Lecture Outline

Reminders to self:

- ❑ Turn on lecture recording to Cloud
- ❑ Turn on Zoom microphone

- Last Lecture
  - Finished adders with carry look ahead
  - K-maps for POS; implicants (prime and essential prime)
  - Started multi-level logic
- Today's Lecture
  - Review of Participation Quiz 5
  - Continue multi-level logic
    - NAND-NAND 2-level logic (and others)
    - Limited Fan-in Design

Handouts and Announcements

- Announcements
  - Homework Problems 5-3 and 5-4
    - Posted on Carmen Saturday (2/4)
    - Due: 11:59pm Thursday 2/9
  - Homework Reminders: HW 5-1 and 5-2 past due
  - Read for Wednesday: pages 214-218

Handouts and Announcements

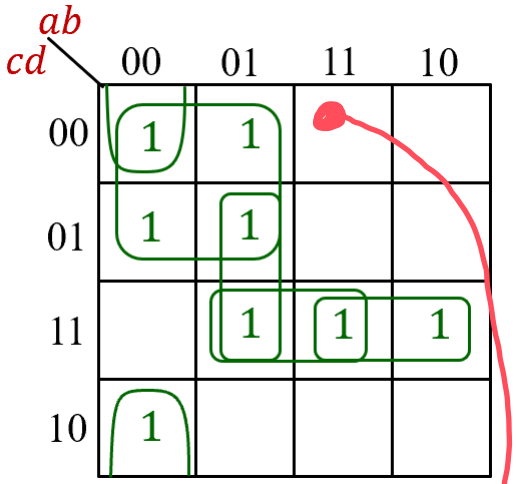
- Announcements
  - Mini-Exam 2 Reminder
    - Available 5pm Monday 2/6 through 5:00pm Tuesday 2/7
    - Due in Carmen PROMPTLY at 5:00pm on 2/7
    - Designed to be completed in ~36 min, but you may use more
    - When planning your schedule:
      - I recommend building in 10-15 min extra
      - To allow for downloading exam, signing and dating honor pledge, saving solution as pdf, and uploading to Carmen
    - I also recommend not procrastinating
  - Exam review topics available on Carmen
  - Sample Mini-Exams 1 and 2 from Au20 also available

Implicants

Review of Carmen Quiz 8 - implicants:

Q1) Which of the following is **NOT** an implicant of the function shown in this K-map?

- a'bc'd
- a'bd
- a'c'
- abc'd'

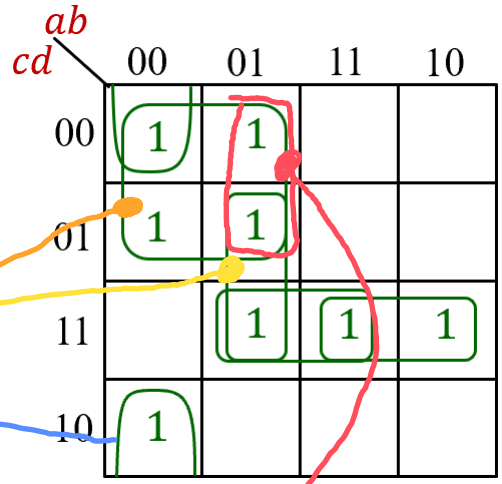


Implicants

Review of Carmen Quiz 8 - implicants:

Q2) Which of the following is **NOT** a *prime* implicant of the function shown in this K-map?

- a'b'd'
- a'bd
- a'c'
- a'bc'

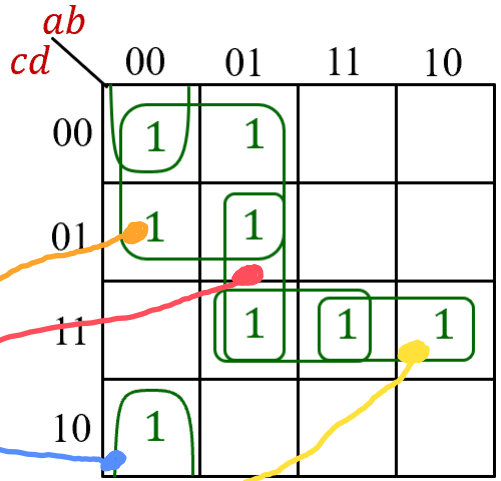


Implicants

Review of Carmen Quiz 8 - implicants:

Q3) Which of the following is **NOT** an *essential* prime implicant of the function shown in this K-map?

- a'b'd'
- acd
- a'c'
- a'bd

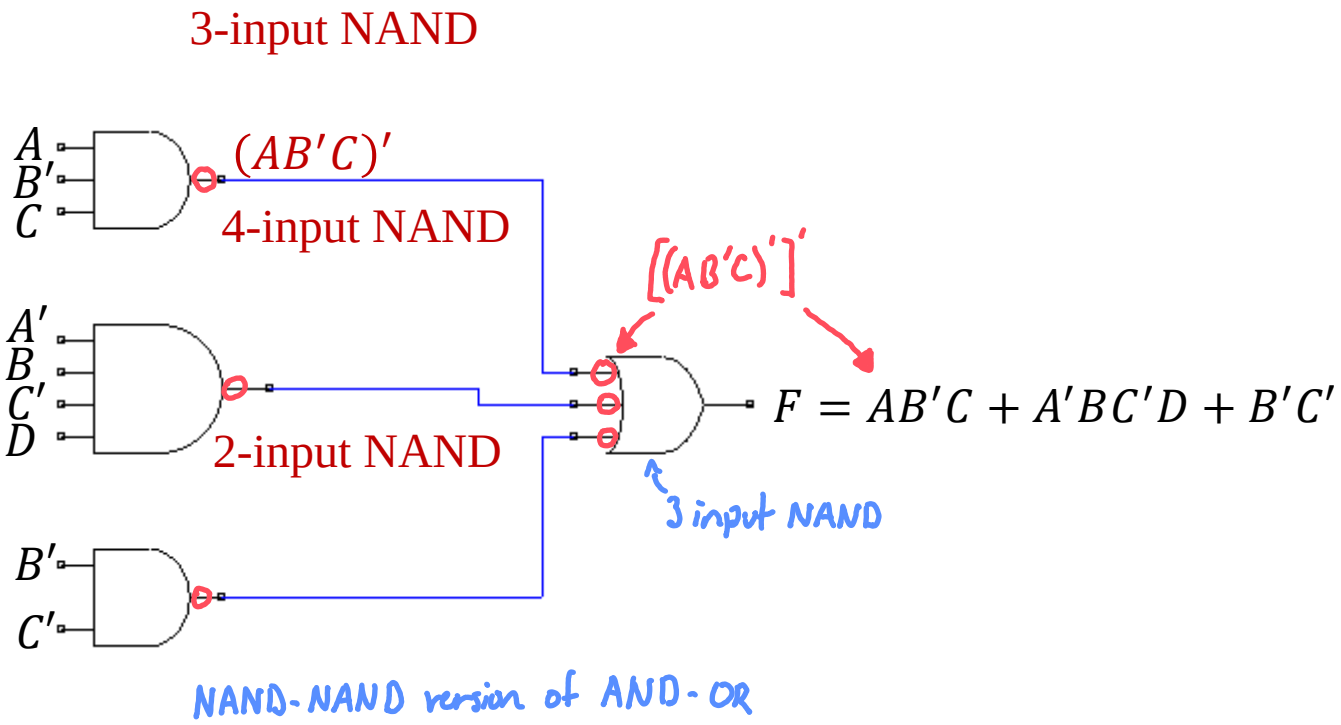


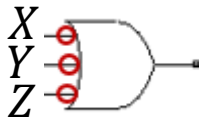
Two-level NAND Gate Logic

- So far for design of logic circuits we have primarily focused on using AND, OR and NOT gates
- But at transistor-level, logic gates are most efficiently designed as modifications of inverters
- NAND and NOR gates are the natural result
  - They require fewer transistors
  - They are faster (shorter gate delay)
  - They use less area on ICs (< $)
- Previously saw (Lecture 7) that NAND is functionally complete
  - Can implement any logic function using only NAND gates
  - Similarly, can implement any logic function using only NOR gates
- We will now look at implementing SOP AND-OR circuits with NAND gates: NAND-NAND

Two-level NAND Gate Logic

- ○ → negation "bubble"



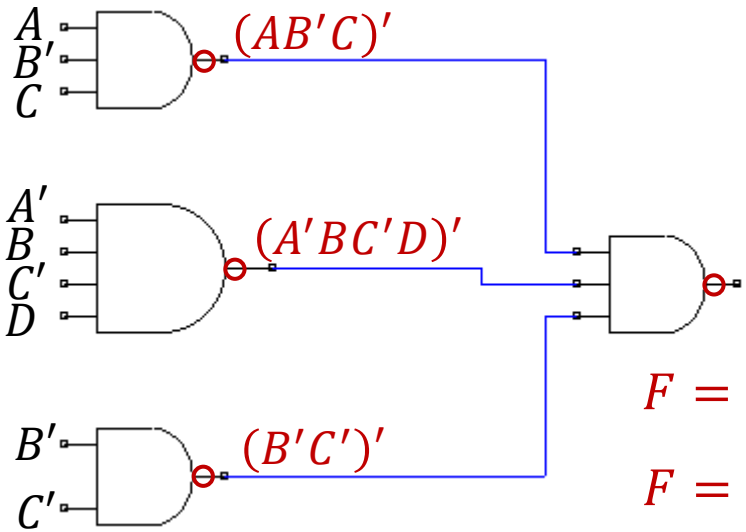


$\bar{X} + \bar{Y} + \bar{Z} = \overline{XYZ}$ DeMorgan's Theorem

This is another way to draw a 3-input NAND gate

Alternate Gate Symbols

NAND - NAND = AND - OR
in 2 level logic



$$F = \overline{\overline{AB'C}\ \overline{A'BC'D}\ \overline{B'C'}}$$

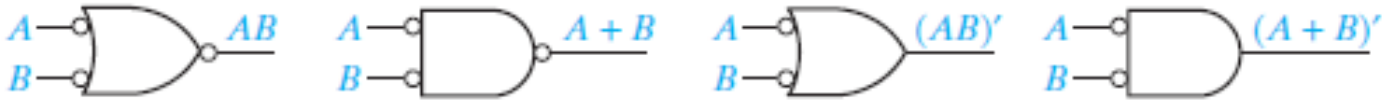
$$F = \overline{\overline{AB'C}} + \overline{\overline{A'BC'D}} + \overline{\overline{B'C'}}$$

$$F = AB'C + A'BC'D + B'C'$$

Two-level NAND Gate Logic

- Alternate symbol for inverters:

- Alternate gate symbols derived using DeMorgan's Laws



$AB = (A' + B')'$ (a) AND

$A + B = (A'B')'$ (b) OR

$(AB)' = A' + B'$ (c) NAND

$(A + B)' = A'B'$ (d) NOR

FIGURE 7-14
Alternative Gate Symbols

Two-Level Logic with NAND or NOR gates

Conversion of AND & OR Gate Circuits to NAND & NOR

- Two-level circuit composed of AND and OR gates easily converted to circuit composed of NAND gates or NOR gates.
- Conversion is carried out by using $F = (F')'$ and applying DeMorgan's laws

$$(X_1 + X_2 + \cdots + X_n)' = X_1'X_2' \ldots X_n' \quad (7\text{-}11)$$

$$(X_1X_2 \ldots X_n)' = X_1' + X_2' + \cdots + X_n' \quad (7\text{-}12)$$

- Will be referred to by equation number on next few slides
- Earlier this lecture we did conversion of AND-OR to NAND-NAND
- Will look at other forms now, via an example

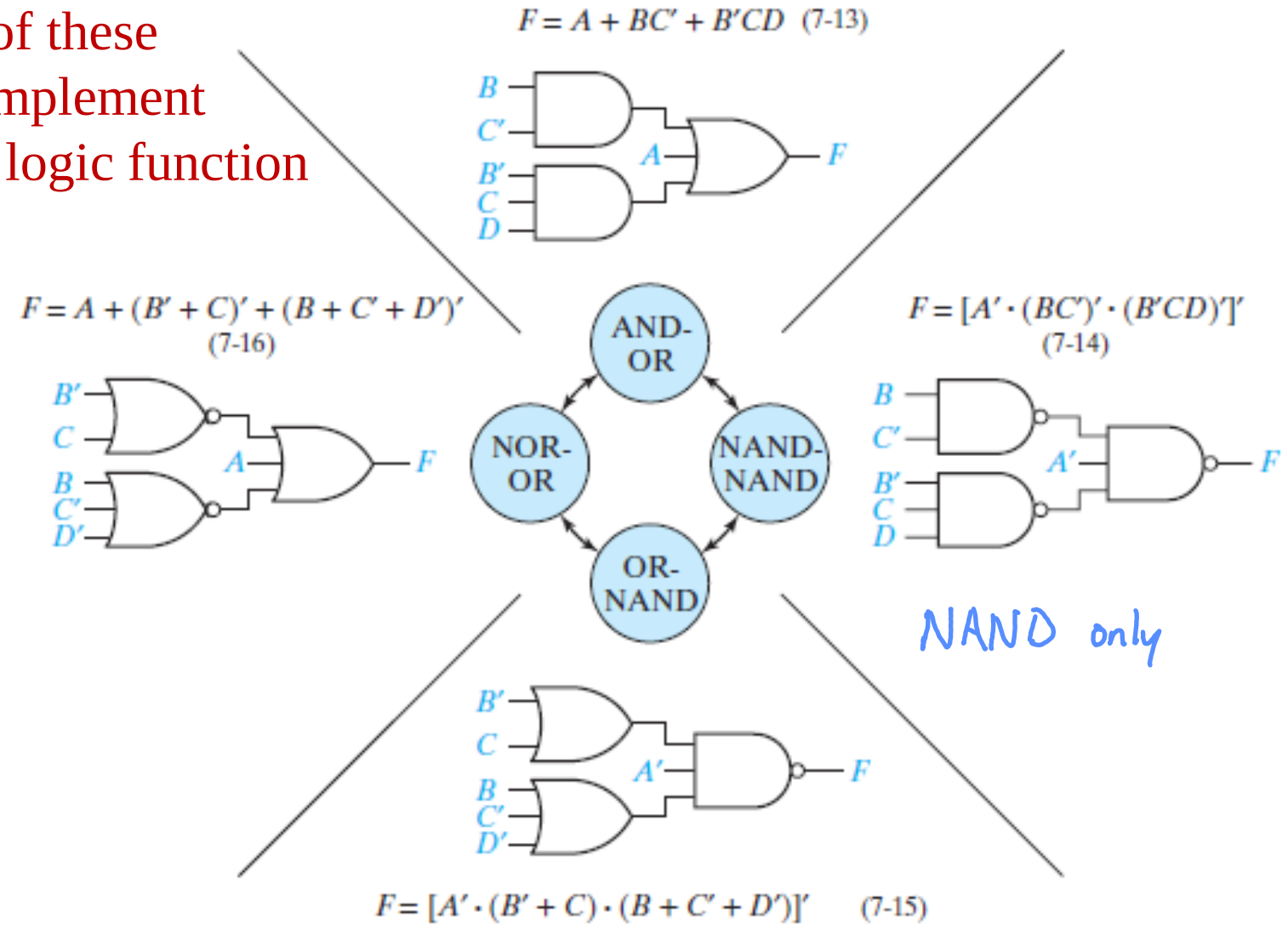
Two-Level Logic with NAND or NOR gates

- Function $F = A + BC' + B'CD$ is in SOP form (AND - OR)
- Earlier this lecture
  - straight forward to implement in NAND-NAND form
  - used a circuit conversion approach with "negation bubbles" on logic symbols
- Now: Mathematical look using $F = (F')'$ and DeMorgan's

$$F = A + BC' + B'CD = [(A + BC' + B'CD)']'$$
$$= [A' \cdot (BC')' \cdot (B'CD)']' \quad \text{(NAND - NAND)} \qquad \text{(by 7-11)}$$
$$= [A' \cdot (B' + C) \cdot (B + C' + D')]' \quad \text{(OR - NAND)} \qquad \text{(by 7-12)}$$
$$= A + (B' + C)' + (B + C' + D')' \quad \text{(NOR - OR)} \qquad \text{(by 7-12)}$$

Two-Level Logic with NAND or NOR gates

All four of these circuits implement the same logic function



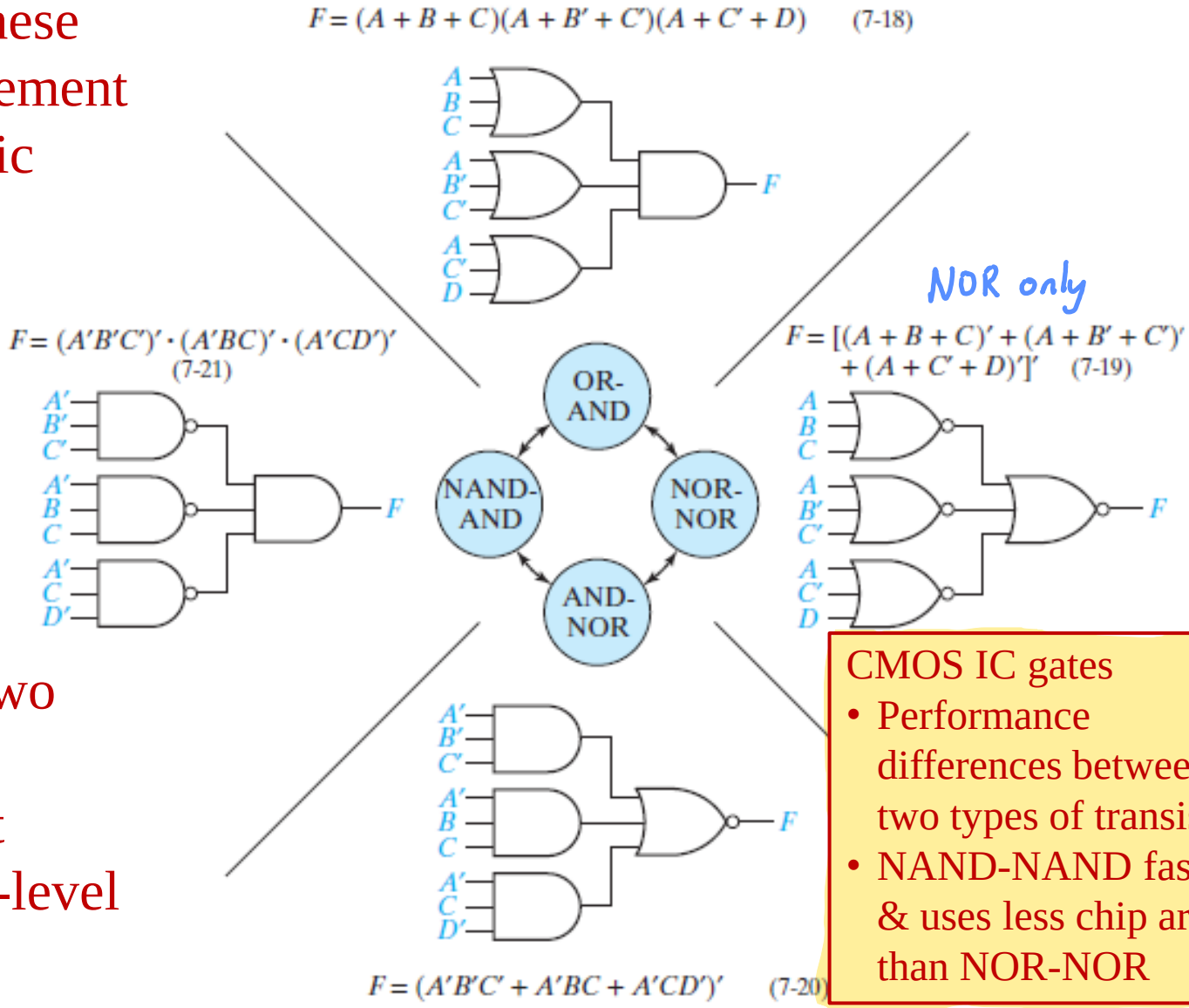
Two-Level Logic with NAND or NOR gates

- Function $F = A + BC' + B'CD$ is in SOP form (AND-OR)
- It can also be written in POS form:
  $F = (A + B + C)(A + B' + C')(A + C' + D)$ (OR-AND)
  - straight forward to implement in NOR-NOR form
  - Could use a circuit conversion approach with "negation bubbles" on logic symbols similar to what we did earlier this lecture
- But by math using $F = (F')'$ and DeMorgan's:

$$
\begin{aligned}
F &= (A + B + C)(A + B' + C')(A + C' + D) \\
&= \{[(A + B + C)(A + B' + C')(A + C' + D)]'\}' \\
&= [(A + B + C)' + (A + B' + C')' + (A + C' + D)']' && \text{(NOR-NOR)} \quad \text{(by 7-12)} \\
&= (A'B'C' + A'BC + A'CD')' && \text{(AND-NOR)} \quad \text{(by 7-11)} \\
&= (A'B'C')' \cdot (A'BC)' \cdot (A'CD')' && \text{(NAND-AND)} \quad \text{(by 7-11)}
\end{aligned}
$$

Two-Level Logic with NAND or NOR gates

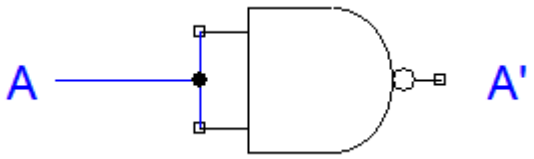
- All four of these circuits implement the same logic function
- And same function as two slides ago
- Overall eight possible two-level forms



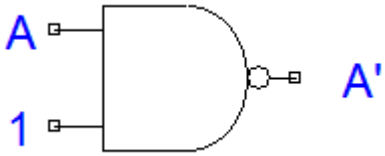
CMOS IC gates
- Performance differences between two types of transistors
- NAND-NAND faster & uses less chip area than NOR-NOR

NOT with NAND or NOR gates

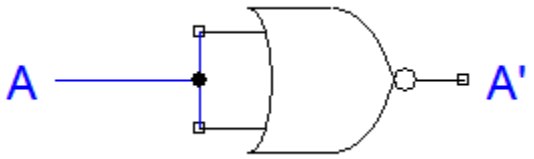
Previously saw (Lecture 7)



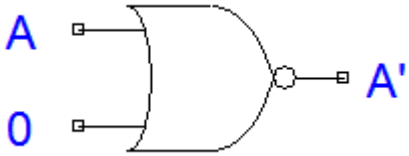
Better



A	1	$\overline{A1}$
0	1	1
1	1	0



Better



A	0	$\overline{A+0}$
0	0	1
1	0	0

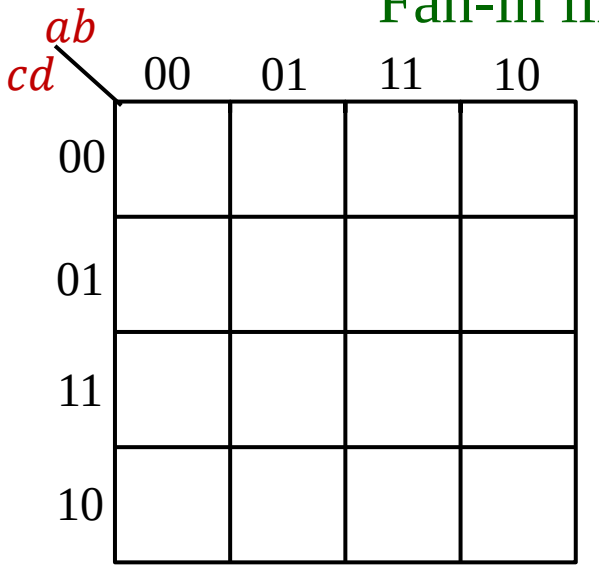
Fan-out: Number of gate inputs that may be attached to the output of a gate

- The top two attach 2 gate inputs to the output of the prior gate → slower switching
- The bottom two do not load the output of the prior gate as much → faster switching

Limited Fan-In Designs

- No "Always Successful" or Optimal Method for a most reduced design
- General Approach:
  - K-Map → Reduced SOP
  - Look for similar "parts" in the products (Double-use products)
  - Probably going to need more than two stages (No longer SOP)
- Example: $F = abc'd' + abc'd + ab'cd' + ab'cd + a'b'cd'$

  Fan-in limited to 2 inputs per gate



Limited Fan-In Designs

$$F = abc' + ab'c + b'cd'$$

Sketch brute force AND & OR

Not drawing all that

Just get his notes on canvas

4 level

8 gates

16 inputs

Limited Fan-In Designs

$F = abc' + ab'c + b'cd'$

$F = (ab)c' + b'c(a + d')$

Sketch AND & OR

Limited Fan-In Designs

$$F = abc' + ab'c + b'cd'$$

$$F = (ab)c' +$$

Sketch AND & OR

Limited Fan-In Designs

$$F = abc' + ab'c + b'cd'$$

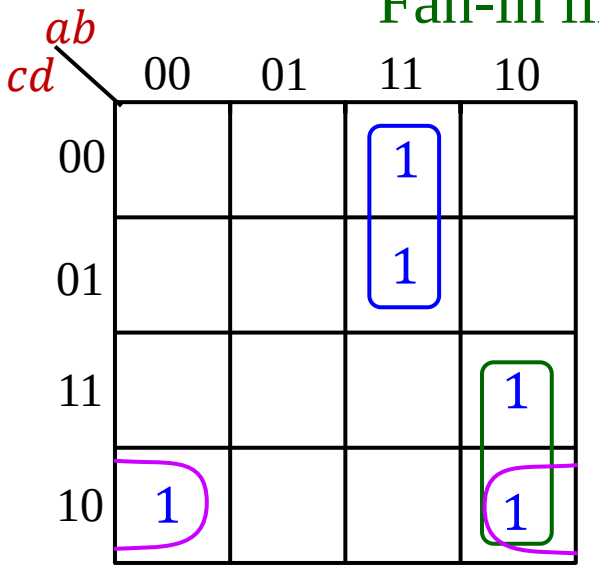
$$F = a[(bc') + \quad] \quad d'$$

Sketch AND & OR

Limited Fan-In Designs

- Example: $F = abc'd' + abc'd + ab'cd' + ab'cd + a'b'cd'$

  Fan-in limited to 2 inputs per gate



$F = abc' + ab'c + b'cd'$

Looked at four 2-NAND designs

1. Brute force: 8 gates, 4 levels
2. Boolean algebra v1: 6 gates, 3 levels
3. Boolean algebra v2: 6 gates, 4 levels
4. Boolean algebra v3: 6 gates, 4 levels

Limited Fan-In Designs

$F = abc' + ab'c + b'cd'$

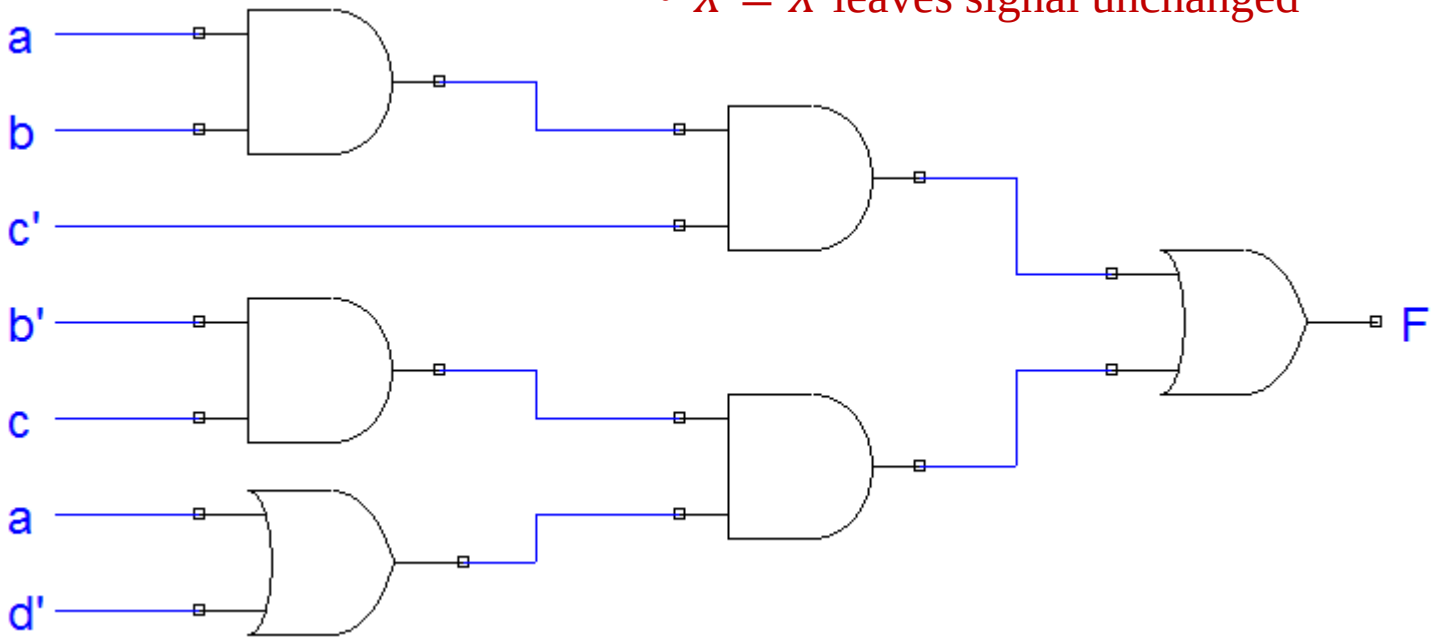
Both of these are NAND →

$F = (ab)c' + b'c(a + d')$

Sketch NAND version

- Graphical approach: inversion bubbles in pairs on wires
- $\bar{\bar{X}} = X$ leaves signal unchanged

CAD redraw of circuit #2



Graphical Circuit Conversion to NAND

Summaries of the procedures applied on the previous slide

1. Convert all AND gates to NAND gates by adding an inversion bubble at output
2. Convert all OR gates to NAND gates by adding inversion bubbles at inputs
3. Whenever an inverted output drives an inverted input, no further action is needed since the two inversions cancel
4. Whenever a non-inverted gate output drives an inverted gate input or vice versa, insert an inverter so that the bubbles will cancel. (To make bubbles cancel, choose an inverter with the bubble at the input or output.)

5. Whenever an input variable drives an inverted input, complement the variable (or add an inverter) so the complementation cancels the inversion at the input

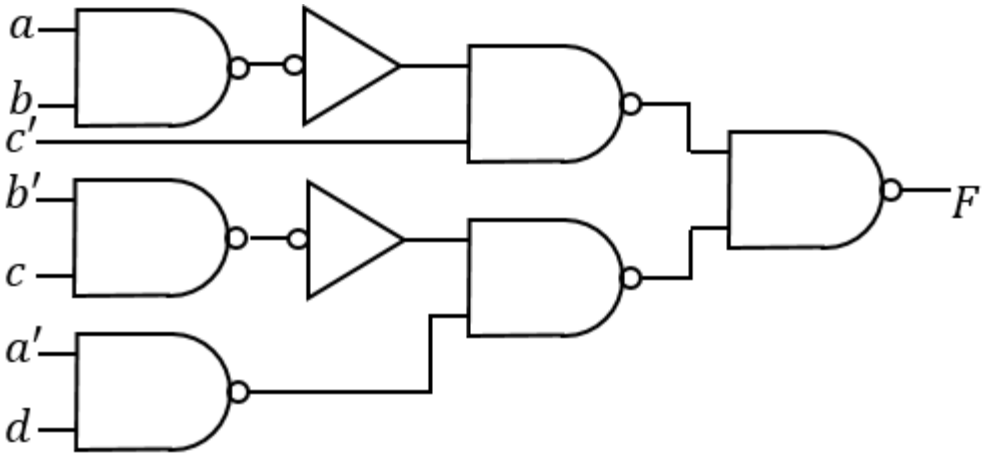
Limited Fan-In Designs

$F = abc' + ab'c + b'cd'$ Both of these are NAND →

$F = (ab)c' + b'c(a + d')$ Sketch NAND version

Redrawn with regular NAND symbols and showing Inverters (NOT)

Inserted NOT so that gate is NAND



NAND
Inverters
total gates

levels counting inverters

Limited Fan-In Designs

$$F = abc' + ab'c + b'cd'$$

$$F = (ab)c' + b'c(a + d')$$

Sketch NAND version

Algebraic approach using DeMorgan's

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

$$F = \overline{\left[\overline{(ab)\bar{c}}\right]\left[\overline{(\bar{b}c)(a + \bar{d})}\right]}$$

Last and 2nd last levels NAND form
Stuff inside:

$$F = \overline{\left[\overline{(ab)\bar{c}}\right]\left[\overline{(\bar{b}c)\overline{\overline{a + \bar{d}}}}\right]}$$

$$F = \overline{\left[\overline{(ab)\bar{c}}\right]\left[\overline{(\bar{b}c)(\bar{a}d)}\right]}$$

$$F = \overline{\left[\overline{\overline{(ab)\bar{c}}}\right]\left[\overline{\overline{(\bar{b}c)(\bar{a}d)}}\right]}$$