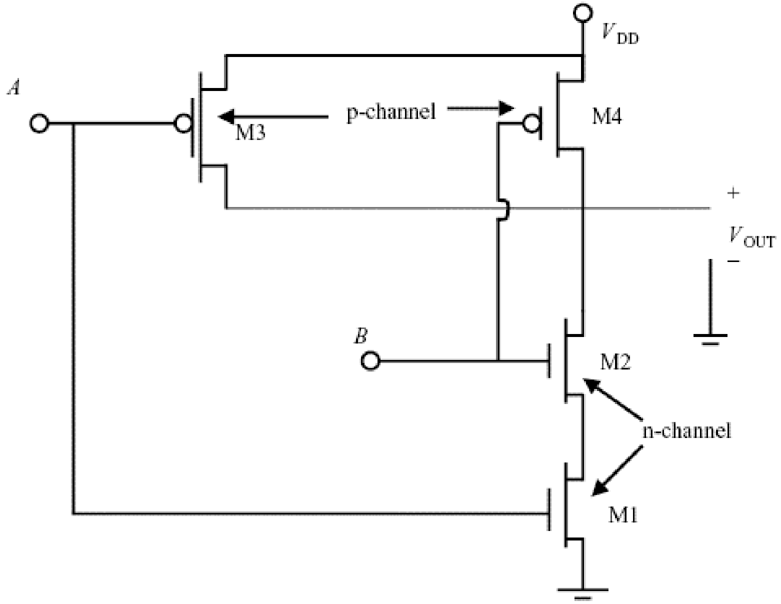
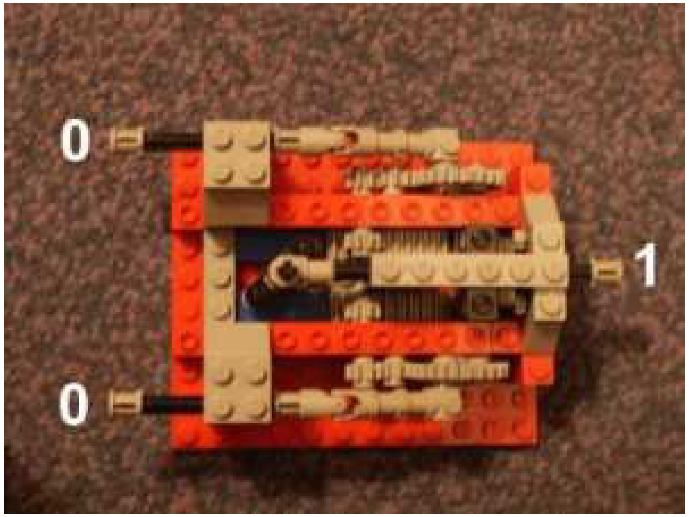
Introduction to Logic Gates

- Using transistor technology, we can create basic logic gates that perform boolean operations on high (5V) and low (0V) signals.
- Example: NAND gate



Intro to Logic Gates

- Digital logic gates form the basis for all implemented computing machines.
- Basic gates:
  - AND : $*$, $\wedge$
  - OR : $+$, $\vee$
  - NOT : $\neg$, $\bar{\ }$
  - NAND
  - NOR
  - XOR : $\oplus$
  - XNOR

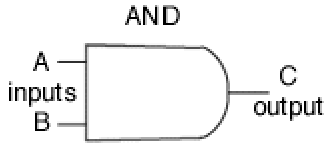


- Relationships between inputs and outputs are outlined in truth tables

Truth Tables

- AND gate

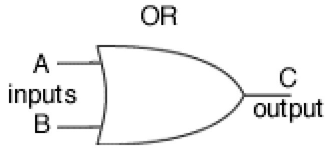
  → C = A · B



A	B	C
0	0	0
1	0	0
0	1	0
1	1	1

- OR gate

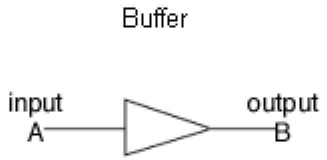
  → C = A+B



A	B	C
0	0	0
1	0	1
0	1	1
1	1	1

Truth Tables (cont’d)

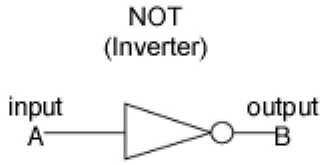
- Buffer



A	B
0	0
1	1

- NOT Gate

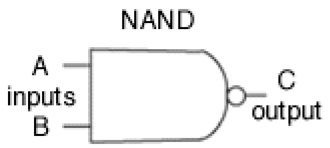
→ $B = \overline{A}$



A	B
0	1
1	0

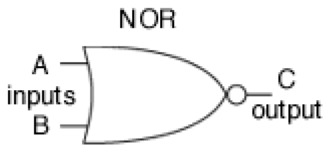
Truth Tables (cont'd)

- NAND Gate



A	B	C
0	0	1
1	0	1
0	1	1
1	1	0

- NOR Gate

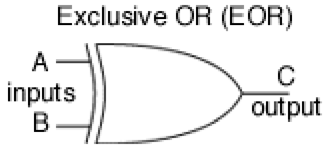


A	B	C
0	0	1
1	0	0
0	1	0
1	1	0

Truth Tables (cont'd)

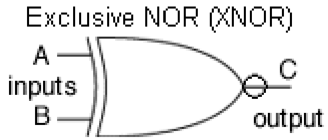
- XOR Gate

  → C = A ⊕ B



A	B	C
0	0	0
1	0	1
0	1	1
1	1	0

- XNOR Gate



A	B	C
0	0	1
1	0	0
0	1	0
1	1	1

Logic Operations

- Logic gates all follow the same rules as logic operators in programming languages.

AND:	$1 \cdot x = x$
	$0 \cdot x = 0$
OR:	$1 + x = 1$
	$0 + x = x$
NOT:	$x + \overline{x} = 1$
	$x \cdot \overline{x} = 0$
XOR:	$x \oplus y = y \oplus x$

- Order of operations also applies to these operations, when putting together the digital circuit.

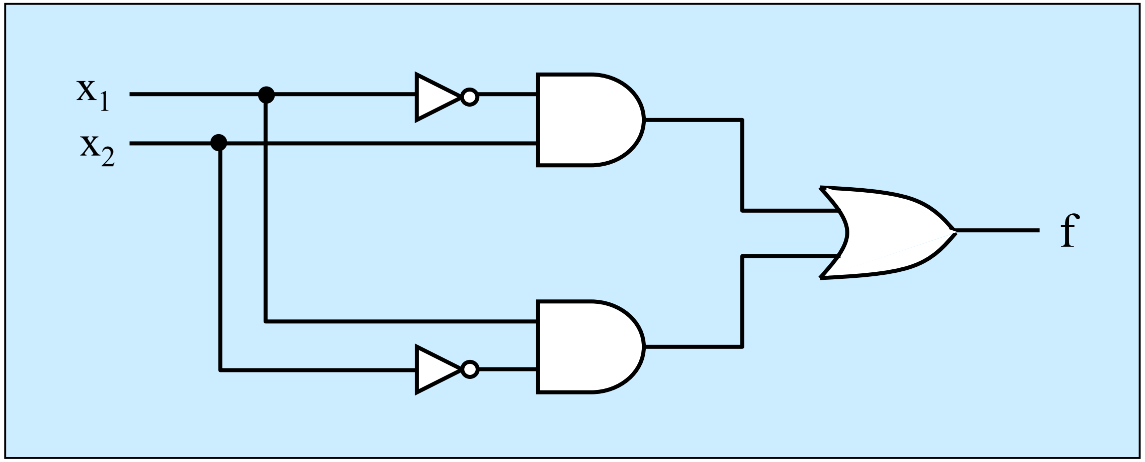
Designing a Circuit

- Task #1: Given a logic expression, determine the equivalent gate representation.
  - Example: $f = \overline{x_1} \cdot x_2 + x_1 \cdot \overline{x_2}$
- Group terms according to order of operations:
  - NOT terms first
  - AND terms next
  - OR terms last

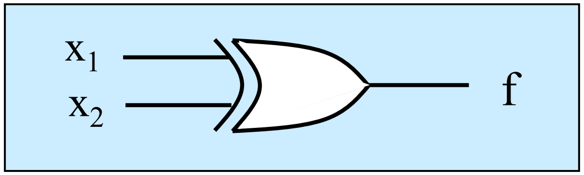
- The example gets rewritten as:

$$f = ((\overline{x_1}) \cdot x_2) + (x_1 \cdot (\overline{x_2}))$$

Circuit Example



- By evaluating the terms of the expression from the inside-out, we construct the diagram above.
- Of course, this can also be represented as a single gate, but the underlying complexity is the same.



More Circuit Design

- Task #2: Given a truth table that specifies a logic circuit's behaviour, design the equivalent circuit.
  - Example: three-input circuit

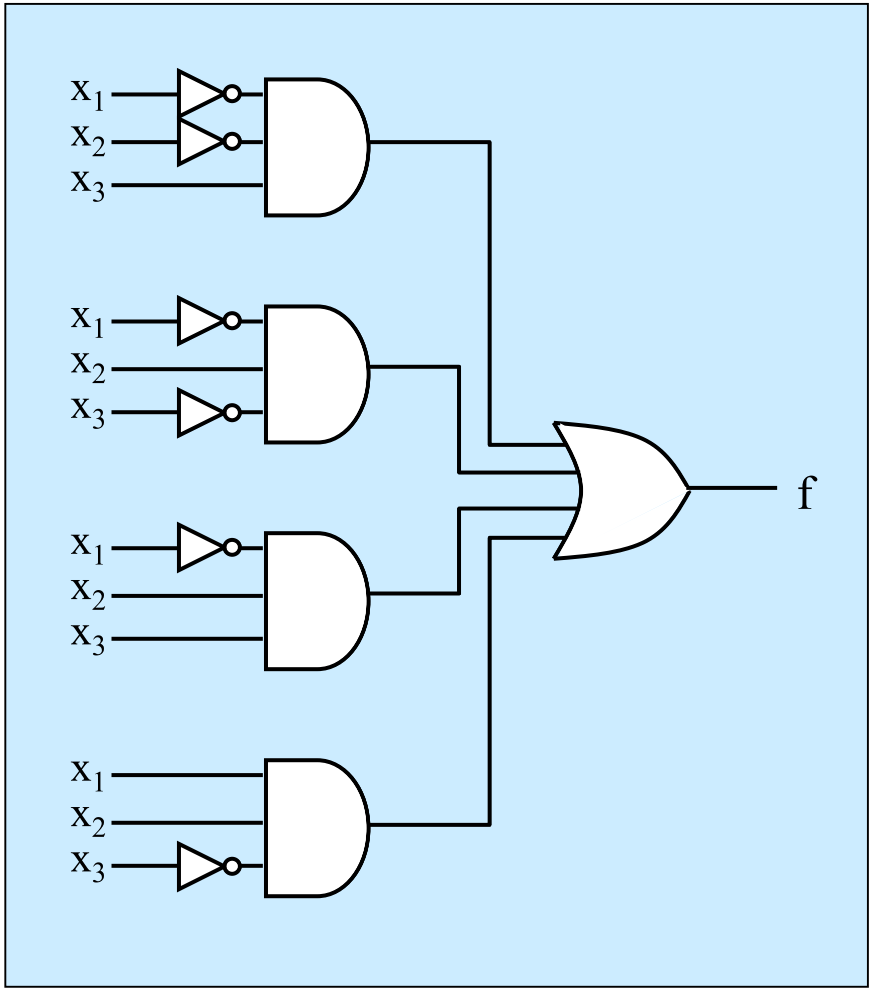
- Sum-of-product technique:
  - Group all rows with an output of f=1 into a single AND term (product)
  - Combine these AND terms with a single OR gate (sum)
- Note: All truth tables can be converted into gate form by using this technique.

x_1	x_2	x_3	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Sum-of-Products

- A minterm is a product term where each of the input variables occurs exactly once.
  - The sum-of-products technique is also referred to as a disjuction of minterms
- Grouping together the minterms for all rows where f=1 yields the following expression:

$$f = \overline{x}_1\overline{x}_2x_3 + \overline{x}_1x_2\overline{x}_3 + \overline{x}_1x_2x_3 + x_1x_2\overline{x}_3$$



Reducing Minterms

$$f = \overline{x}_1\overline{x}_2x_3 + \overline{x}_1x_2\overline{x}_3 + \overline{x}_1x_2x_3 + x_1x_2\overline{x}_3$$

- This is an ugly expression. Can we find some way to minimize the expression, to make it more compact?
- We can, by employing a set of binary logic rules:

Rule Name	Algebraic Identity	
Commutative	$x + y = y + x$	$xy = yx$
Associative	$(x + y) + z = x + (y + z)$	$(xy)z = x(yz)$
Distributive	$x + yz = (x + y)(x + z)$	$x(y + z) = xy + xz$
Idempotent	$x + x = x$	$xx = x$
Involution	$\overline{\overline{x}} = x$	
Complement	$x + \overline{x} = 1$	$x\overline{x} = 0$
de Morgan	$\overline{x + y} = \overline{x} \cdot \overline{y}$	$\overline{xy} = \overline{x} + \overline{y}$

Simplification Example

- Reduce the following sum of products:

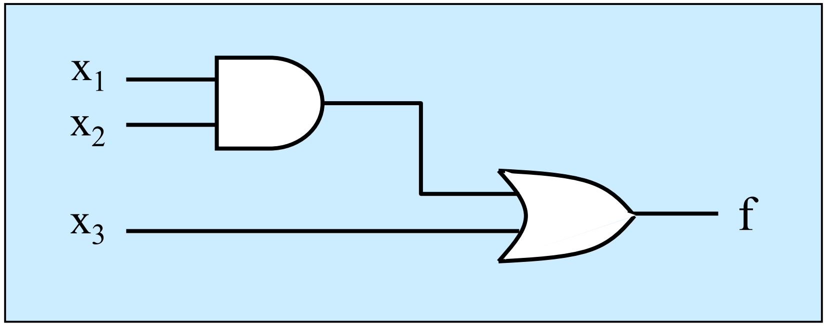
$$f = \overline{x}_1\overline{x}_2x_3 + \overline{x}_1x_2x_3 + x_1\overline{x}_2x_3 + x_1x_2\overline{x}_3 + x_1x_2x_3$$

- Steps:
  - Distributive: $f = \overline{x}_1x_3(\overline{x}_2 + x_2) + x_1\overline{x}_2x_3 + x_1x_2\overline{x}_3 + x_1x_2x_3$
  - Complement: $f = \overline{x}_1x_3 \bullet 1 + x_1\overline{x}_2x_3 + x_1x_2\overline{x}_3 + x_1x_2x_3$
  - Identity: $f = \overline{x}_1x_3 + x_1\overline{x}_2x_3 + x_1x_2\overline{x}_3 + x_1x_2x_3$
  - Idempotent: $f = \overline{x}_1x_3 + x_1\overline{x}_2x_3 + x_1x_2\overline{x}_3 + (x_1x_2x_3 + x_1x_2x_3)$
  - Associative: $f = \overline{x}_1x_3 + (x_1\overline{x}_2x_3 + x_1x_2x_3) + (x_1x_2\overline{x}_3 + x_1x_2x_3)$
  - Distributive: $f = \overline{x}_1x_3 + x_1x_3(\overline{x}_2 + x_2) + x_1x_2(\overline{x}_3 + x_3)$
  - Complement: $f = \overline{x}_1x_3 + x_1x_3 \bullet 1 + x_1x_2 \bullet 1$
  - Identity: $f = \overline{x}_1x_3 + x_1x_3 + x_1x_2$

Simplification Example

- Steps (cont'd):
  - (from previous slide): $f = \overline{x}_1 x_3 + x_1 x_3 + x_1 x_2$
  - Distributive: $f = x_3(\overline{x}_1 + x_1) + x_1 x_2$
  - Complement: $f = x_3 \bullet 1 + x_1 x_2$
  - Identity: $f = x_3 + x_1 x_2$

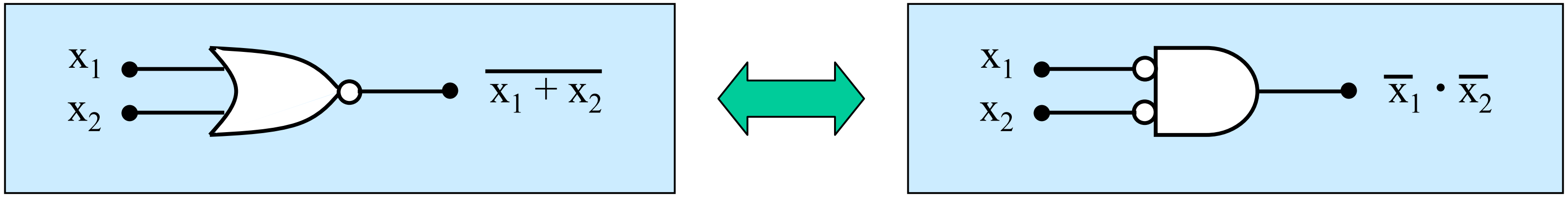
- Solution is: $\boxed{f = x_3 + x_1 x_2}$



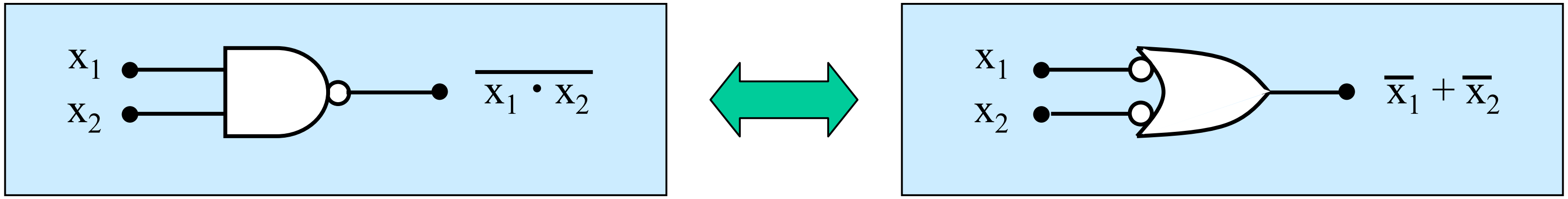
More de Morgan

- Implications of de Morgan's Law:

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



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

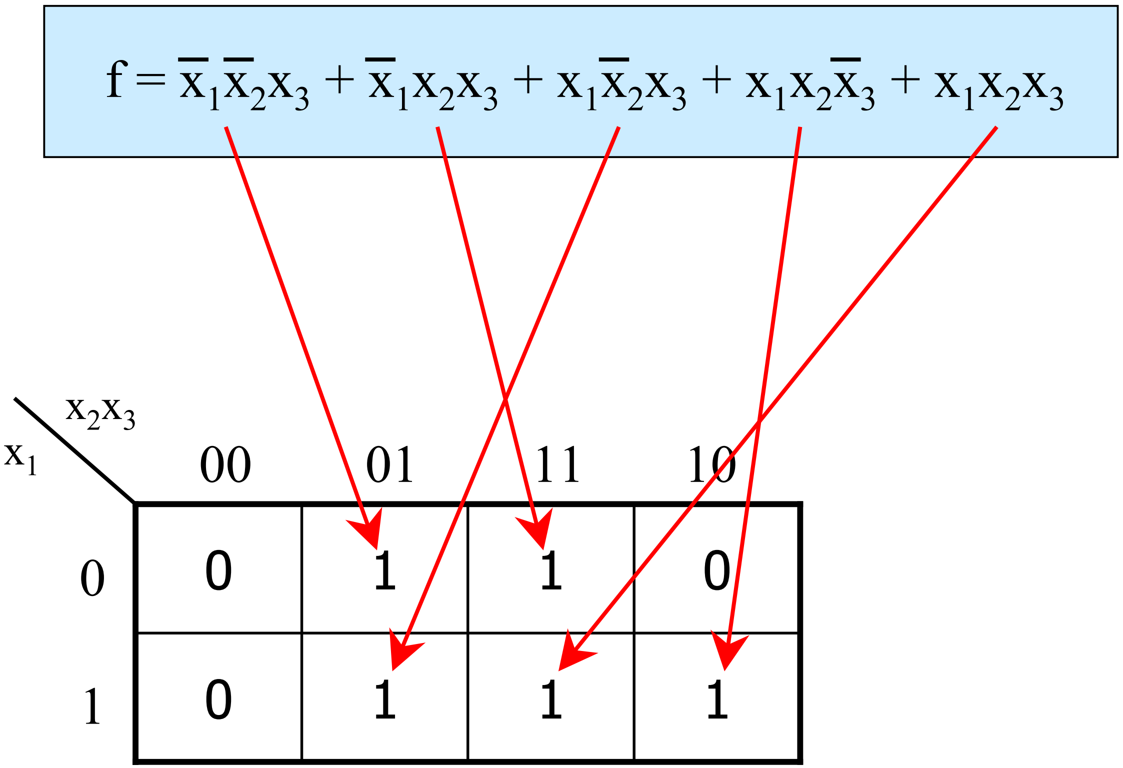


Karnaugh Maps

- A Karnaugh map is another way of representing a circuit’s truth table
  - Presented as a two-dimensional grid of 2^n squares.
  - Each axis represents the different input values to the circuit, and the contents of each square represents the output value for the intersecting input values.
    - Horizontally and vertically adjacent squares only differ by a single input variable
    - Number of rows and columns must be a power of 2
    - Note: row and column labels must also differ by a single digit.
  - Simplified circuit is found by circling groups of adjacent 1’s on the grid.
  - Result: The minimal expression for the circuit.

Karnaugh Maps

- Example:

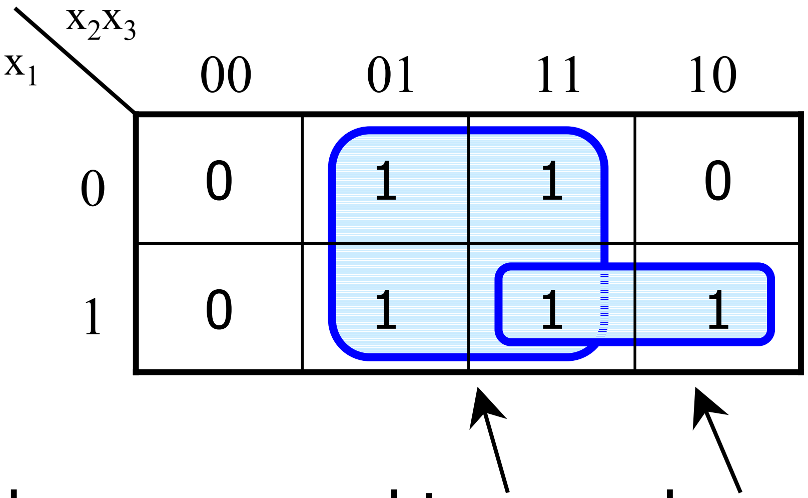


$$f = \overline{x}_1\overline{x}_2x_3 + \overline{x}_1x_2x_3 + x_1\overline{x}_2x_3 + x_1x_2\overline{x}_3 + x_1x_2x_3$$

x_1 \ x_2x_3	00	01	11	10
0	0	1	1	0
1	0	1	1	1

Karnaugh Maps

- Next step: circle blocks of 1's
  - Cannot contain any 0's in the block
  - Height and width of block must be a power of 2
  - Blocks are allowed to overlap

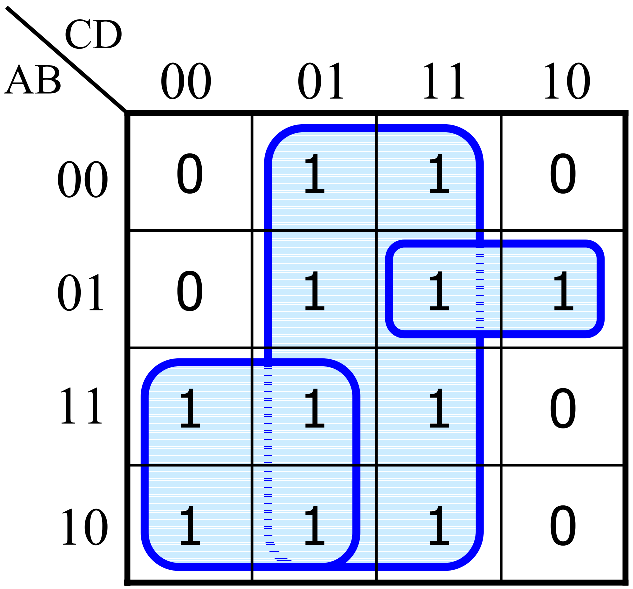


x_1 \ x_2x_3	00	01	11	10
0	0	1	1	0
1	0	1	1	1

- Circled blocks correspond to x_3 and x_1x_2

Karnaugh Example

- Task: Given the truth table on the right, determine the simplest equivalent gate arrangement.



AB \ CD	00	01	11	10
00	0	1	1	0
01	0	1	1	1
11	1	1	1	0
10	1	1	1	0

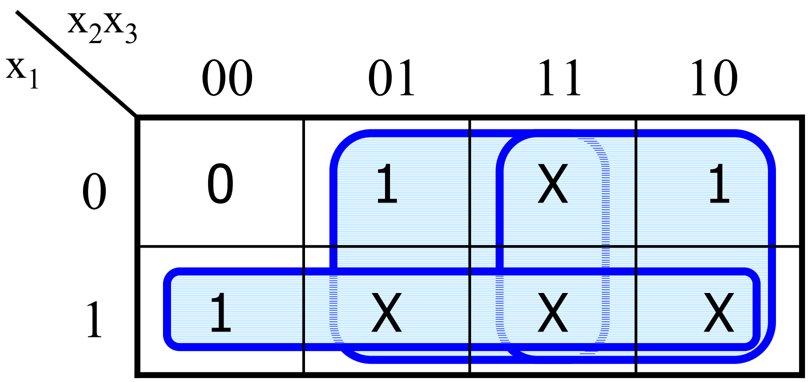
$$X = A\overline{C} + D + \overline{A}BC$$

Inputs				Output
A	B	C	D	X
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

"Don't Care" Conditions

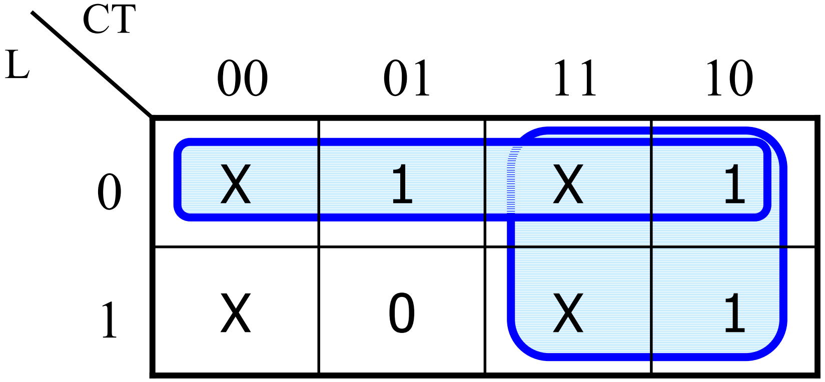
- Sometimes certain outputs are not defined
  - Example: A three-input circuit where the output is 1 if any of the inputs is 1, and 0 if all of the inputs are 0.
  - The output can be anything in the case where two or more inputs are 1, since we don't consider those cases.
  - In those cases, label the output as "don't care" (or "X").

- "Don't care" conditions are useful because they can be treated as either 1 or 0, depending on which makes things easier.
  - In real life, you usually have to set it to some sensible value.



“Don’t Care” Example

- Adding milk to beverages:
  - If a patron orders coffee (C), add milk. If the patron orders tea (T), only add milk if lemon (L) hasn’t already been added.

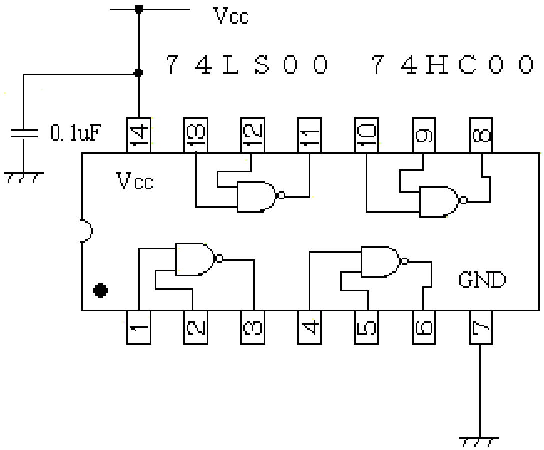


L \ CT	00	01	11	10
0	X	1	X	1
1	X	0	X	1

  - The “don’t care” conditions are sometimes written as “d” (as in the textbook, for example). In industry though, “X” is more commonly used, as in the phrase “X-propagation”.

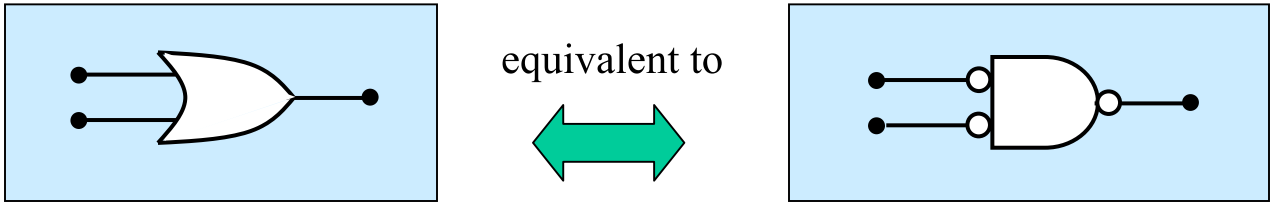
The Importance of NAND

- NAND gates are considered to be the “universal” gate, because any other gate can be synthesized using NAND.
- In fact, most gates are implemented in solid-state TTL chips (Transistor-Transistor Logic)
  - e.g. 74LS00 integrated circuit (IC)

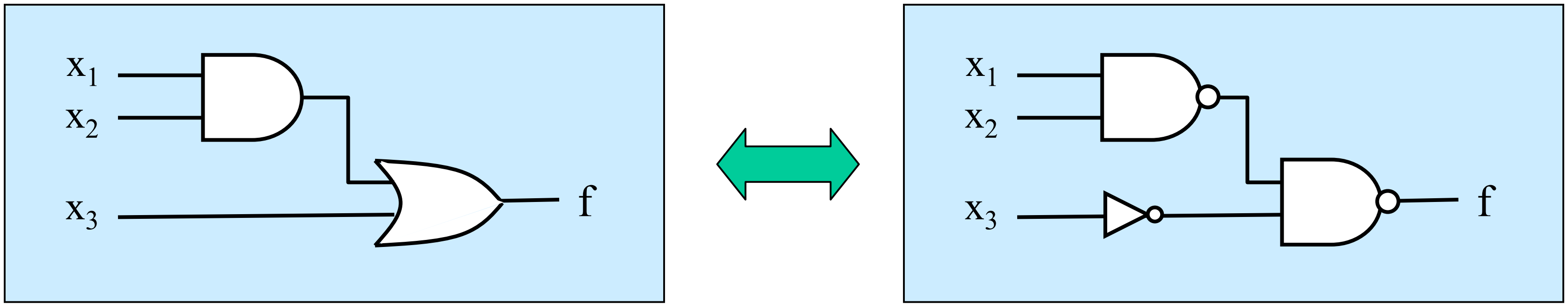


Getting to NAND

- How do we create a NAND-based circuit, if our approach so far has been to find sum-of-products?
  - Answer: By using de Morgan's rule!
- According to de Morgan (see slide 15),



- Therefore, the following are equivalent as well:



Neither...NOR

- NAND and NOR gates are the cheapest to make, and the most commonly found in IC chips
- Typical process for creating a NAND-based circuit is:
  1. represent truth table in Karnaugh map
  2. isolate smallest terms that produce high output
  3. produce sum-of-product model with AND & OR gates
  4. convert model to NAND representation
- What about creating an equivalent circuit out of NOR gates? Is that possible?
  – Requires obtaining maxterms of the truth table, and producing a product-of-sums model

Maxterms

- A maxterm is a sum of input values where every input value occurs exactly once.
- When expressing a truth table, each maxterm represents a row where the output is 0, such that the maxterm will give a true value in all cases except for that row.
  - Maxterms for example table:
    - $(x_1 + x_2 + x_3)$
    - $(\bar{x}_1 + x_2 + x_3)$
    - $(\bar{x}_1 + x_2 + \bar{x}_3)$
    - $(\bar{x}_1 + \bar{x}_2 + \bar{x}_3)$

x_1	x_2	x_3	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

POS (Product of Sums)

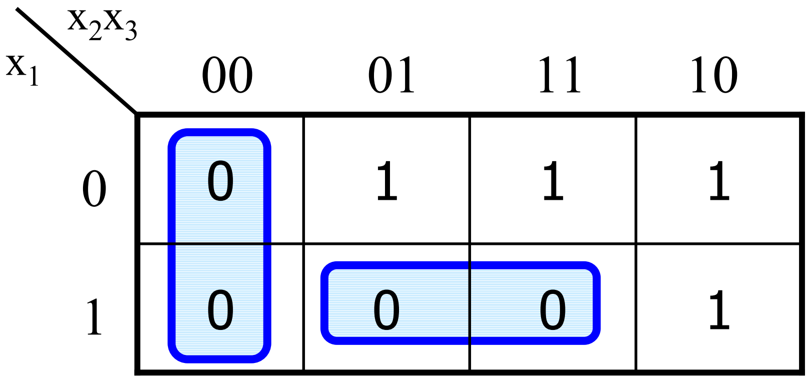
- Instead of creating a disjunction of cases where the output of the circuit is 1, the product-of-sums technique creates a conjunction of the cases where the output is 0.
  - From previous example:

$$(x_1 + x_2 + x_3) \cdot (\bar{x}_1 + x_2 + x_3) \cdot (\bar{x}_1 + x_2 + \bar{x}_3) \cdot (\bar{x}_1 + \bar{x}_2 + \bar{x}_3)$$

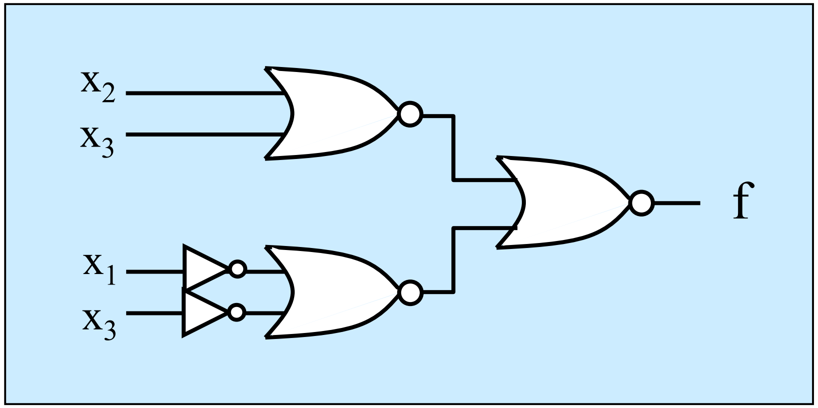
- These equations can be reduced using the same techniques used on minterms.
  - boolean logic rules
  - Karnaugh maps

NOR Circuit

- A product-of-sum circuit will result in several OR gates, united by a single AND gate.
- Through de Morgan's rule, this converts to a circuit made up entirely of NOR gates.
- Design decision: of the maxterm and minterm representations, which has the fewest reduced terms?

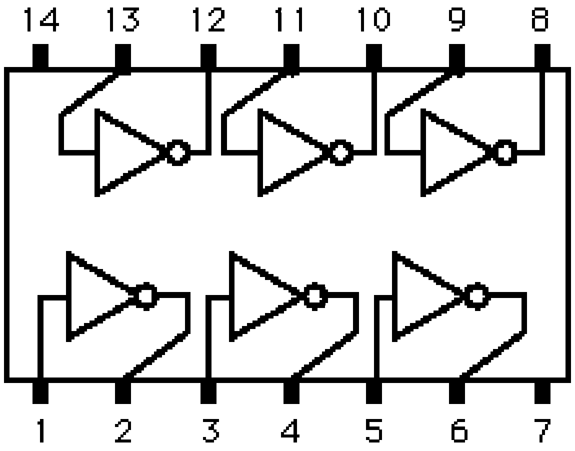


x_1 \ x_2x_3	00	01	11	10
0	0	1	1	1
1	0	0	0	1



Other TTL Logic

- In addition to NAND circuits, inverter circuits are commonly used as well (e.g. 74LS04).



- When using these chips, make sure you have the pins and the orientation correct. For example, applying the voltage source to the ground pin can have some very unpleasant results.