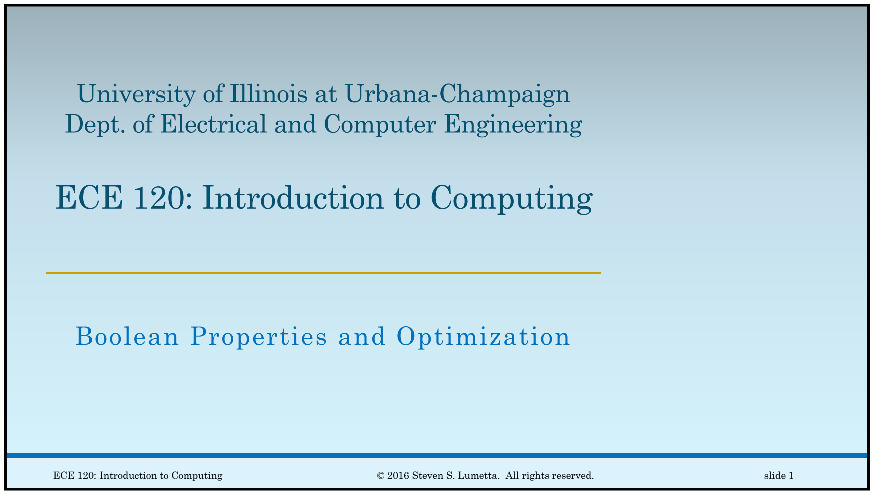
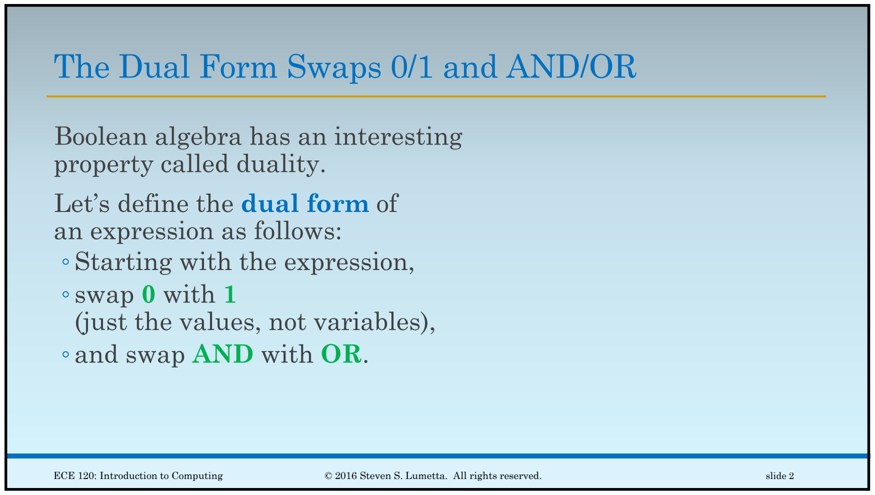


University of Illinois at Urbana-Champaign
Dept. of Electrical and Computer Engineering

ECE 120: Introduction to Computing

Boolean Properties and Optimization

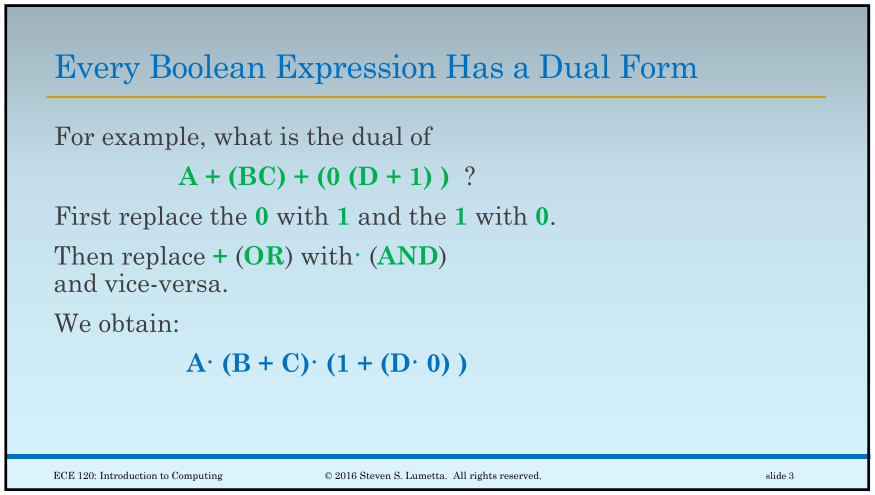


The Dual Form Swaps 0/1 and AND/OR

Boolean algebra has an interesting property called duality.

Let's define the **dual form** of an expression as follows:

- Starting with the expression,
- swap **0** with **1** (just the values, not variables),
- and swap **AND** with **OR**.



Every Boolean Expression Has a Dual Form

For example, what is the dual of

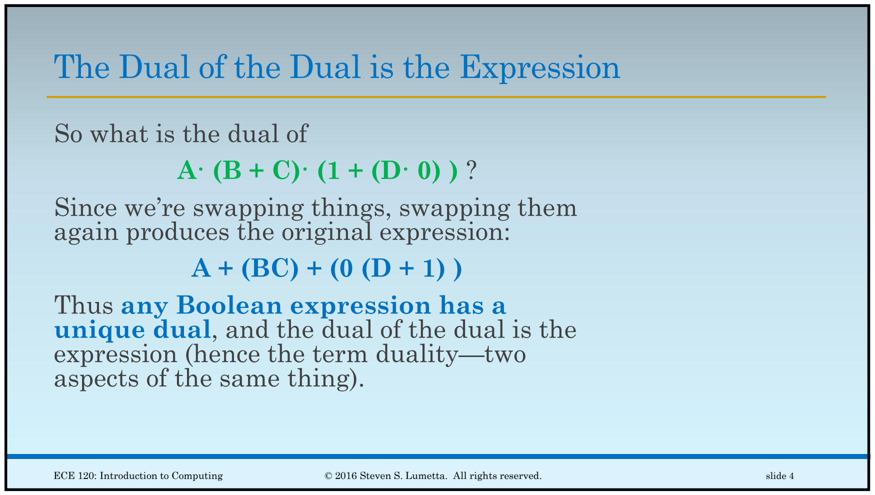
$$A + (BC) + (0\ (D + 1)) \ ?$$

First replace the **0** with **1** and the **1** with **0**.

Then replace **+** (**OR**) with · (**AND**) and vice-versa.

We obtain:

$$A \cdot (B + C) \cdot (1 + (D \cdot 0))$$



The Dual of the Dual is the Expression

So what is the dual of

$$A \cdot (B + C) \cdot (1 + (D \cdot 0)) \ ?$$

Since we're swapping things, swapping them again produces the original expression:

$$A + (BC) + (0\ (D + 1))$$

Thus **any Boolean expression has a unique dual**, and the dual of the dual is the expression (hence the term duality—two aspects of the same thing).

Pitfall: Do Not Change the Order of Operations

Be careful not to change the order of operations when finding a dual form.

For example, the dual form of

A + BC

is

A (B + C)

The operation on **B** and **C** must happen before the other operation.

Why Do You Care? One Reason: the Principle of Duality

Three reasons:
- CMOS gate structures are dual forms
- Quick way to complement any expression
- the principle of duality

Let's start with the last, which we'll use shortly (when we examine more properties).

Principle of duality: **If a Boolean theorem or identity is true/false, so is the dual of that theorem or identity.**

Generalized DeMorgan is Quick and Easy

Let's say that we have an expression **F**.

To find **F'** … apply DeMorgan's Laws …

Apply repeatedly, as many times as necessary.

Or use the generalized version based on duality:
- Write the dual form of **F**.
- Swap variables and complemented variables.
- (That's all.)

An Example of Finding a Complement with the Dual Form

F = AB (C + (DL'G(B' + A + E))) (H + (J'A'B))

What's F'?

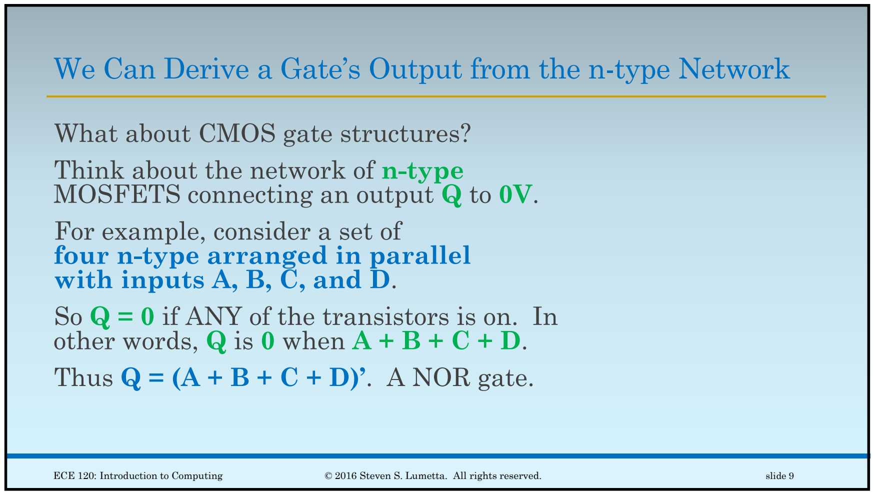
The dual is

**A + B + (C (D + L' + G + (B'AE))) +
(H (J' + A' + B))**

So

**F' = A' + B' + (C' (D' + L + G' + (BA'E'))) +
(H' (J + A + B'))**

You can skip the middle step once you're comfortable with the process.



We Can Derive a Gate's Output from the n-type Network

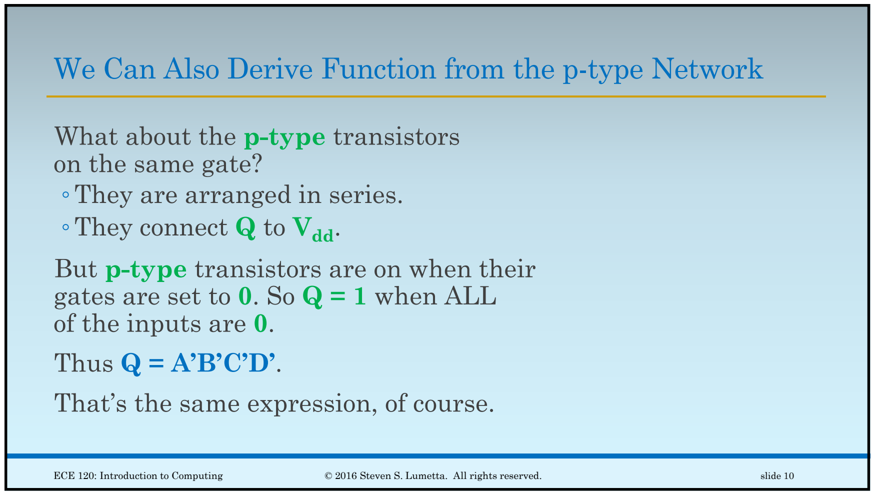
What about CMOS gate structures?

Think about the network of **n-type** MOSFETS connecting an output **Q** to **0V**.

For example, consider a set of **four n-type arranged in parallel with inputs A, B, C, and D**.

So **Q = 0** if ANY of the transistors is on. In other words, **Q** is **0** when **A + B + C + D**.

Thus **Q = (A + B + C + D)'**. A NOR gate.



We Can Also Derive Function from the p-type Network

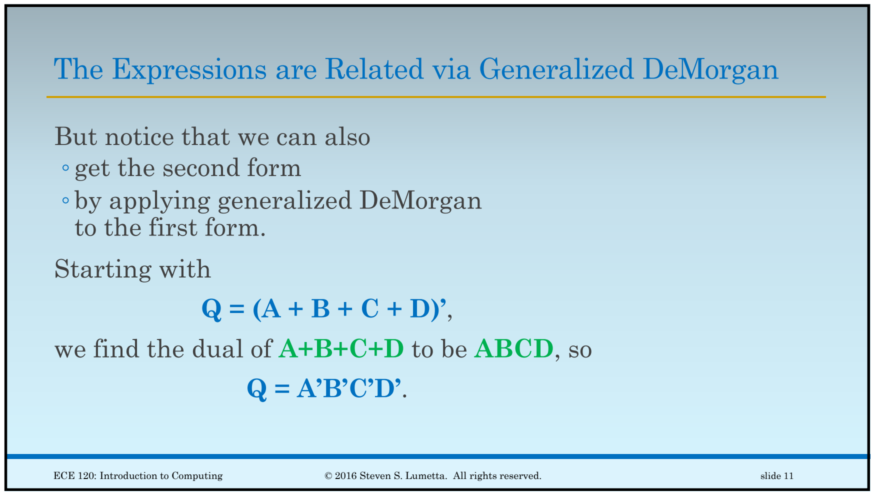
What about the **p-type** transistors on the same gate?

- They are arranged in series.
- They connect **Q** to $\mathbf{V_{dd}}$.

But **p-type** transistors are on when their gates are set to **0**. So **Q = 1** when ALL of the inputs are **0**.

Thus **Q = A'B'C'D'**.

That's the same expression, of course.



The Expressions are Related via Generalized DeMorgan

But notice that we can also

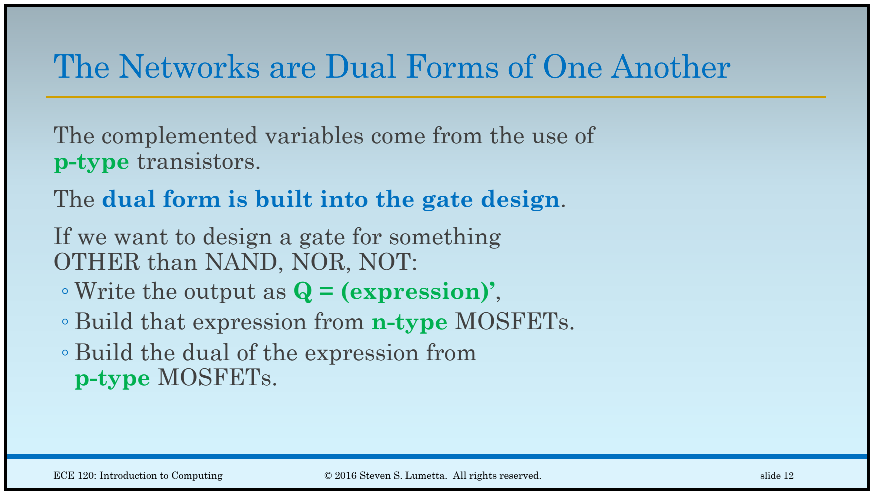
- get the second form
- by applying generalized DeMorgan to the first form.

Starting with

Q = (A + B + C + D)',

we find the dual of **A+B+C+D** to be **ABCD**, so

Q = A'B'C'D'.



The Networks are Dual Forms of One Another

The complemented variables come from the use of **p-type** transistors.

The **dual form is built into the gate design**.

If we want to design a gate for something OTHER than NAND, NOR, NOT:

- Write the output as **Q = (expression)'**,
- Build that expression from **n-type** MOSFETs.
- Build the dual of the expression from **p-type** MOSFETs.

An Example of an Unusual Gate

Consider the gate here:

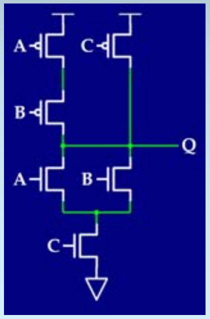
From the **n-type** network,

Q = ((A + B) C)'

The dual of the expression (ignoring the complement) is

AB + C

which is the structure of the **p-type** network.



Area and Speed for the Unusual Gate

So the function **Q = ((A + B) C)'** requires **six transistors and one gate delay**.

We can, of course, limit ourselves to NAND/NOR gates.

In that case, **Q = ((A'B')' C)'**

We use one two-input NAND for **(A'B')'**, and a second two-input NAND for **Q**.

If we assume that **A'** and **B'** are available, **the NAND design requires eight transistors and two gate delays**.

Optimization versus Abstraction

Most designers just use NAND and NOR (or, today, even higher-level abstractions!).

In general:
- breaking abstraction boundaries can give us an advantage,
- but the boundaries make the design task less complex,
- which improves human productivity and reduces the likelihood of mistakes.

That's another tradeoff.

Computer aided design (CAD) tools can perform some of these optimizations for us, too.

Simple Boolean Properties

Easy, but useful to commit to memory for analyzing circuits…

$1 + A = 1$	$0 \cdot A = 0$
$1 \cdot A = A$	$0 + A = A$
$A + A = A$	$A \cdot A = A$
$A \cdot A' = 0$	$A + A' = 1$

(Each row gives two dual forms.)

More Dual Form Boolean Properties

DeMorgan’s Laws are also dual forms

(A + B)’ = A’B’ **(AB)’ = A’ + B’**

What about distributivity? Here’s the rule that you know from our usual algebra

A(B + C) = AB + AC

(multiplication distributes over addition)

It’s also true in Boolean algebra:
AND distributes over OR.

OR Also Distributes Over AND in Boolean Algebra

A(B + C) = AB + AC

Now take the dual form…

A + BC = (A + B)(A + C)

OR distributes over AND!

(Note that this property does NOT hold in our usual algebra. $14 + 7 \cdot 4 \neq (14 + 7)(14 + 4)$)

One More Property: Consensus

The last property is non-intuitive.

AB + A’C + BC = AB + A’C

It’s called “consensus” because
- the first two terms TOGETHER (when both are true, and thus reach a consensus) imply the third term
- so the third term can be dropped.

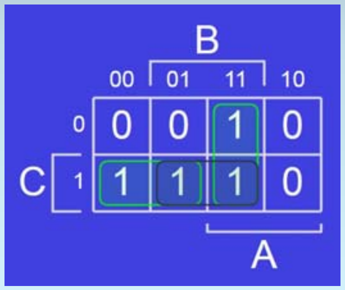
A K-Map Illustrates Consensus Well

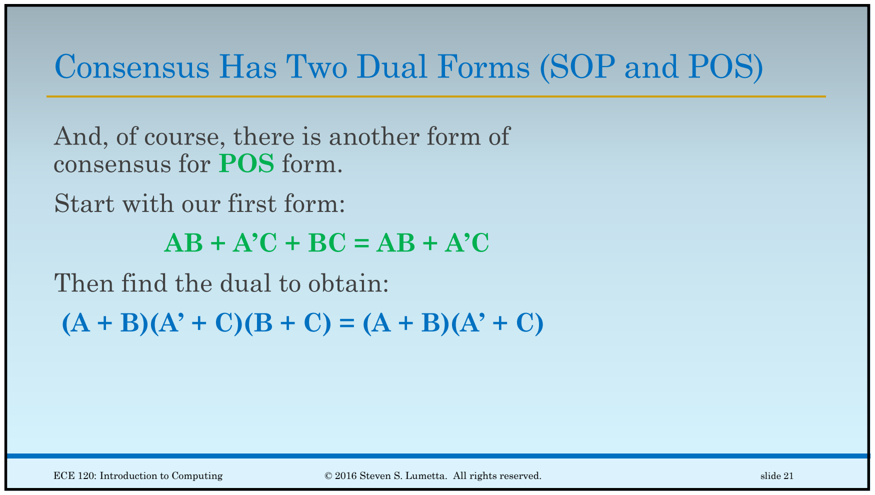
Let’s look at a K-map.

AB is the vertical green loop.

A’C is the horizontal green loop.

BC is the black loop.





Consensus Has Two Dual Forms (SOP and POS)

And, of course, there is another form of consensus for **POS** form.

Start with our first form:

$$AB + A'C + BC = AB + A'C$$

Then find the dual to obtain:

$$(A + B)(A' + C)(B + C) = (A + B)(A' + C)$$