{z}$	$\bar{z}$ 0	z 1
$\bar{x}\bar{y}$ 00	1 000	1 001
$\bar{x}y$ 01	0 010	0 011
xy 11	0 110	0 111
$x\bar{y}$ 10	1 100	1 101

Simplifying Boolean Expressions

- The useful of K-Maps for simplifying Boolean expressions should be apparent

- However, K-Maps will not explicitly show:
  - NAND Operations
  - NOR Operations
  - XOR Operations
  - XNOR Operations

Simplifying Boolean Expressions

- No groupings in this K-Map
- Result:

  $\bar{x}\bar{y} + xy$
- Must recognize as XNOR

  $x \odot y$

$\boldsymbol{x}$ \ $\boldsymbol{y}$		$\bar{y}$ 0	y 1
$\bar{x}$	0	1 00	0 01
x	1	0 10	1 11

Simplifying Boolean Expressions

- Another ability of K-Maps are that it can ignore certain outputs.
- These outputs are called **don’t cares**
- If there are certain inputs that we don’t care about, we can still use those inputs in the simplification process

Simplifying Boolean Expressions

- Starting with a truth table…
- F represents the output of this function
- X represents outputs we don’t care about
  - We don’t what the output is for $x\bar{y}\bar{z}$ or $xy\bar{z}$

x	y	z	$\boldsymbol{F}$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	X
1	0	1	0
1	1	0	X
1	1	1	1

Simplifying Boolean Expressions

- Building the K-Map

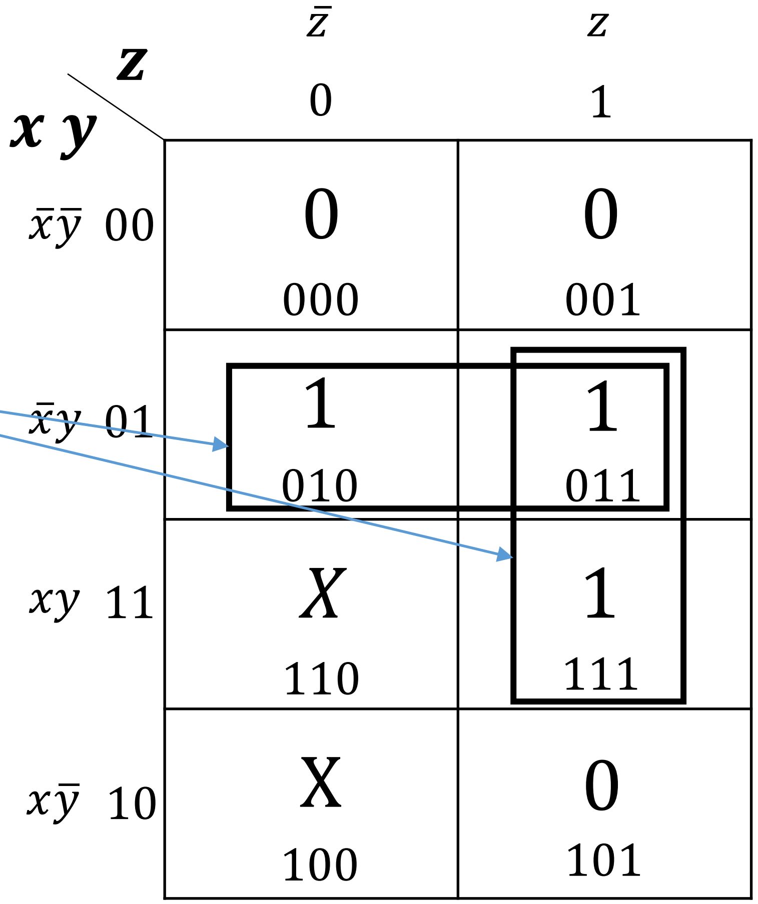
x	y	z	$\boldsymbol{F}$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	X
1	0	1	0
1	1	0	X
1	1	1	1

$\boldsymbol{x\ y}$ \ $\boldsymbol{z}$	$\bar{z}$ 0	z 1
$\bar{x}\bar{y}$ 00	0 000	0 001
$\bar{x}y$ 01	1 010	1 011
xy 11	X 110	1 111
$x\bar{y}$ 10	X 100	0 101

Simplifying Boolean Expressions

- Normally, we'd get the result:

$$\bar{x}y + yz$$

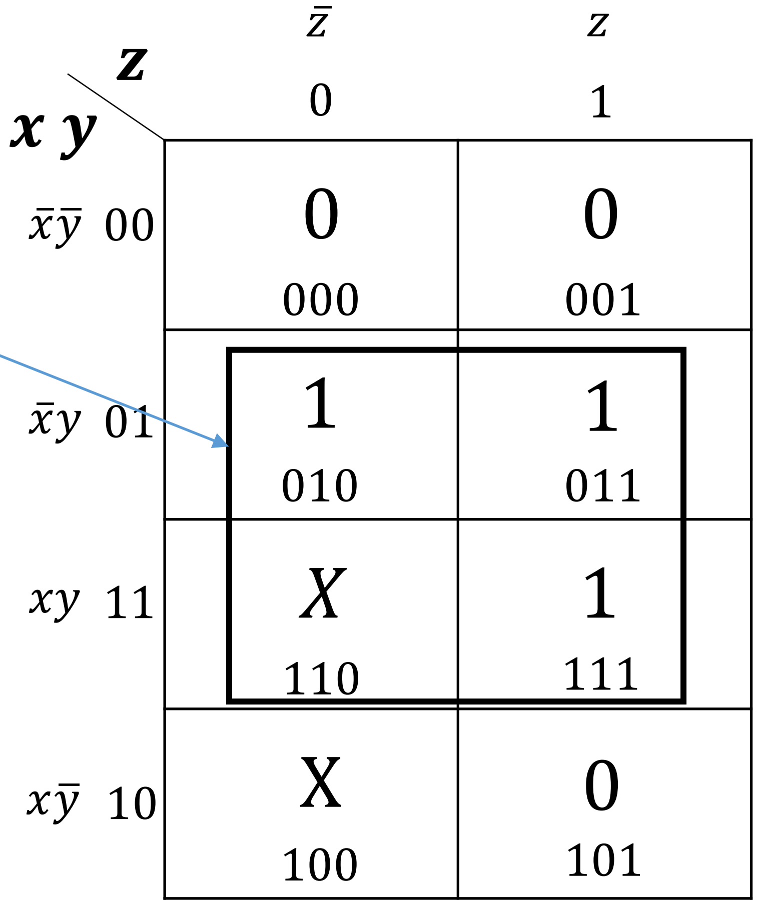


$x\,y$ \ z	$\bar{z}$ 0	z 1
$\bar{x}\bar{y}$ 00	0 000	0 001
$\bar{x}y$ 01	1 010	1 011
xy 11	X 110	1 111
$x\bar{y}$ 10	X 100	0 101

- But, we can use the don't cares as a 1 if it helps simply further

Simplifying Boolean Expressions

- Now, our result is: y



$\boldsymbol{x\,y} \backslash \boldsymbol{z}$	$\bar{z}$ 0	z 1
$\bar{x}\bar{y}$ 00	0 000	0 001
$\bar{x}y$ 01	1 010	1 011
xy 11	X 110	1 111
$x\bar{y}$ 10	X 100	0 101

Simplifying Boolean Expressions

- Checking with the truth table…

- Again, this function *F* doesn’t care about the output for the inputs of $x\bar{y}\bar{z}$ or $xy\bar{z}$

x	y	z	$\boldsymbol{F}$	$\bar{x}y$	yz	$\bar{x}y + yz$
0	**0**	0	**0**	0	0	**0**
0	**0**	1	**0**	0	0	**0**
0	**1**	0	**1**	1	0	**1**
0	**1**	1	**1**	1	1	**1**
~~1~~	~~0~~	~~0~~	~~X~~	~~0~~	~~0~~	~~0~~
1	**0**	1	**0**	0	0	**0**
~~1~~	~~1~~	~~0~~	~~X~~	~~0~~	~~0~~	~~0~~
1	**1**	1	**1**	0	1	**1**

Logic Circuits

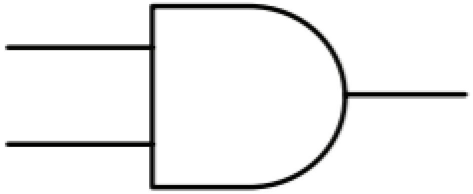
- The Boolean expressions we have been working with are the basis of constructing logic circuits.

- A **logic circuit** is a diagram of a Boolean expression.

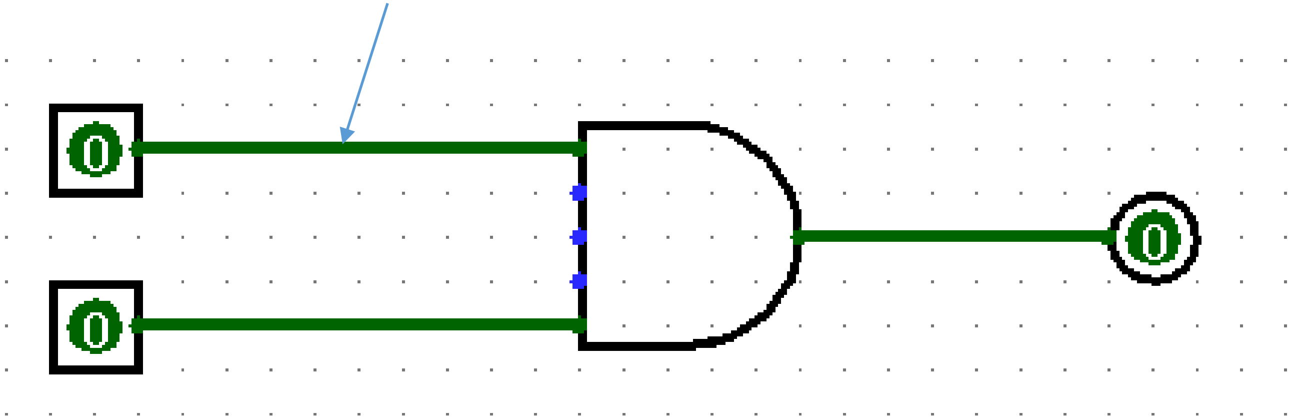
- Logic circuits are built using **logic gates** that perform the different logical operations

Logic Circuits

- The AND Gate

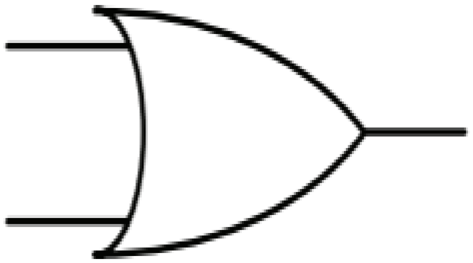


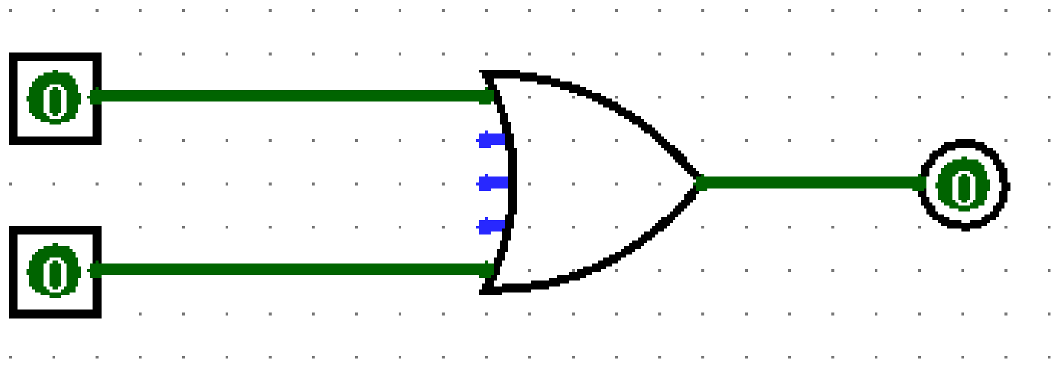
“Wires”



Logic Circuits

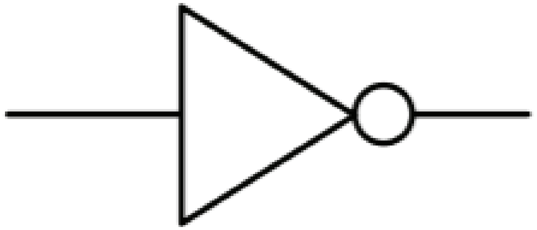
- The OR Gate

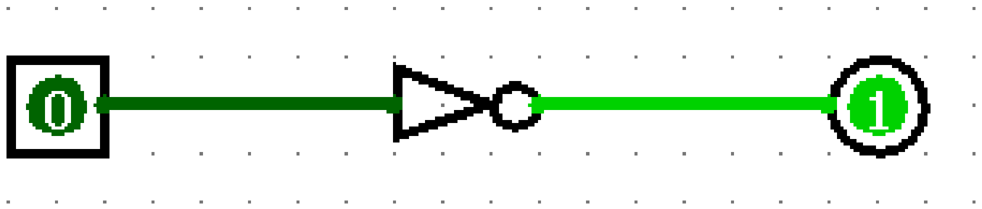




Logic Circuits

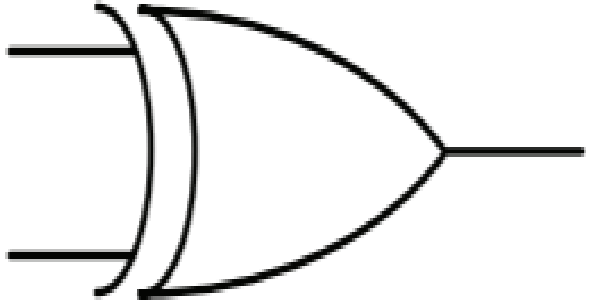
- The NOT Gate

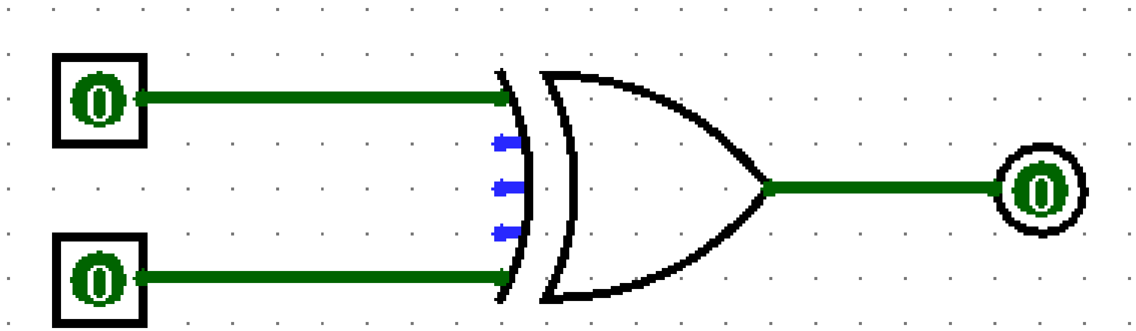




Logic Circuits

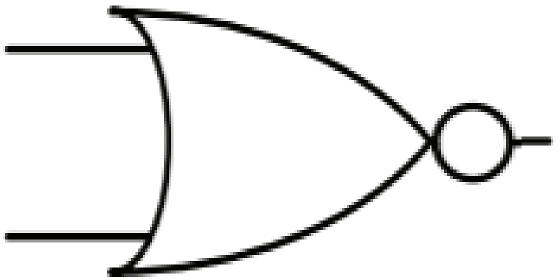
- The XOR Gate

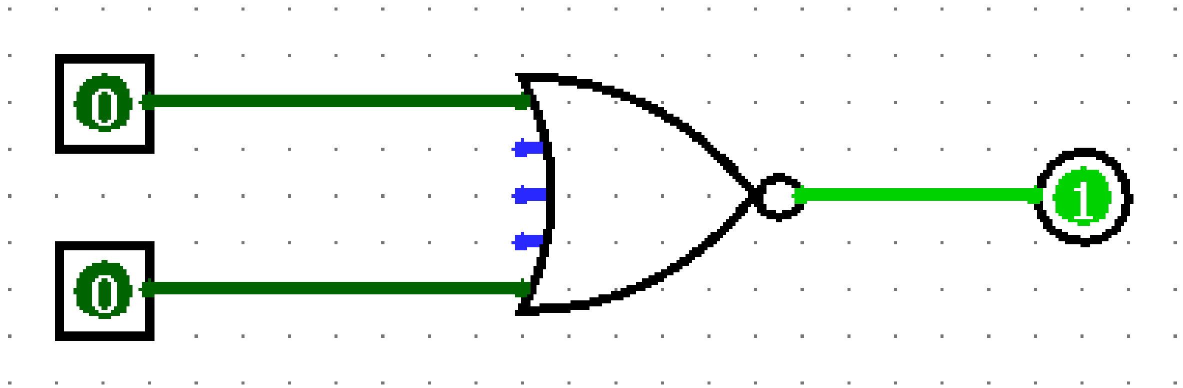




Logic Circuits

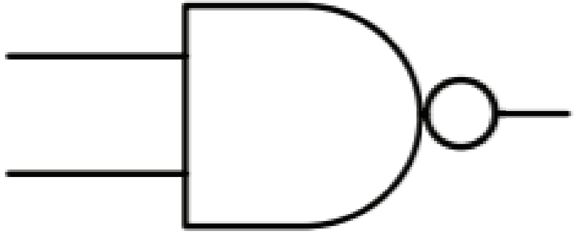
- The NOR Gate

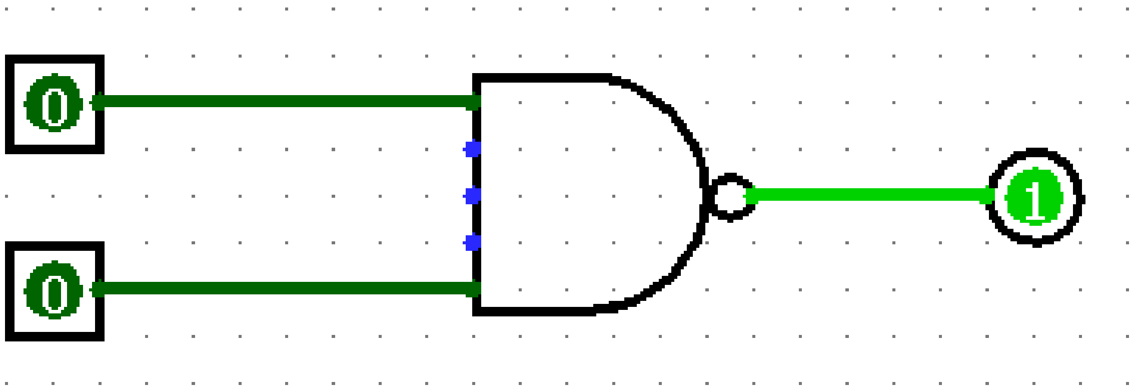




Logic Circuits

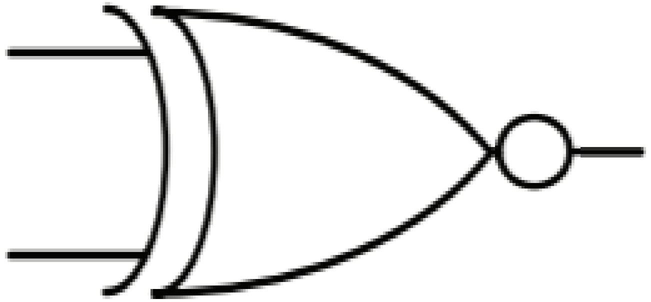
- The NAND Gate

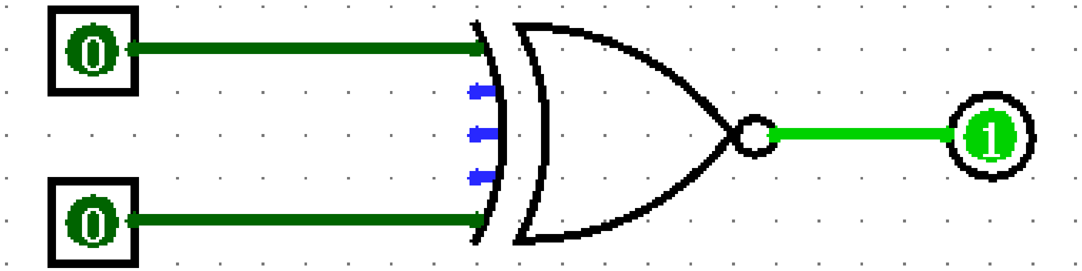




Logic Circuits

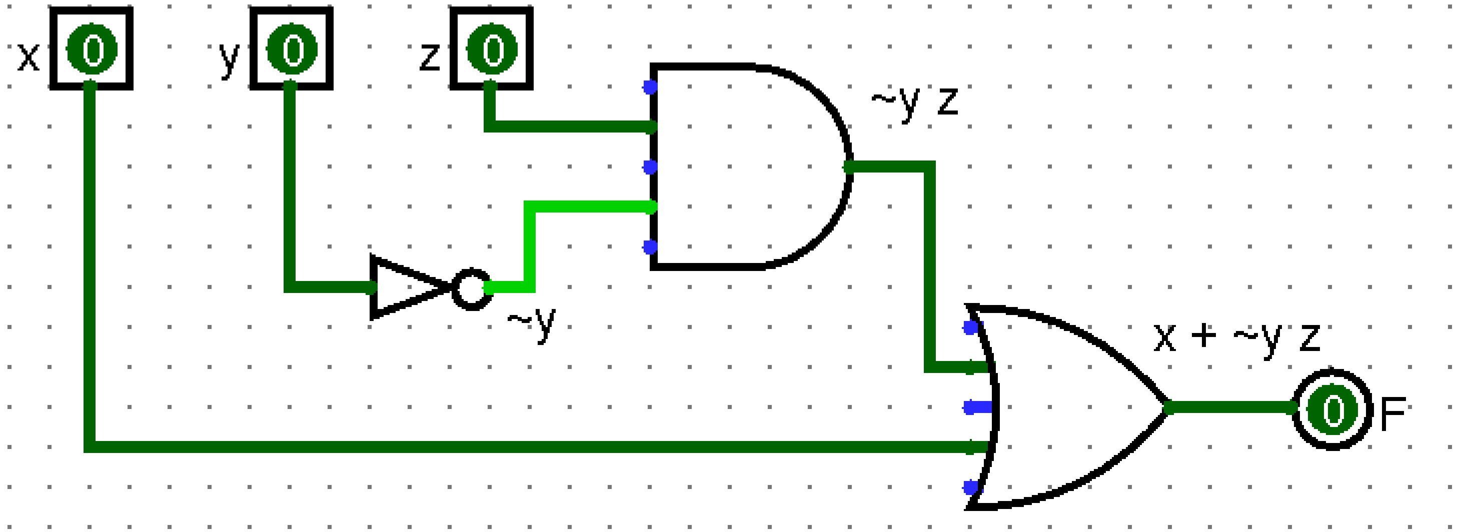
- The XNOR Gate





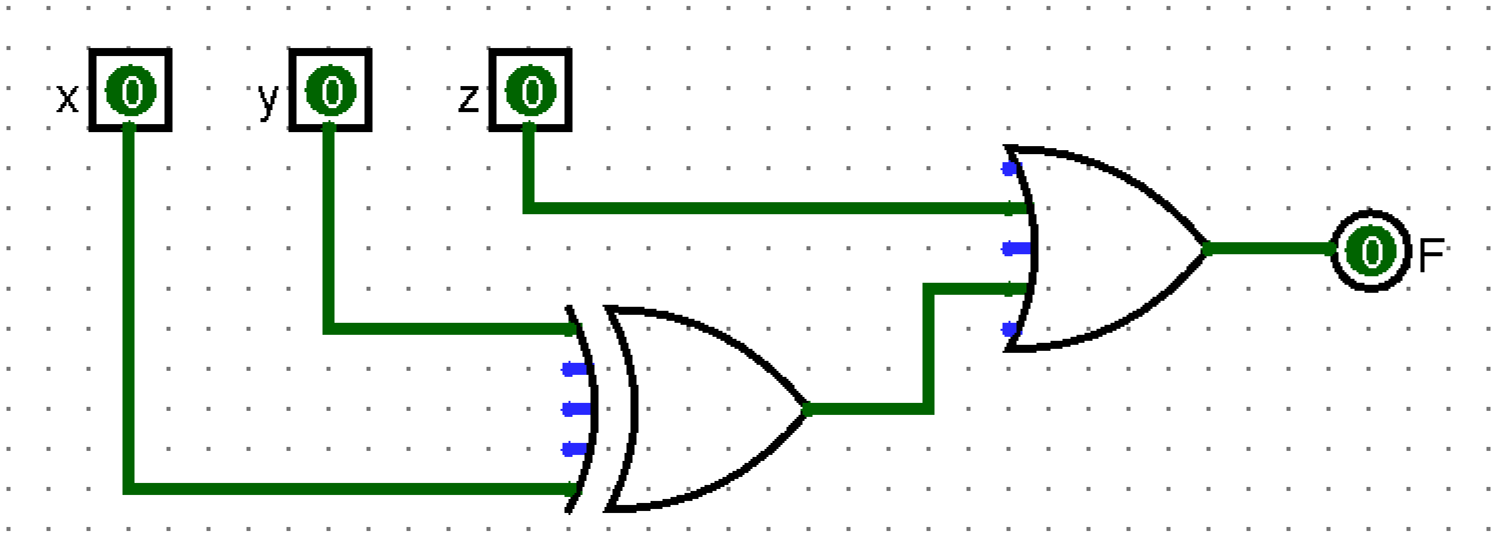
Logic Circuits

- A logic circuit for the Boolean expression $x + \bar{y}z$
  - Be sure to follow order of operations



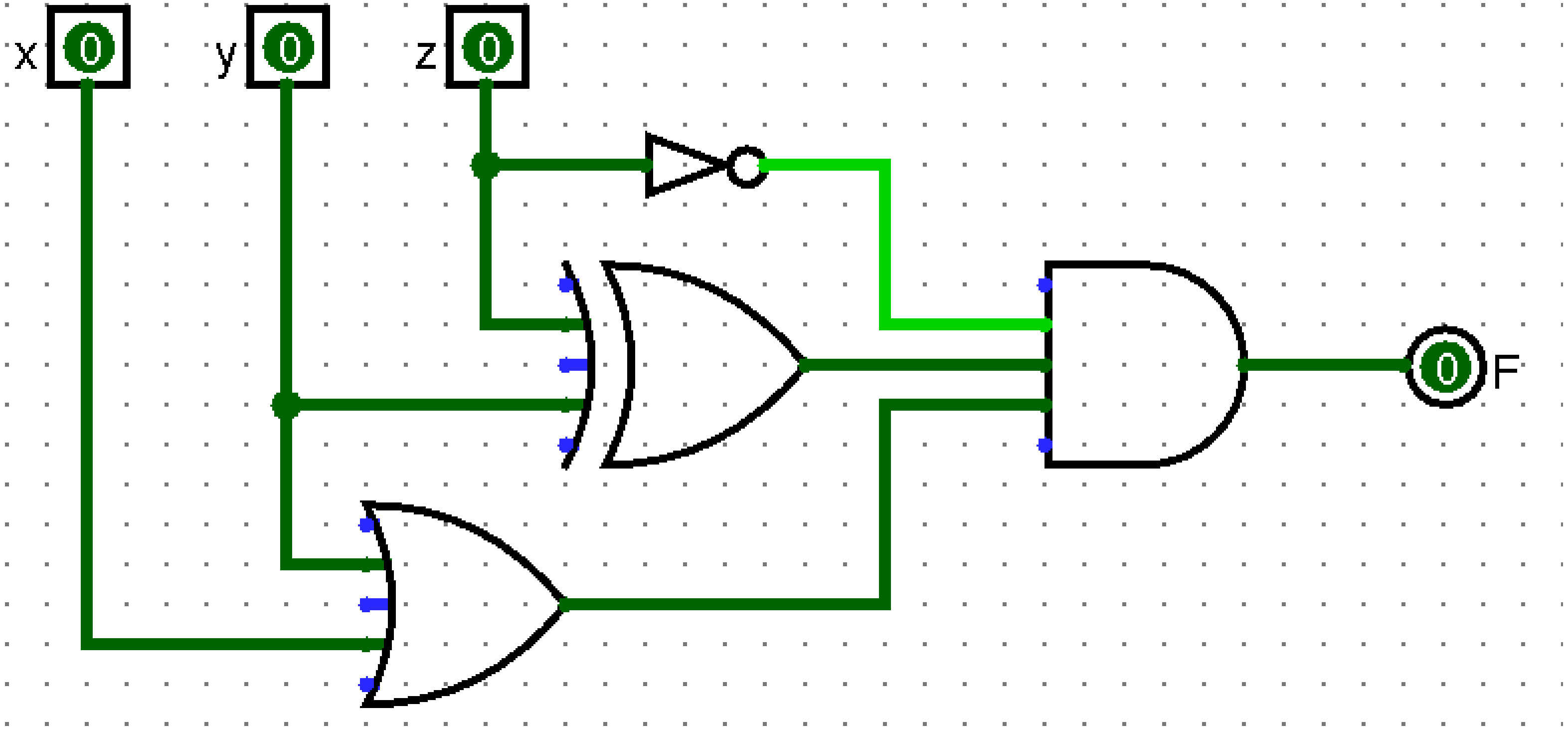
Logic Circuits

- A logic circuit for the Boolean expression $x \oplus y + z$



Logic Circuits

- A logic circuit for the Boolean expression $(x + y)(y \oplus z)\bar{z}$



Buses

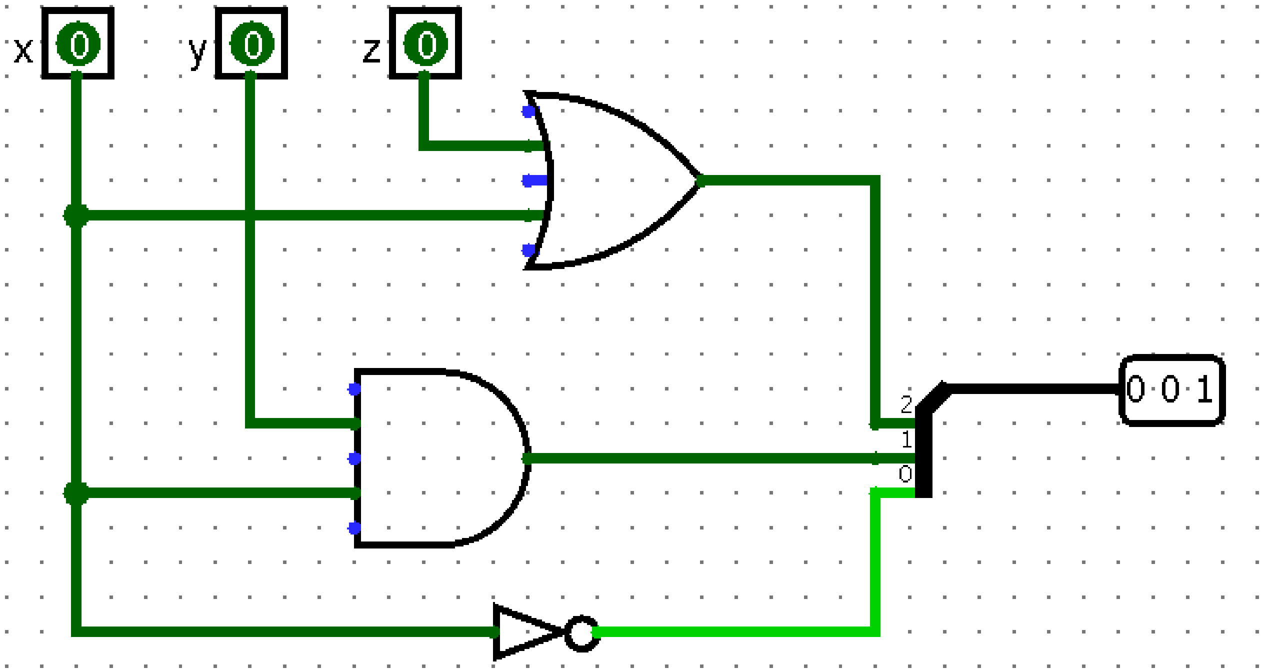
- A **bus** is several parallel wires going from one component to another.

- The bus’s **width** is the number of parallel wires

- In a logic circuit, many parallel wires (like, say in a 32-bit bus) may make the diagram large and confusing

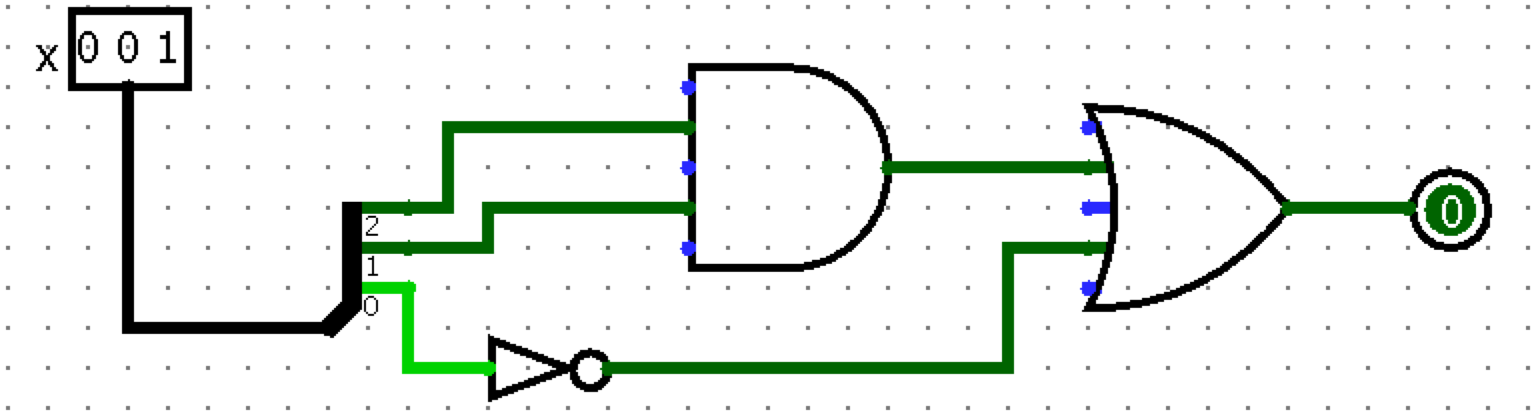
Buses

- This circuit illustrates bundling three wires into one 3-bit bus



Buses

- This circuit illustrates splitting the wires from a 3-bit bus



Sign Extend

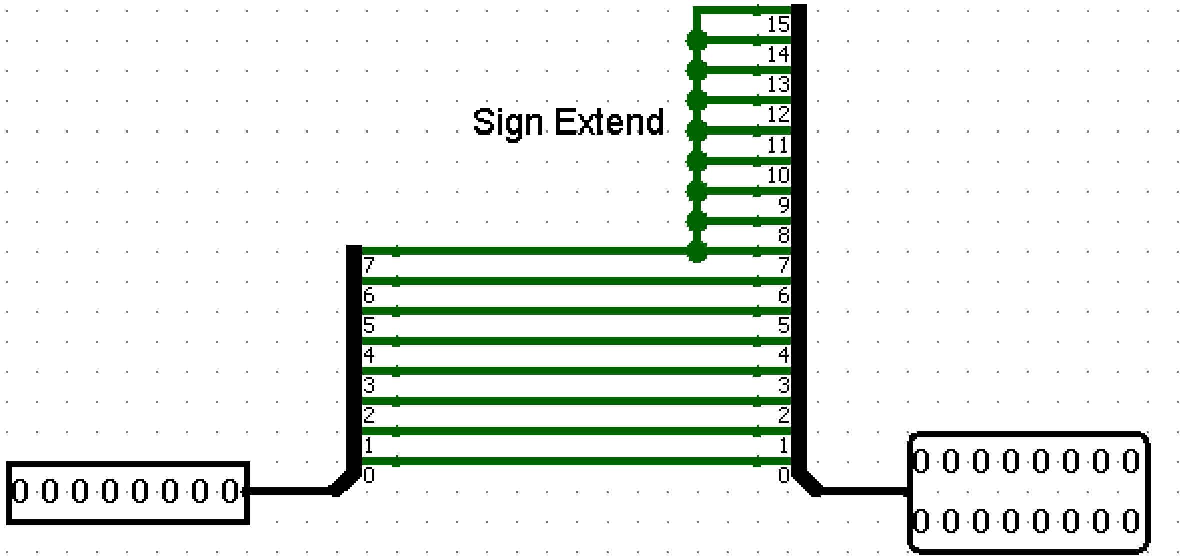
- A sign extend is a combinational logic circuit that preserves the sign bit of a number on a bus when the bus's width is increased.

- For example, we'll say two 8-bit (two's complement) binary numbers 11001100 (negative) and 01010101 (positive) are sent on an 8-bit bus

- The bus's width is extended to 16-bits
  - A sign extend preserves the sign bit when this happens

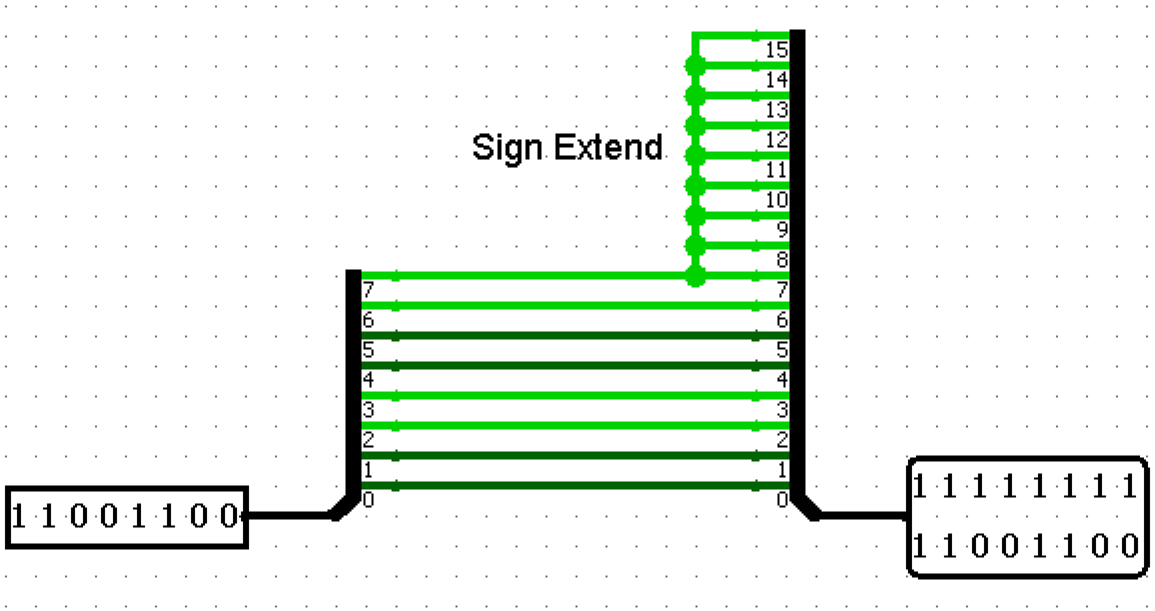
Sign Extend

Input Bus (8-bits)	Output Bus (16-bits)
11001100	**11111111** 11001100
01010101	**00000000** 01010101



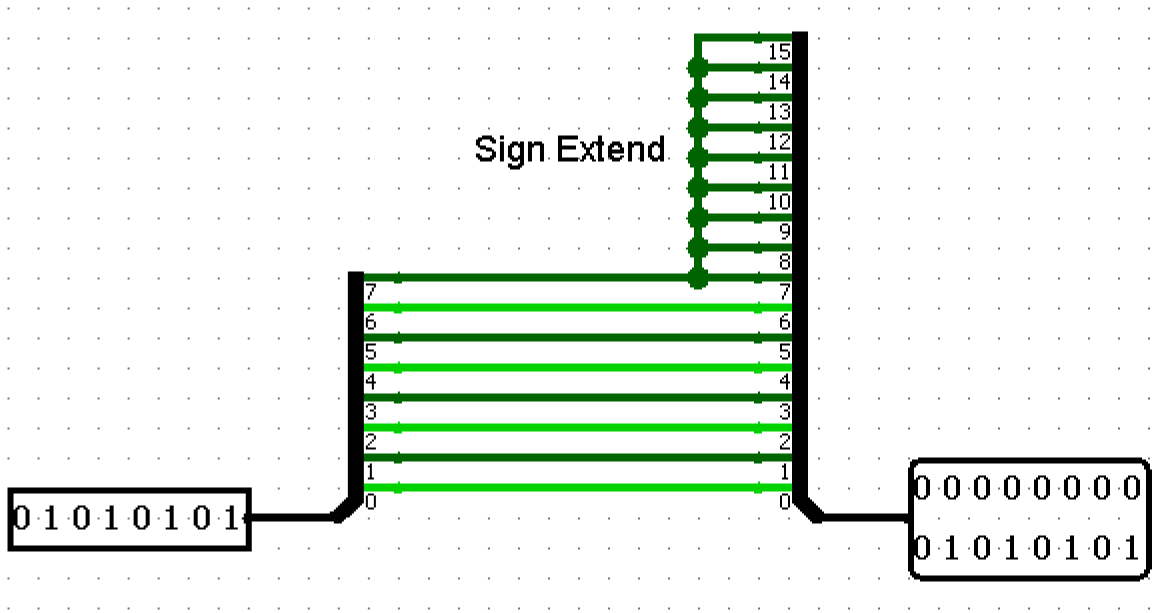
Sign Extend

Input Bus (8-bits)	Output Bus (16-bits)
11001100	**11111111 11001100**
01010101	00000000 01010101



Sign Extend

Input Bus (8-bits)	Output Bus (16-bits)
11001100	11111111 11001100
01010101	**00000000 01010101**



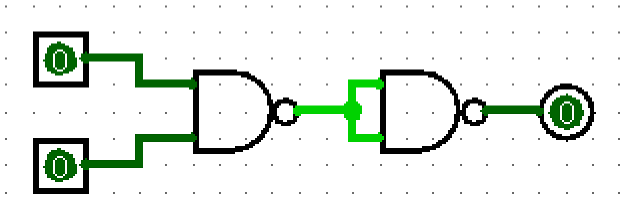
NAND-only/NOR-only Logic Circuits

- All the previously shown gates can be implemented using only NAND gates or implemented using only NOR gates

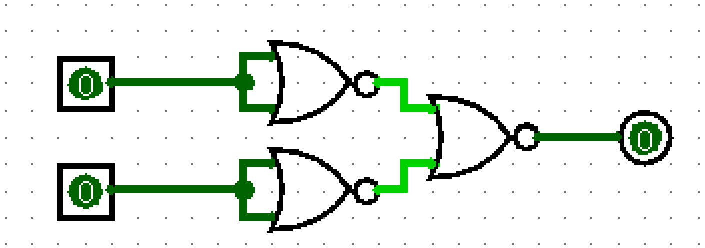
- NAND and NOR gates require fewer transistors than other gates, thus only using these gates will save room

Logic Circuits

- NAND-only AND gate:

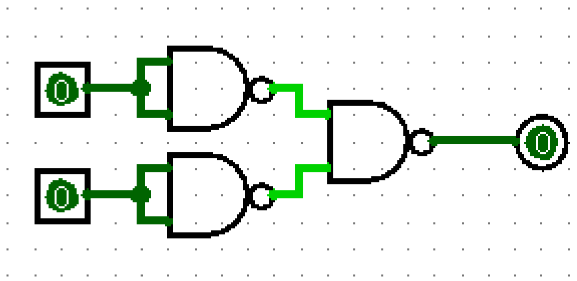


- NOR-only AND gate:

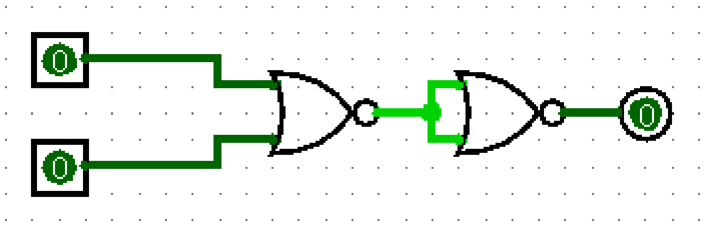


Logic Circuits

- NAND-only OR gate:

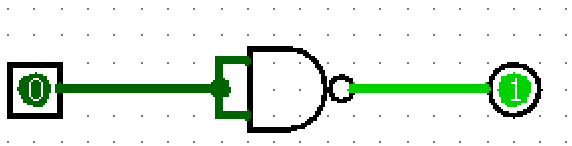


- NOR-only OR gate:

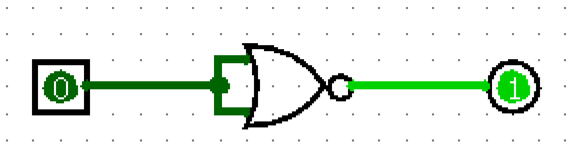


Logic Circuits

- NAND-only NOT gate:

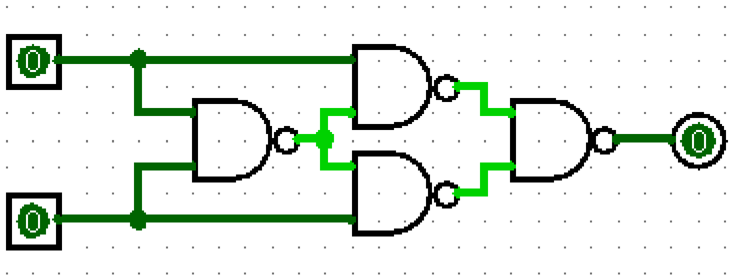


- NOR-only NOT gate:

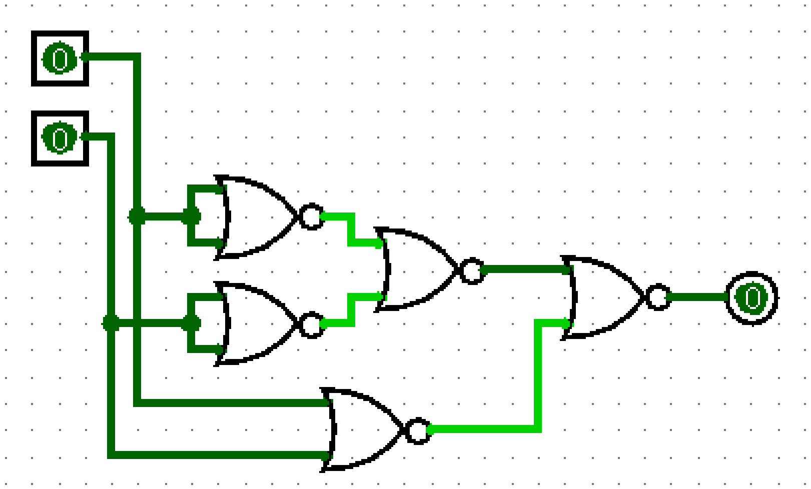


Logic Circuits

- NAND-only XOR gate:

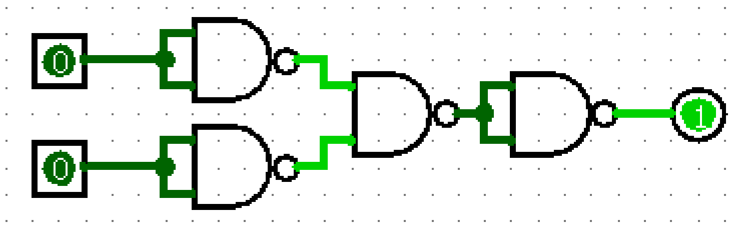


- NOR-only XOR gate:

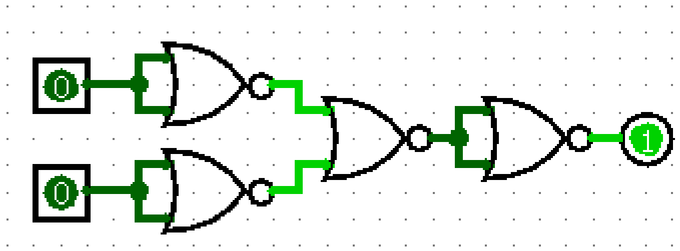


Logic Circuits

- NAND-only NOR gate:

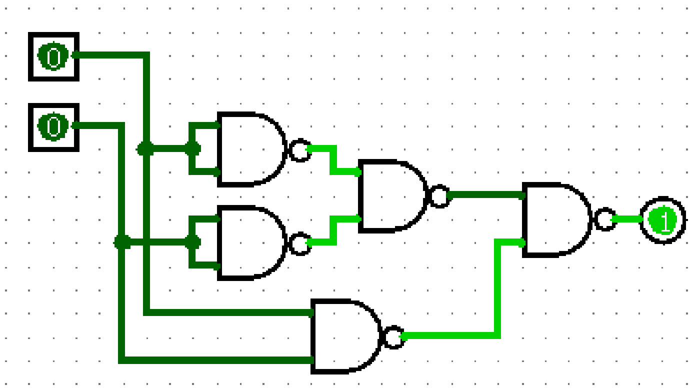


- NOR-only NAND gate:



Logic Circuits

- NAND-only XNOR gate:



- NOR-only XNOR gate: