MODULE 3: Boolean Algebra and Digital Logic

Lecture 3.3
Logic Gates

Prepared By:

- Scott F. Midkiff, PhD
- Luiz A. DaSilva, PhD
- Kendall E. Giles, PhD

Electrical and Computer Engineering

Virginia Tech

VirginiaTech
Invent the Future®

Lecture 3.3 Objectives

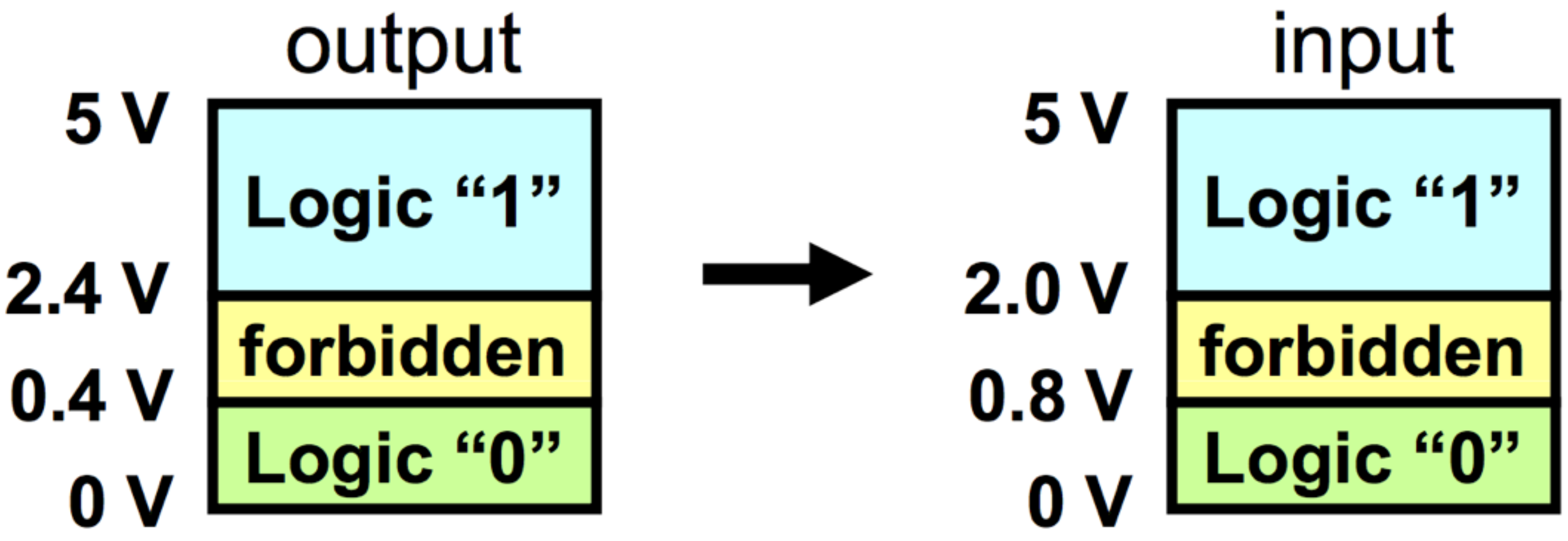
- Describe the relationship between physical voltage values and logic values of "0" and "1"
- Define the difference between combinational and sequential logic
- Draw the gate symbols associated with the following Boolean operations: AND, OR, NOT, NAND, NOR, XOR, and XNOR
- Derive a gate implementation from an algebraic expression

Physical Realization of Logic Values (1)

- Binary logic values take one of two discrete values
    - “0”(or “LOW” or “FALSE”)
    - “1”(or “HIGH” or “TRUE”)
- While “0” and “1” (or “LOW” and “HIGH” or “TRUE” and “FALSE”) are abstractions, hardware requires physical realization of these values
- The physical world is analog – voltages take on an infinite number of possible values over some continuous range

Physical Realization of Logic Values (2)

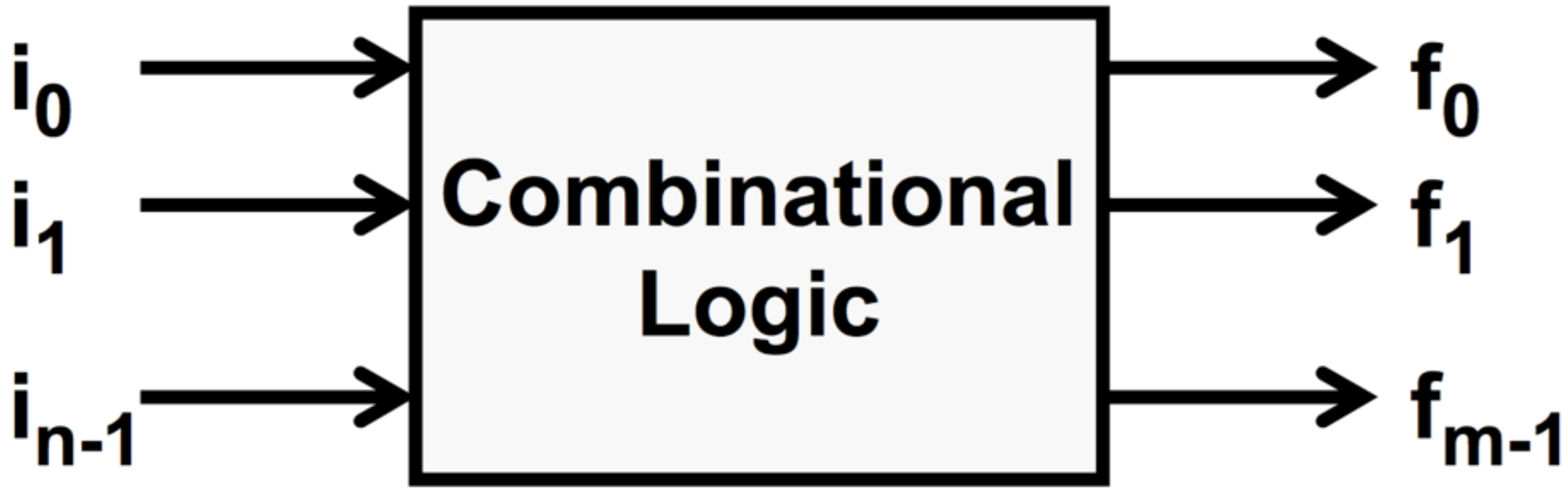
- Digital signals are based on assigned thresholds
- Thresholds to generate logic values are more constrained than thresholds to interpret logic values to avoid ambiguity or errors



(Example thresholds for a common logic family)

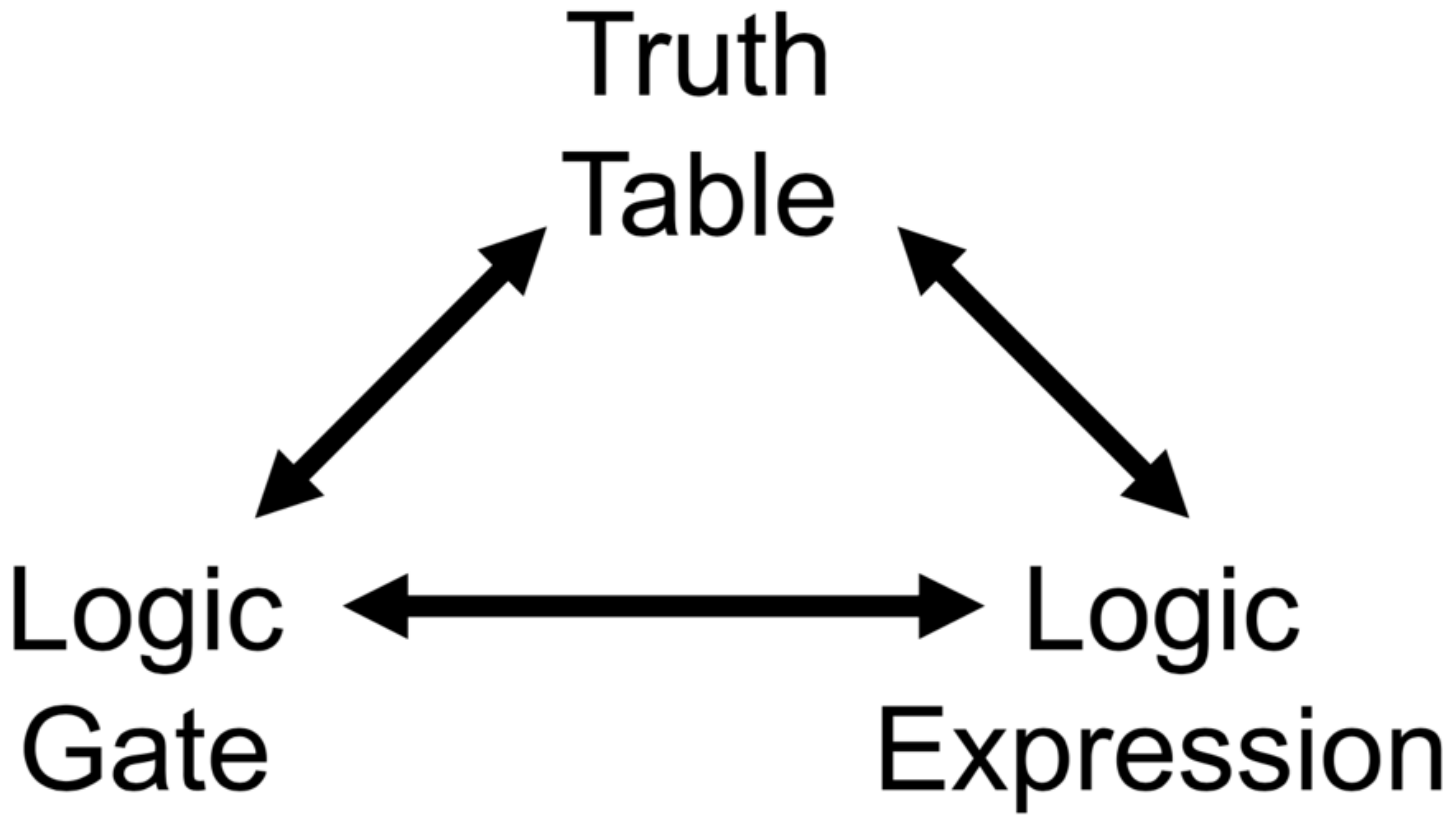
Combinational and Sequential Logic

- Consider a logic unit with inputs and outputs (“0” or “1”)
- Outputs of combinational logic depend only on the present values of the inputs
  - The combination of input values completely determines the output values
- Outputs of a sequential logic unit depend on both present input values and past input values, i.e. there is memory



Logic Gates

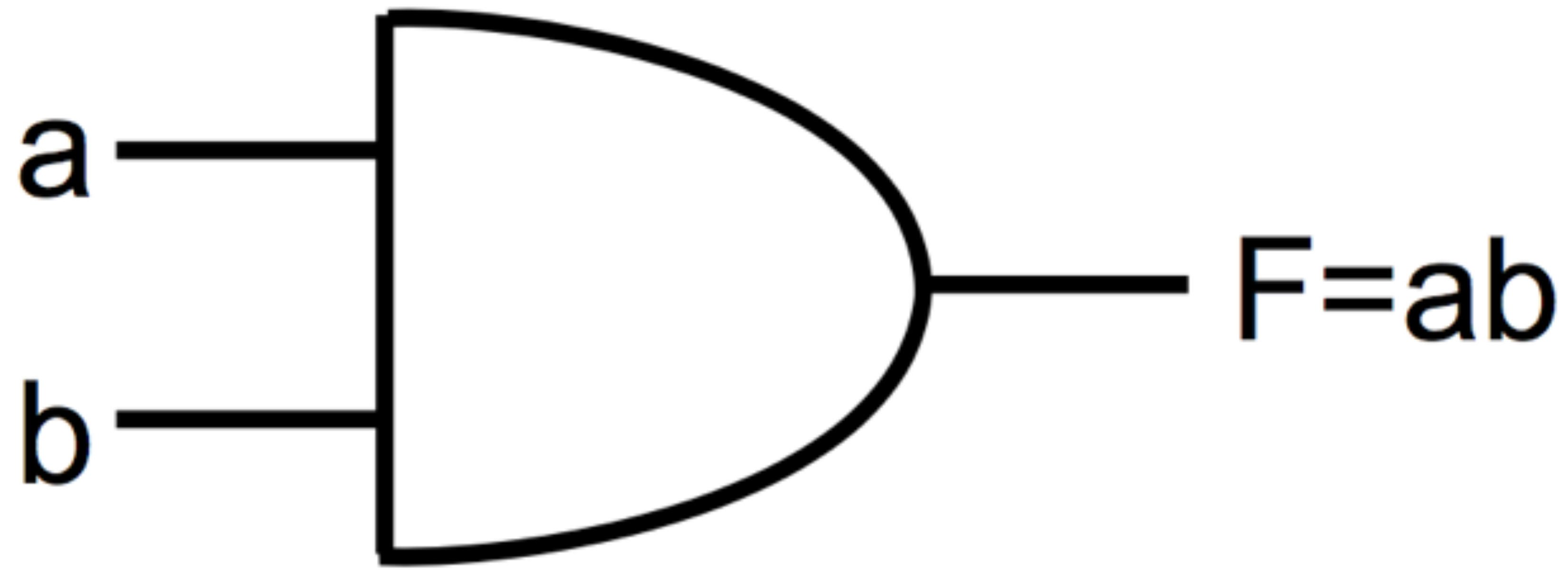
- Logic gates are physical devices that realize simple logic functions
- Symbols are used to represent the fundamental gates Truth tables can be used specify the logic functions



AND Gate

- AND gate implements the AND operation
  Output is TRUE ("1") if and only if all inputs are TRUE ("1")

a	b	F=ab
0	0	0
0	1	0
1	0	0
1	1	1

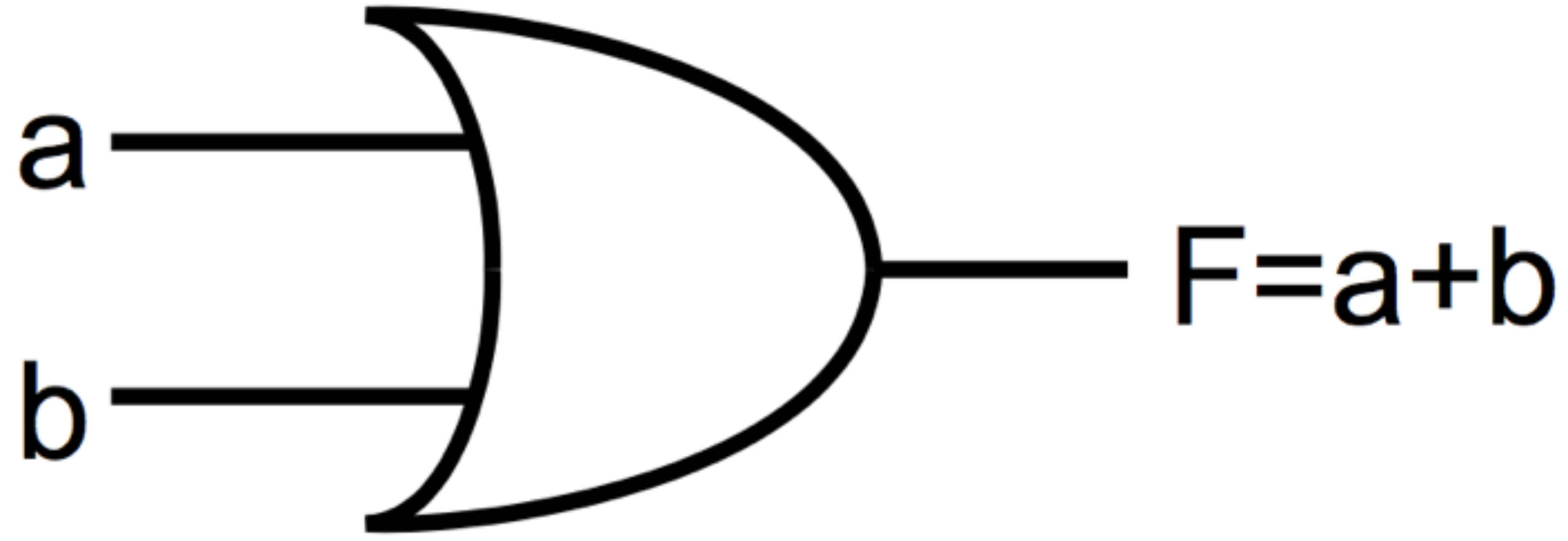


$$F = a \bullet b = ab$$

OR Gate

- OR gate implements the OR operation
- Output is TRUE (“1”) if and only if at least one or more input is TRUE (“1”)

a	b	F=a+b
0	0	0
0	1	1
1	0	1
1	1	1



F = a+b

CHECK POINT

As a checkpoint of your understanding, please pause the video and make sure you can do the following:

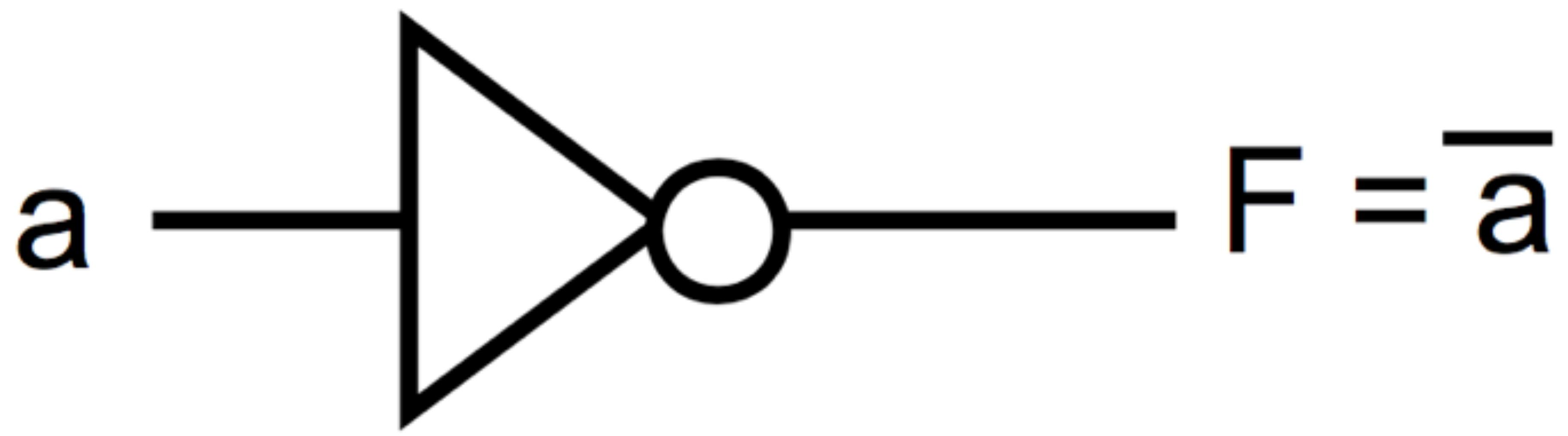
- Write the truth table and the function f, and draw the logic gate symbol for the AND and OR operations.

If you have any difficulties, please review the lecture video before continuing.

NOT Gate (Inverter)

- NOT gate (inverter) implements the NOT operation
- Output is TRUE (“1”) if and only if the input is FALSE (“0”)

a	$F=\overline{a}$
0	1
1	0

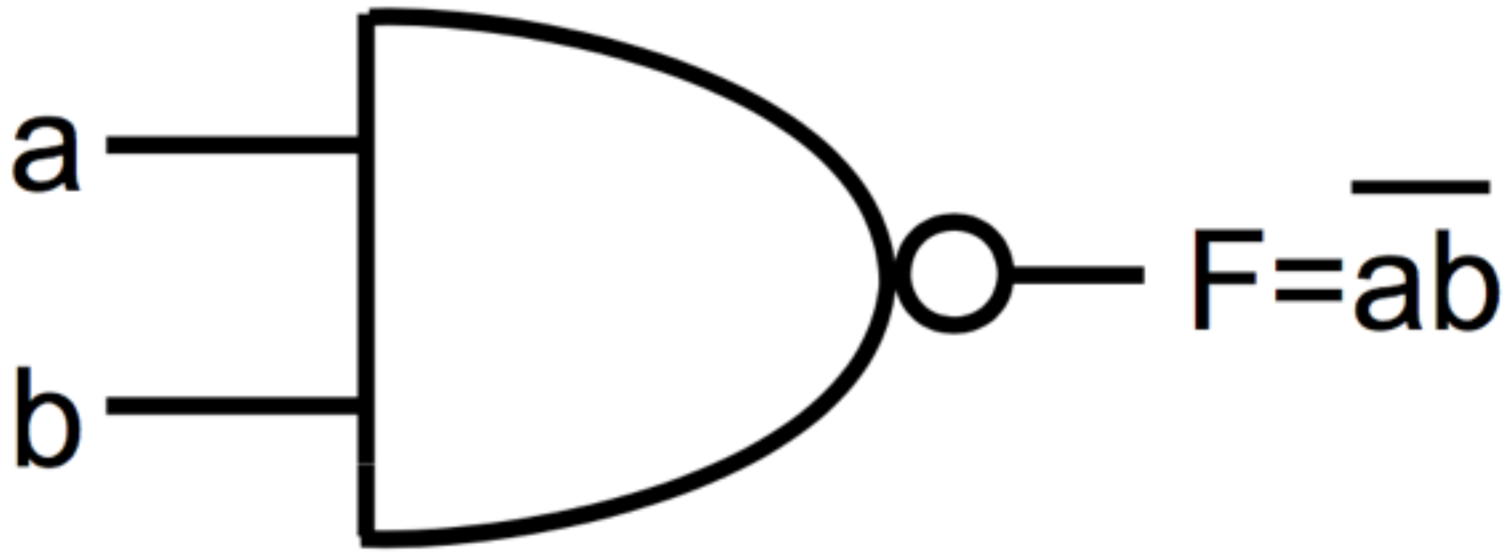


$$F = a' = \overline{a}$$

NAND Gate

- NAND gate implements the NAND operation – logically merges an AND gate with a NOT gate
- Output is TRUE (“1”) if and only if any input is FALSE (“0”)

a	b	$F=\overline{ab}$
0	0	1
0	1	1
1	0	1
1	1	0

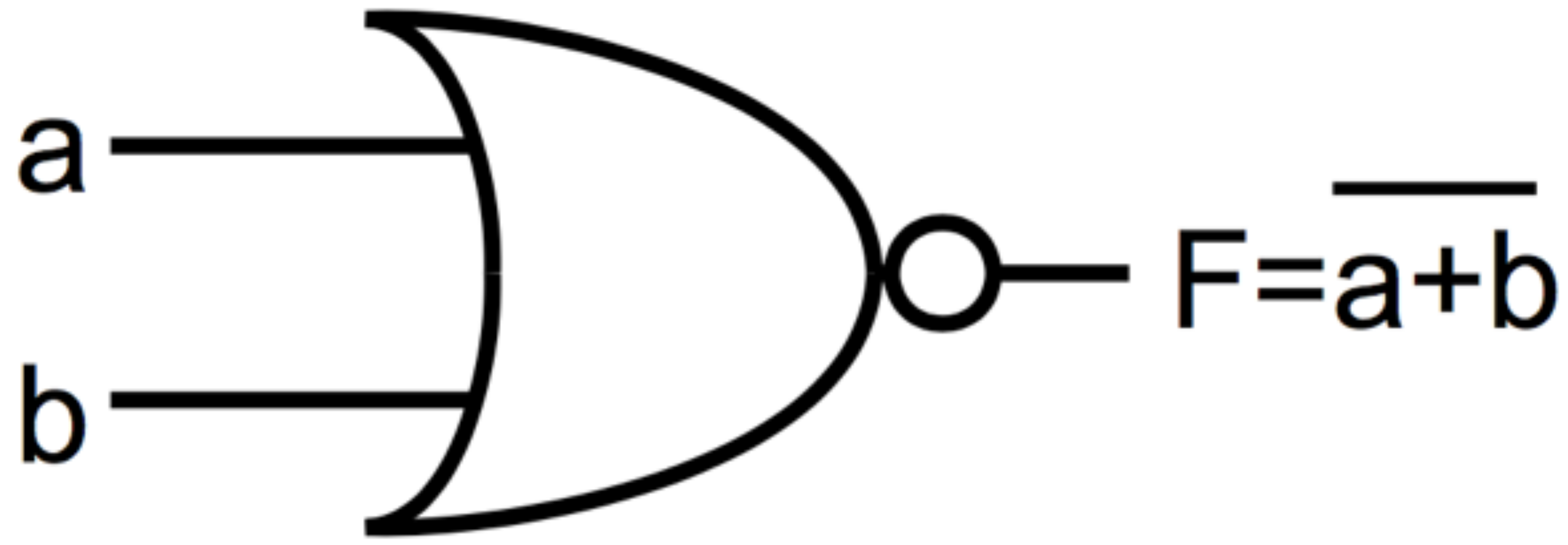


$$F = (ab)' = \overline{ab}$$

NOR Gate

- NOR gate implements the NOR operation – logically merges an OR gate with a NOT gate
- Output is TRUE (“1”) if and only if all inputs are FALSE (“0”)

a	b	$F=\overline{a+b}$
0	0	1
0	1	0
1	0	0
1	1	0

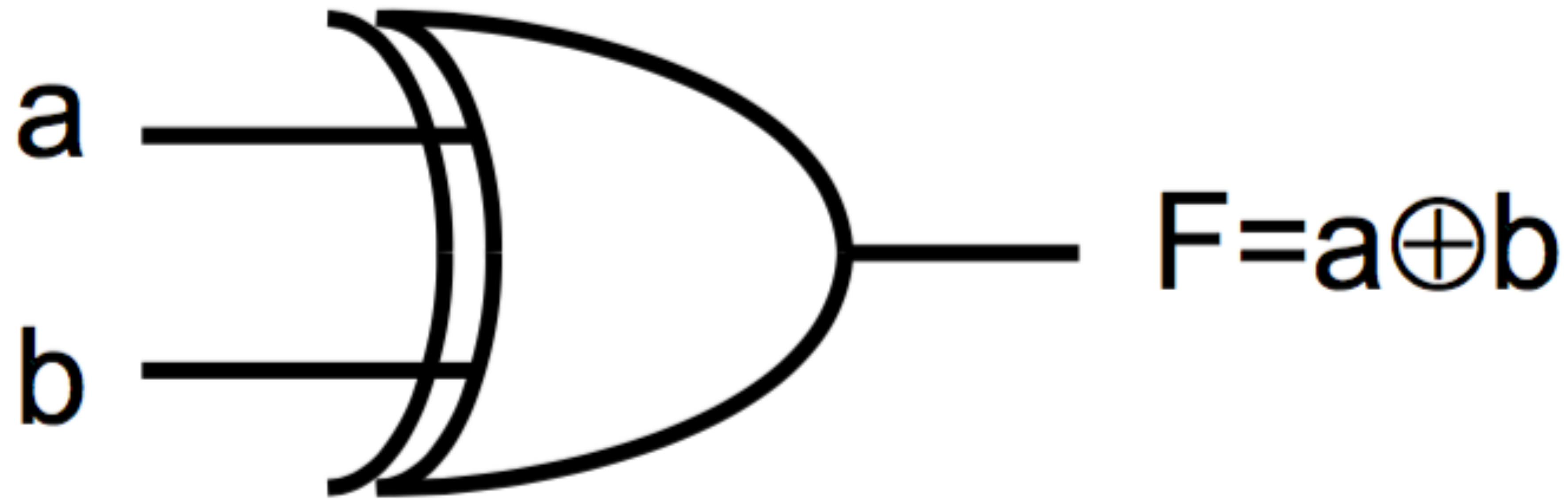


$$F = (a+b)' = \overline{a+b}$$

Exclusive-OR (XOR) Gate

- Output is TRUE ("1") if and only if an odd number of inputs are TRUE ("1")

a	b	$F=a\oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

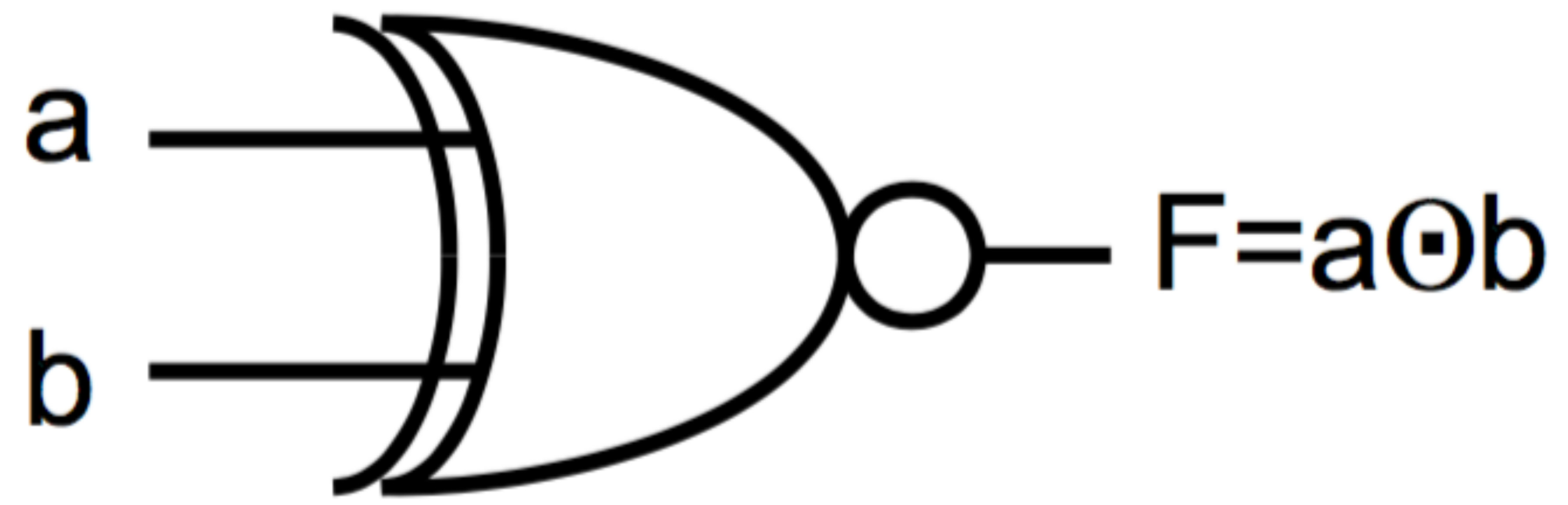


$$F = a\oplus b$$

Exclusive-NOR (XNOR) Gate

- Output is TRUE (“1”) if and only if an even number of inputs are TRUE (“1”)
- The inverse of the XOR gate

a	b	$F = a \odot b$
0	0	1
0	1	0
1	0	0
1	1	1



$$F = (a \oplus b)' = a \odot b$$

CHECK POINT

As a checkpoint of your understanding, please pause the video and make sure you can:

- Write the truth table and the function f, and draw the logic gate symbol for the NOT, NAND, NOR, XOR, and XNOR operations.

If you have any difficulties, please review the lecture video before continuing.

Example: Definition of Logic Function

- Consider the following Boolean function

  $F(a,b,c,d) = \bar{a}(bd + b\bar{d}) + \overline{(a + b + d)} + \overline{[b + (\bar{c} + d)]} + a\overline{bc}d$

- F is a function of the four Boolean variables a, b, c, and d
- It includes the three basic logic operations, AND, OR, and NOT
- Function F can also be defined by a truth table

Example: Truth Table (Top Half)

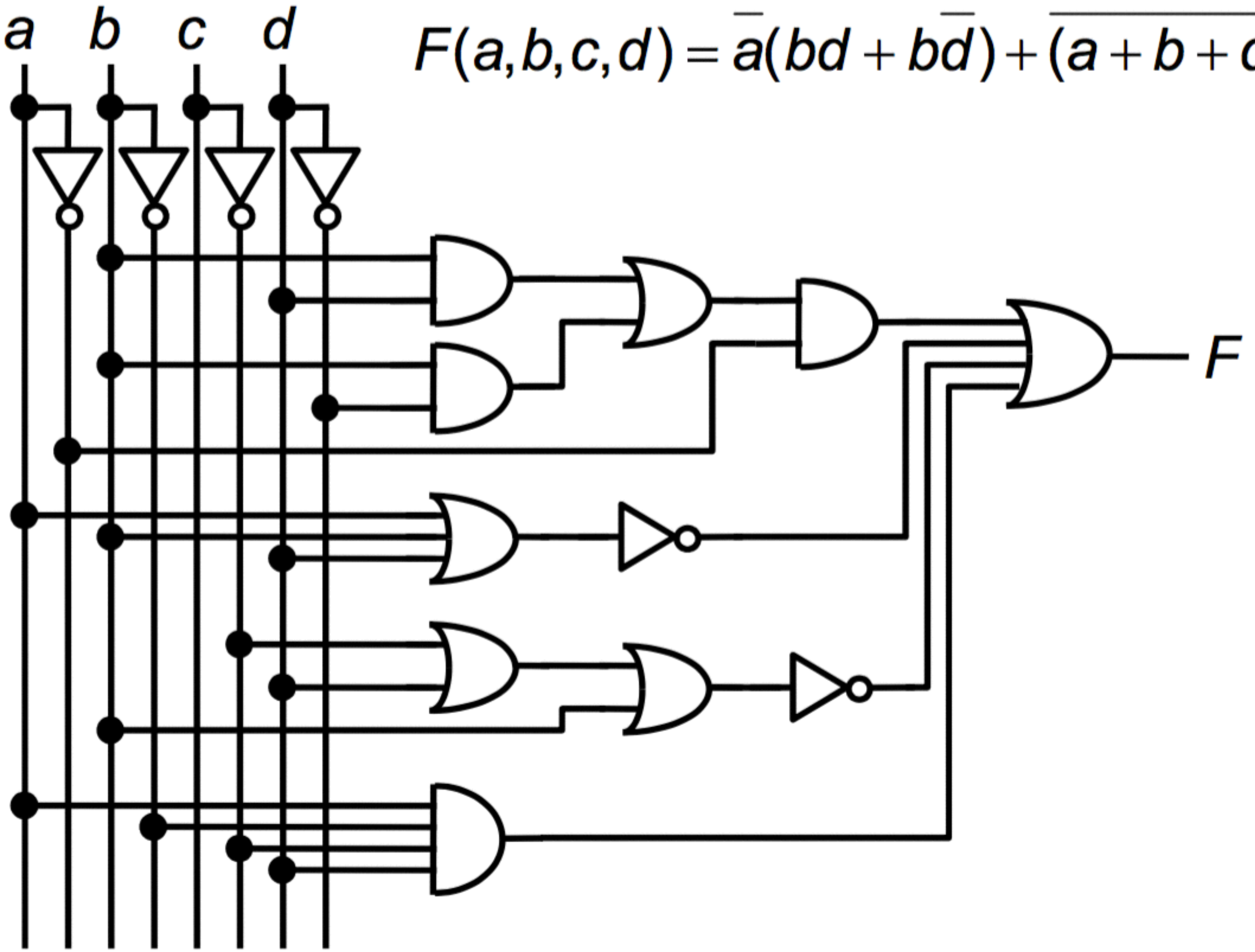
a	b	c	d	bd	bd’	(bd+bd’)	a’(bd+bd’)	(a+b+d)’	(c’+d)	[b+(c’+d)]’	ab’c’d	F
0	0	0	0	0	0	0	0	1	1	0	0	1
0	0	0	1	0	0	0	0	0	1	0	0	0
0	0	1	0	0	0	0	0	1	0	1	0	1
0	0	1	1	0	0	0	0	0	1	0	0	0
0	1	0	0	0	1	1	1	0	1	0	0	1
0	1	0	1	1	0	1	1	0	1	0	0	1
0	1	1	0	0	1	1	1	0	0	0	0	1
0	1	1	1	1	0	1	1	0	1	0	0	1

Example: Truth Table (Bottom Half)

a	b	c	d	bd	bd’	(bd+bd’)	a’(bd+bd’)	(a+b+d)’	(c’+d)	[b+(c’+d)]’	ab’c’d	F
1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	1	0	0	0	0	0	1	0	1	1
1	0	1	0	0	0	0	0	0	0	1	0	1
1	0	1	1	0	0	0	0	0	1	0	0	0
1	1	0	0	0	1	1	0	0	1	0	0	0
1	1	0	1	1	0	1	0	0	1	0	0	0
1	1	1	0	0	1	1	0	0	0	0	0	0
1	1	1	1	1	0	1	0	0	1	0	0	0

Example: Initial Gate Implementation

$$F(a,b,c,d) = \bar{a}(bd + b\bar{d}) + \overline{(a+b+d)} + \overline{[b+(\bar{c}+d)]} + a\bar{b}\bar{c}d$$



Example: Logic Simplification

- It is sometimes advantageous to realize a Boolean function using a different representation of the same function
  - Reduce number of gates
  - Reduce area for an integrated circuit realization
  - Decrease propagation delay
  - Better match an implementation technology
- Manipulation is usually performed by computer-aided design (CAD) tools for logic design
  - Algebraic manipulation
  - Special algorithms, e.g. the Quine-McCluskey algorithm

Example: Algebraic Simplification

$$
\begin{aligned}
F(a,b,c,d) &= \bar{a}\left(bd + b\bar{d}\right) + \overline{\left(a + b + d\right)} + \overline{\left[b + \left(\bar{c} + d\right)\right]} + a\bar{b}\,\bar{c}d \\
&= \bar{a}\left[b\left(d + \bar{d}\right)\right] + \overline{\left(a + b + d\right)} + \overline{\left[b + \left(\bar{c} + d\right)\right]} + a\bar{b}\,\bar{c}d \\
&= \bar{a}\left[b(1)\right] + \overline{\left(a + b + d\right)} + \overline{\left[b + \left(\bar{c} + d\right)\right]} + a\bar{b}\,\bar{c}d \\
&= \bar{a}b + \overline{\left(a + b + d\right)} + \overline{\left[b + \left(\bar{c} + d\right)\right]} + a\bar{b}\,\bar{c}d \\
&= \bar{a}b + \bar{a}\,\bar{b}\,\bar{d} + \overline{\left[b + \left(\bar{c} + d\right)\right]} + a\bar{b}\,\bar{c}d \\
&= \bar{a}b + \bar{a}\,\bar{b}\,\bar{d} + \bar{b}\,\overline{\left(\bar{c} + d\right)} + a\bar{b}\,\bar{c}d \\
&= \bar{a}b + \bar{a}\,\bar{b}\,\bar{d} + \bar{b}\left(\bar{\bar{c}}\,\bar{d}\right) + a\bar{b}\,\bar{c}d \\
&= \bar{a}b + \bar{a}\,\bar{b}\,\bar{d} + \bar{b}\left(c\bar{d}\right) + a\bar{b}\,\bar{c}d \\
&= \bar{a}b + \bar{a}\,\bar{b}\,\bar{d} + \bar{b}\,c\bar{d} + a\bar{b}\,\bar{c}d
\end{aligned}
$$

Example: Algebraic Simplification (more)

$$
\begin{aligned}
F(a,b,c,d) &= \bar{a}\,b + \bar{a}\,\bar{b}\,\bar{d} + \bar{b}\,c\,\bar{d} + a\,\bar{b}\,\bar{c}\,d \\
&= \bar{a}\,b\,(1) + \bar{a}\,\bar{b}\,\bar{d} + \bar{b}\,c\,\bar{d} + a\,\bar{b}\,\bar{c}\,d \\
&= \bar{a}\,b\,(1 + \bar{d}) + \bar{a}\,\bar{b}\,\bar{d} + \bar{b}\,c\,\bar{d} + a\,\bar{b}\,\bar{c}\,d \\
&= \bar{a}\,b + \bar{a}\,b\,\bar{d} + \bar{a}\,\bar{b}\,\bar{d} + \bar{b}\,c\,\bar{d} + a\,\bar{b}\,\bar{c}\,d \\
&= \bar{a}\,b + \bar{a}\,\bar{d}\,b + \bar{a}\,\bar{d}\,\bar{b} + \bar{b}\,c\,\bar{d} + a\,\bar{b}\,\bar{c}\,d \\
&= \bar{a}\,b + \bar{a}\,\bar{d}\,(b + \bar{b}) + \bar{b}\,c\,\bar{d} + a\,\bar{b}\,\bar{c}\,d \\
&= \bar{a}\,b + \bar{a}\,\bar{d}\,(1) + \bar{b}\,c\,\bar{d} + a\,\bar{b}\,\bar{c}\,d \\
&= \bar{a}\,b + \bar{a}\,\bar{d} + \bar{b}\,c\,\bar{d} + a\,\bar{b}\,\bar{c}\,d
\end{aligned}
$$

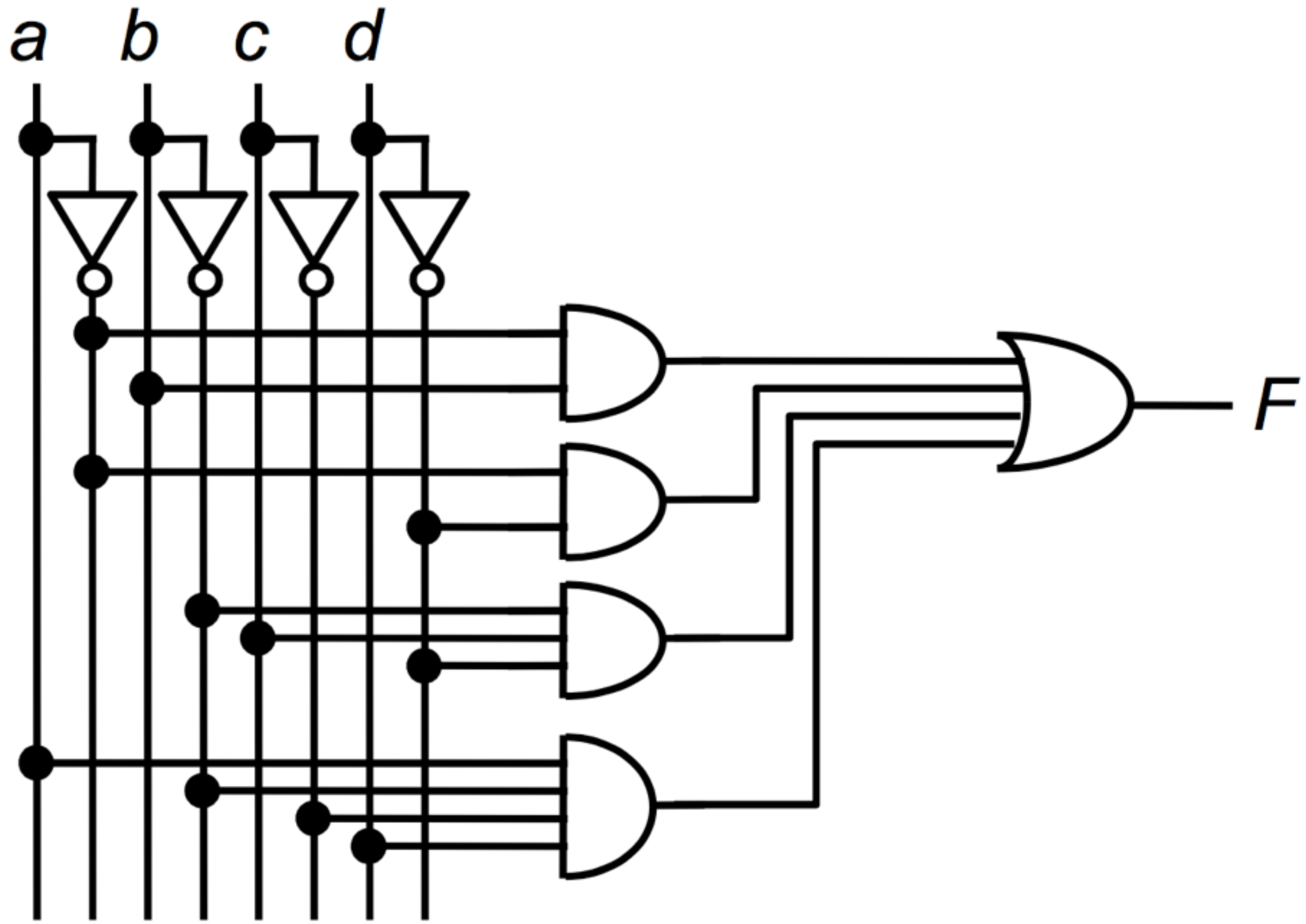
CHECK POINT

As a checkpoint of your understanding, please pause the video and make sure you can:

- Work through by hand the steps to simply the function F on slide 21.

If you have any difficulties, please review the lecture video before continuing.

Simplified Implementation



$$F(a,b,c,d) = \bar{a}b + \bar{a}\bar{d} + \bar{b}c\bar{d} + a\bar{b}\bar{c}d$$

Example: Comparison

- Both logic circuits realize the same Boolean function
- Let's compare the original realization and the new realizatior

	Inverters	Gates	Longest Path
Original	6	9	5
New	4	5	3

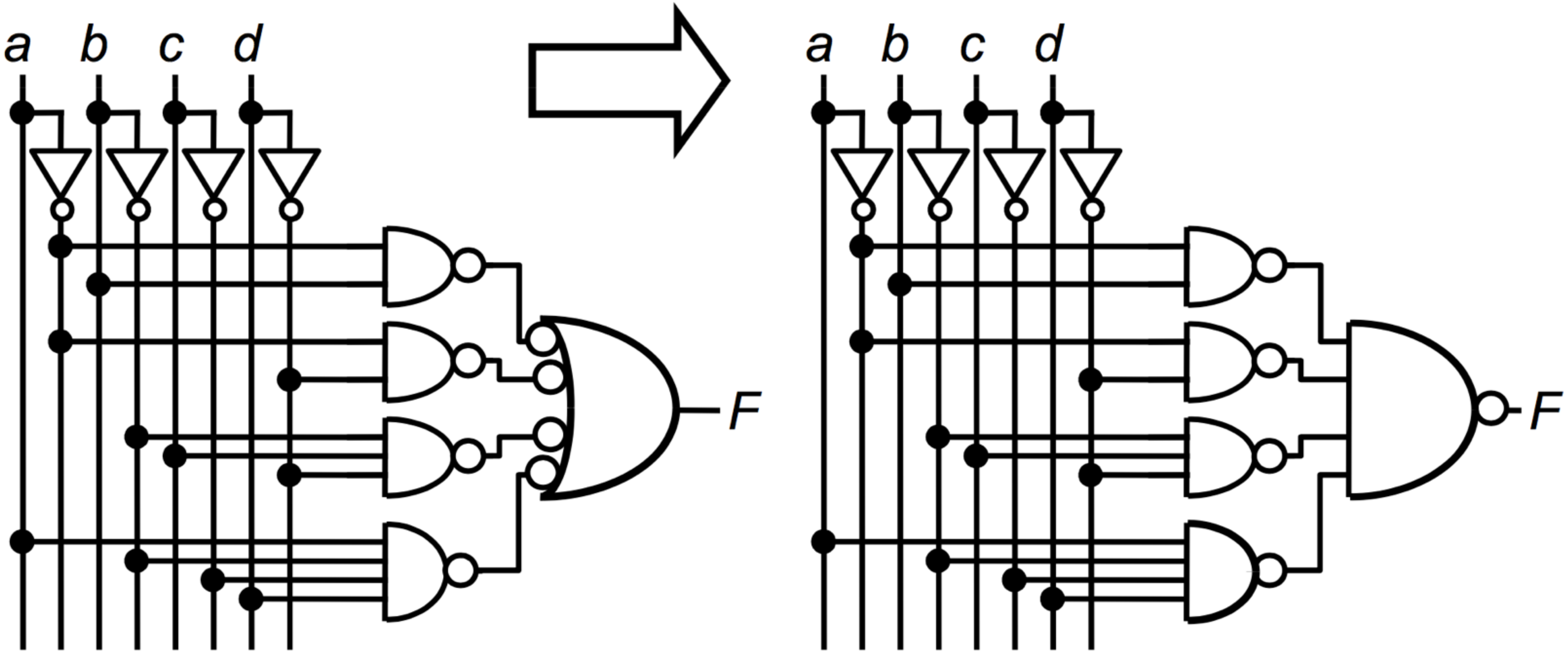
- While an exact comparison depends on the implementation technology, it is likely that the new design requires less area and operates with less delay than the original design

Example: NAND Gate Implementation (1)

- Back to back “invert bubbles” are added between the first-level AND gates and the second-level OR gate
- NAND gates replace each AND and OR gate

$$F(a,b,c,d) = \overline{\overline{\bar{a}b} \bullet \overline{\bar{a}\bar{d}} \bullet \overline{\bar{b}c\bar{d}} \bullet \overline{a\bar{b}\bar{c}d}}$$

Example: NAND Gate Implementation (2)



CHECK POINT

As a checkpoint of your understanding, please pause the video and make sure you can:

- On slide 24 write by hand the simplified function F and draw its logic gate diagram.

If you have any difficulties, please review the lecture video before continuing.

Summary

- Logic values (“1” and “0”) are abstractions of physical signals with “1” or “0” values based on thresholds for a particular technology
- Combinational logic functions depend only on current inputs; sequential logic functions depend on current and past inputs
- Logic gates implement simple logic operations such as AND, OR, NOT, and NAND
- As an example, we realized a Boolean function using gates and found an alternative realization using algebraic manipulation of the expression

MODULE 3: Boolean Algebra and Digital Logic

Lecture 3.3
Logic Gates

Prepared By:

- Scott F. Midkiff, PhD
- Luiz A. DaSilva, PhD
- Kendall E. Giles, PhD

Electrical and Computer Engineering

Virginia Tech