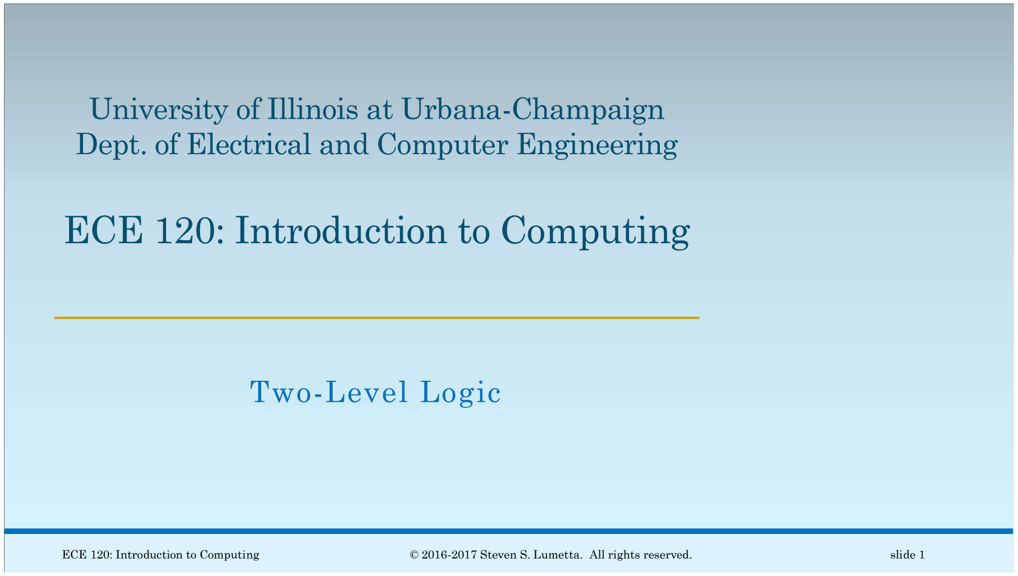
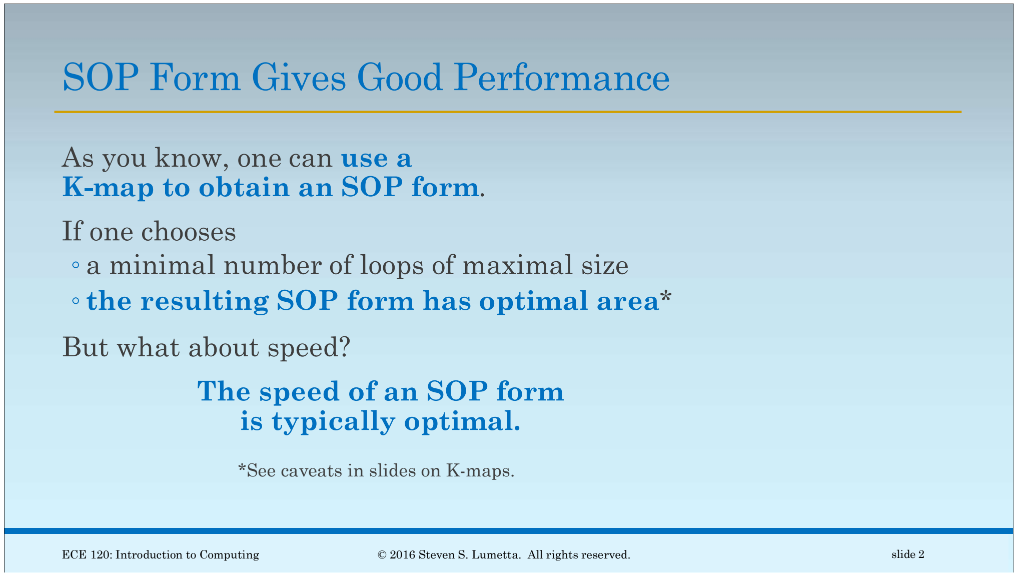


University of Illinois at Urbana-Champaign
Dept. of Electrical and Computer Engineering

ECE 120: Introduction to Computing

Two-Level Logic



SOP Form Gives Good Performance

As you know, one can **use a K-map to obtain an SOP form**.

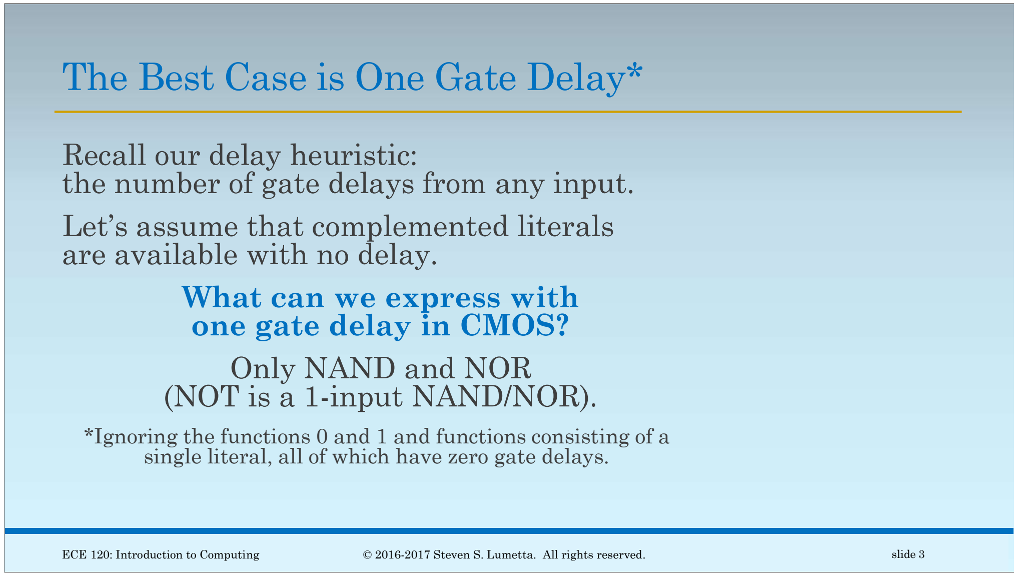
If one chooses

- a minimal number of loops of maximal size
- **the resulting SOP form has optimal area***

But what about speed?

The speed of an SOP form is typically optimal.

*See caveats in slides on K-maps.



The Best Case is One Gate Delay*

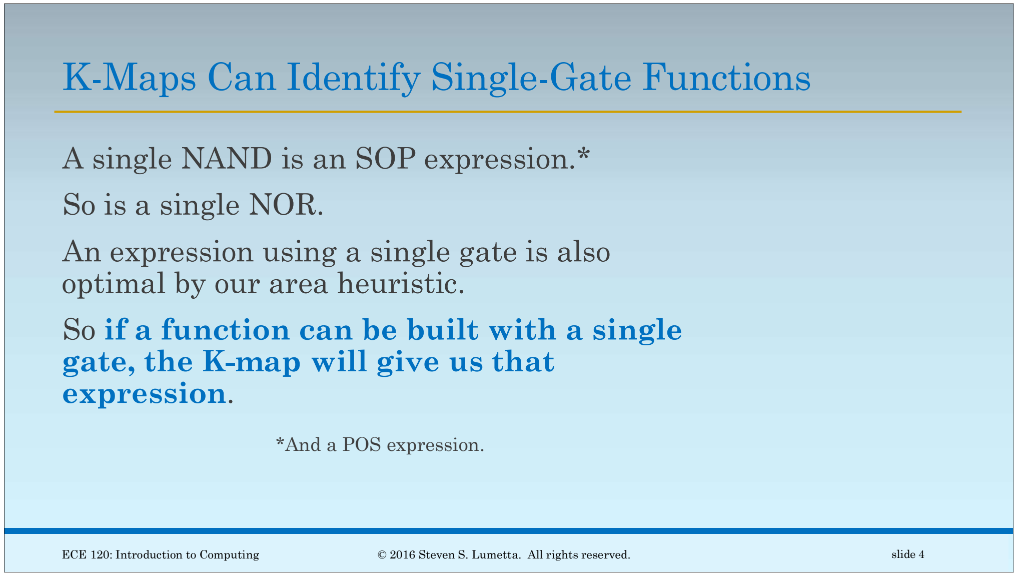
Recall our delay heuristic:
the number of gate delays from any input.

Let's assume that complemented literals are available with no delay.

What can we express with one gate delay in CMOS?

Only NAND and NOR
(NOT is a 1-input NAND/NOR).

*Ignoring the functions 0 and 1 and functions consisting of a single literal, all of which have zero gate delays.



K-Maps Can Identify Single-Gate Functions

A single NAND is an SOP expression.*

So is a single NOR.

An expression using a single gate is also optimal by our area heuristic.

So **if a function can be built with a single gate, the K-map will give us that expression**.

*And a POS expression.

Is Counting AND/OR Gates Realistic?

Most functions cannot be expressed as a single NAND/NOR gate.

So **how fast is an SOP expression**?

Two gate delays.

AND, followed by OR.

But in CMOS, we only have NAND and NOR.

How many gate delays do we get if we only use NAND/NOR?

Let's Introduce Some Algebra

A little Boolean algebra will help us:

DeMorgan's Laws

(AB)' = A' + B' **(A+B)' = A'B'**

Want a proof? Use a truth table (4 lines each).

They also generalize to more than two inputs.

For example,

(ABC)' = A' + B' + C' **(A+B+C)' = A'B'C'**

DeMorgan's Laws Relate NAND/NOR to AND/OR

What do DeMorgan's Laws mean?

Here's one way to think about them:

- **(AB)' = A' + B'** NAND is the same as OR on the complements of the inputs.
- **(A+B)' = A'B'** NOR is the same as AND on the complements of the inputs.

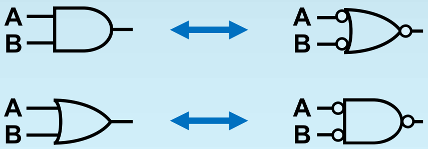
A Graphical Representation Can Be Useful, Too

Let's also think about them graphically.

Complement both sides first, so we have…

AB = (A' + B')' **A+B = (A'B')'**

and now we can draw gates…



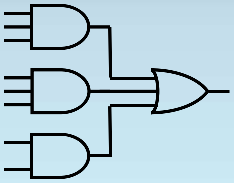
How Do We Draw an SOP Form? AND, then OR

What were we talking about?

Ah, speed of SOP forms.

SOP is AND followed by OR.

Something like this…

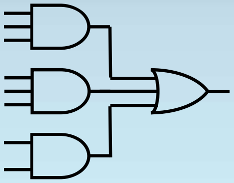


(with some number of AND gates, each with some number of inputs)

Apply DeMorgan's Laws Graphically

Use DeMorgan's law on the OR gate.

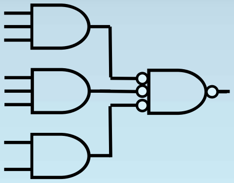
Replace it with a NAND with inverted inputs.



Apply DeMorgan's Laws Graphically

Use DeMorgan's law on the OR gate.

Replace it with a NAND with inverted inputs.



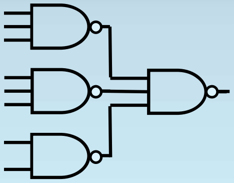
Remember that the **input bubbles mean inverters** (NOT).

Now **slide them down the wires to the left** until they sit in front of the ANDs.

Apply DeMorgan's Laws Graphically

Use DeMorgan's law on the OR gate.

Replace it with a NAND with inverted inputs.



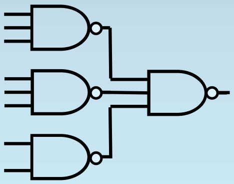
Remember that the **input bubbles mean inverters** (NOT).

Now **slide them down the wires to the left** until they sit in front of the ANDs.

SOP Form Speed is Two Gate Delays

We didn't change the function of the circuit.

But now all of the gates are NAND gates.



So **we can build any SOP function using two levels of NAND**.

And the speed? **Two gate delays.**

SOP and POS Forms Give Us Two-Level Logic

We can use two levels of NANDs to build any SOP expression.

We refer to this approach as **two-level logic**.

For a POS expression
- **one can do exactly the same thing**
- replacing OR followed by AND
- **with NOR followed by NOR**.

So **any POS expression also requires two gate delays** (again, assuming that complemented inputs are free).

Use a K-Map to Find POS Expressions

But **how can we find a POS form?**

Again, **use a K-map**.

1. Given a function **F**, draw a K-map for **F'**.
2. Use K-map to **find an SOP form for F'**.
3. **Complement the result** to find **F**
   - and apply DeMorgan's laws a few times,
   - **complement** of SOP form **is POS form**.

In Practice, Form Loops Around 0s to Find POS

In practice, **just circle 0s instead of 1s**.

Recall that a box in a K-map
- when filled with a 1
- corresponds to a **minterm**.

The same box
- when filled with a 0
- corresponds to a **maxterm**
- an expression that produces exactly one 0 row in its truth table.

Complement Literals When Reading POS Factors

But be careful: the **maxterm** has all variables complemented relative to the **minterm**.

For example,

- a box corresponding to **minterm ABC'** (equal to 1 when **A=1** and **B=1** and **C=0**)
- corresponds to **maxterm A' + B' + C** (equal to 0 when **A=1** and **B=1** and **C=0**)

SOP and POS Forms Give Us Two-Level Logic

To **find a POS form** that has optimal area (among POS forms),

- **follow the same approach** as before,
- but instead of drawing loops around 1s,
- **draw loops around 0s**.

Again, **do not forget to complement the literals relative to their form for implicants!**

(And write each loop as a sum, not as a product.)

Which Form is Better? Solve Both and Compare

Which gives better area, SOP or POS?

That depends on the function.

Solve both ways and compare.

You will have some experience finding POS forms in discussion section.

You can also use the online tool, but the exercises are not as direct as for SOP.