Lecture Outline

Reminders to self:

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

- Last Lecture
  - Reviewed Participation Quiz 5
  - Continued multi-level logic
    - NAND-NAND 2-level logic (and others)
    - Started Limited Fan-in Design
- Today's Lecture
  - Finish Limited Fan-in Design
  - Multiple Output Logic Design

Handouts and Announcements

- Announcements
  - Homework Problem 7-3
    - Posted on Carmen yesterday evening (2/7)
    - Due: 11:25am Monday 2/13
  - Homework Reminders: HW 7-1 and 7-2
    - Posted on Carmen Saturday (2/4)
    - Due: 11:59pm Thursday 2/9
  - Read for Friday: pages 229-240

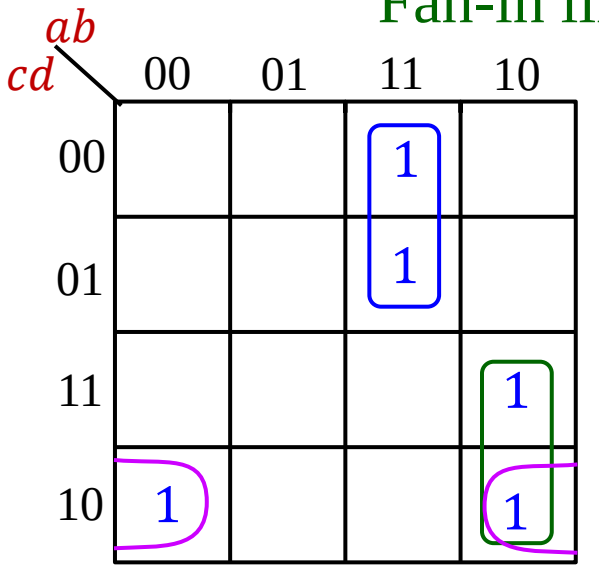
Handouts and Announcements

- Announcements
  - ECE 2060 Laboratories
    - Start next week
    - You must attend your scheduled lab session
    - There are videos you are required to view prior to attending your first lab session
    - Refer to the Carmen pages for your lab section for details
    - Lab questions should be referred to Prof. Chapman and/or the GTA for your lab section

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



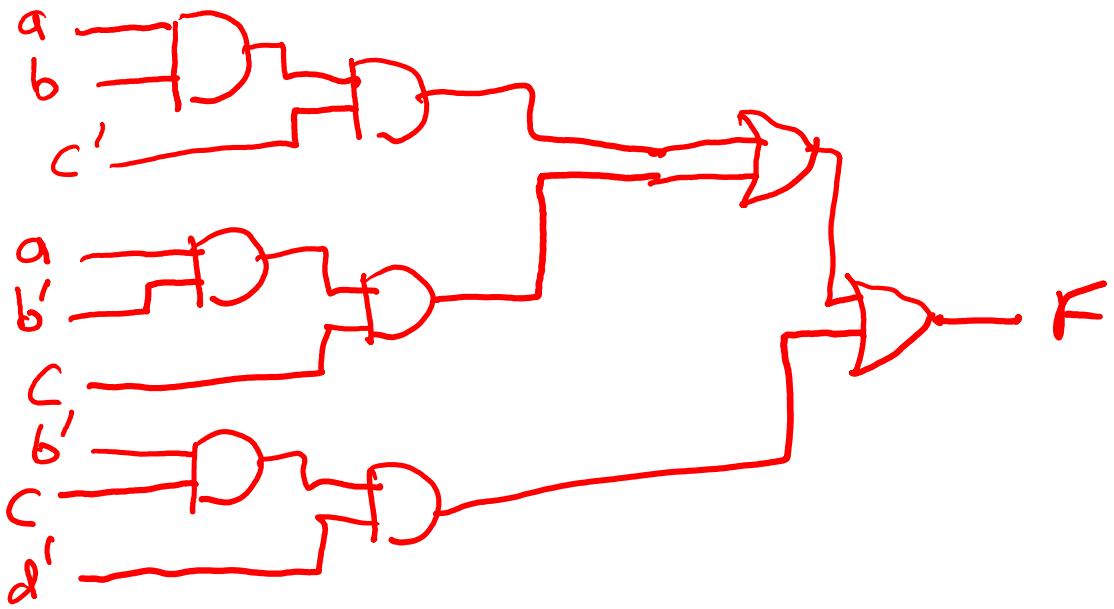
$F = abc' + ab'c + b'cd'$

Review problem we were working on and where we left off

Limited Fan-In Designs

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

Sketch brute force AND & OR



4-level
8 gates
16 inputs

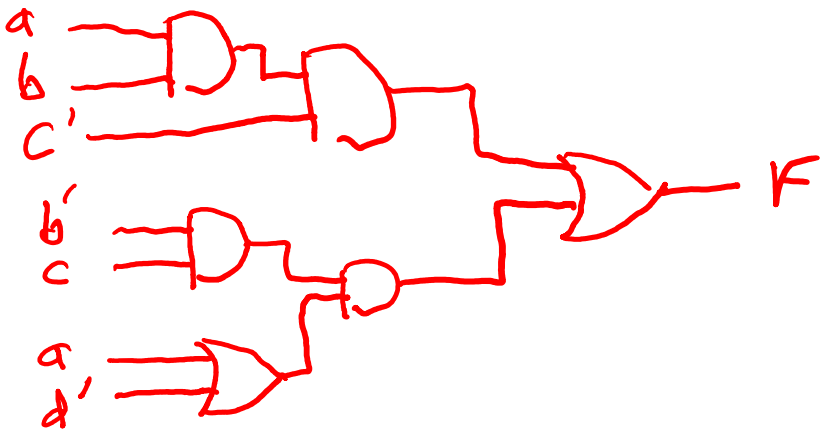
Review problem we were working on and where we left off

Limited Fan-In Designs

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

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

Sketch AND & OR

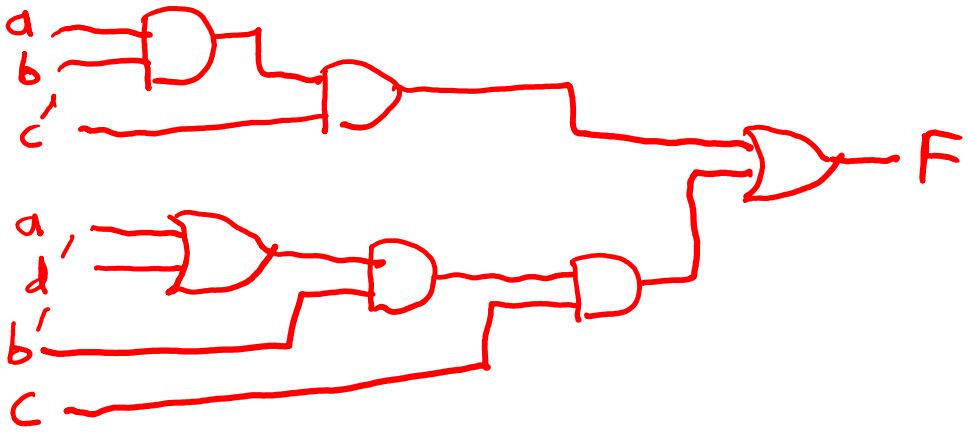


3 levels
6 gates
12 inputs

Limited Fan-In Designs

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

$$F = (ab)c' + c[b'(a + d')]$$ Sketch AND & OR

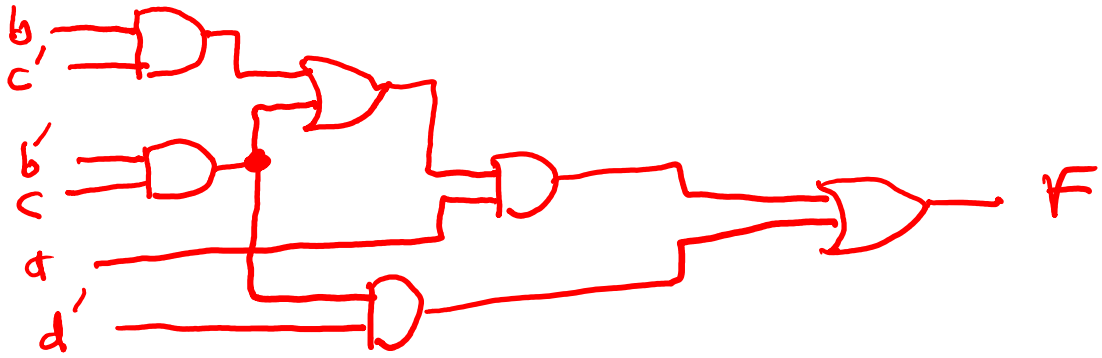


4 levels
6 gates
12 inputs

Limited Fan-In Designs

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

$$F = a[(bc') + (b'c)] + (b'c)d'$$ Sketch AND & OR

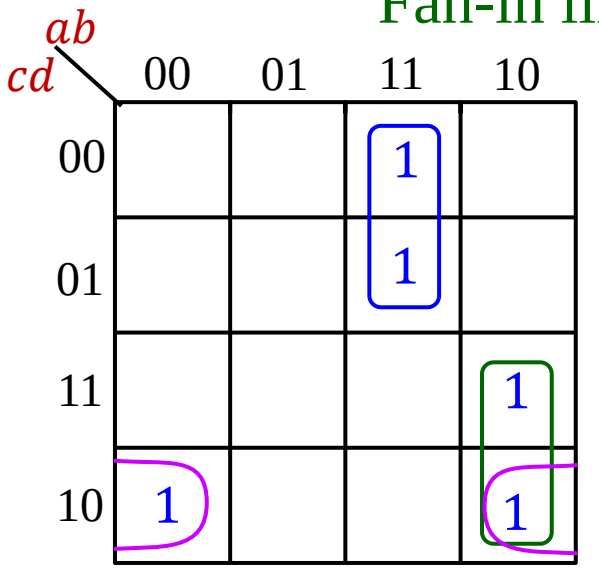


4 levels
6 gates
12 inputs

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 ← best
3. Boolean algebra v2: 6 gates, 4 levels
4. Boolean algebra v3: 6 gates, 4 levels

Limited Fan-In Designs

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

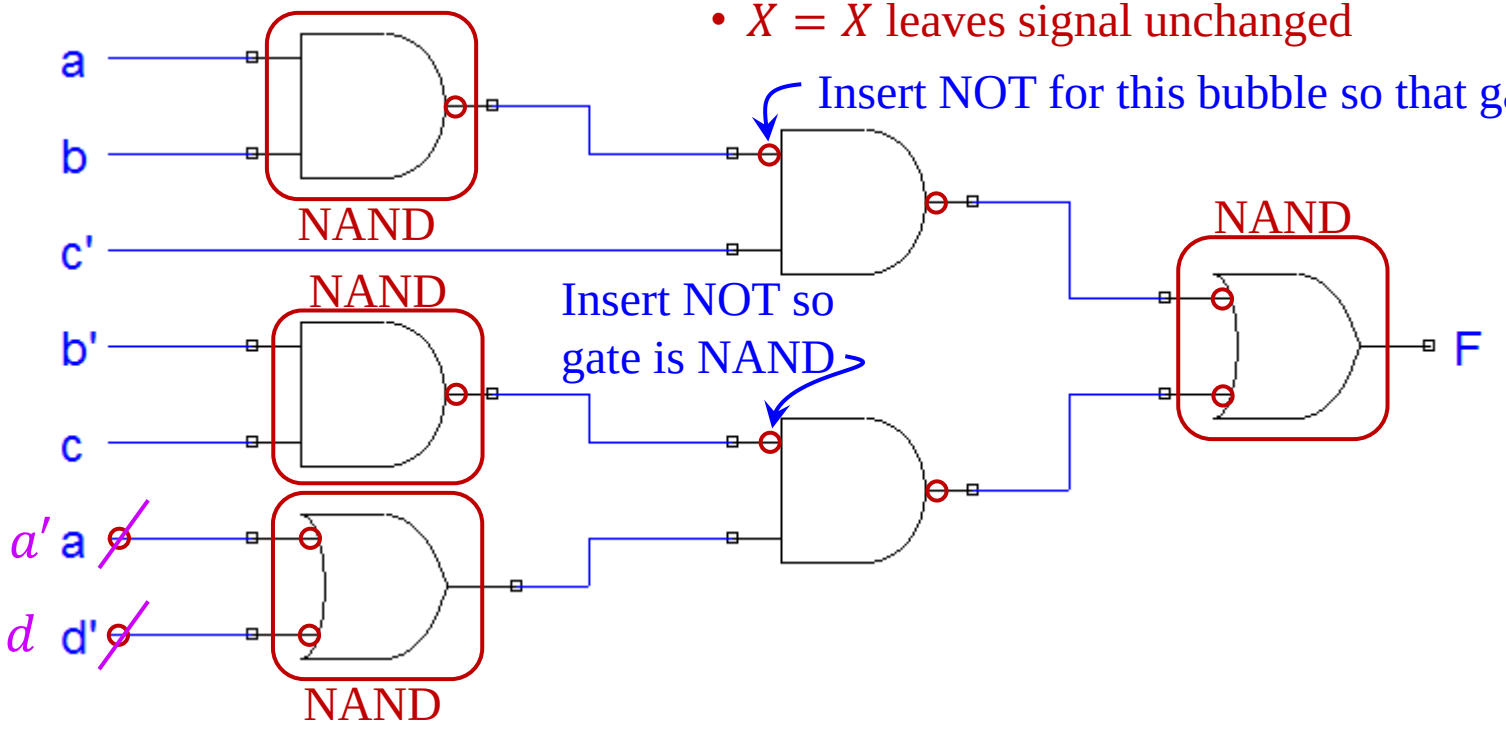
Both of these are NAND →

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

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

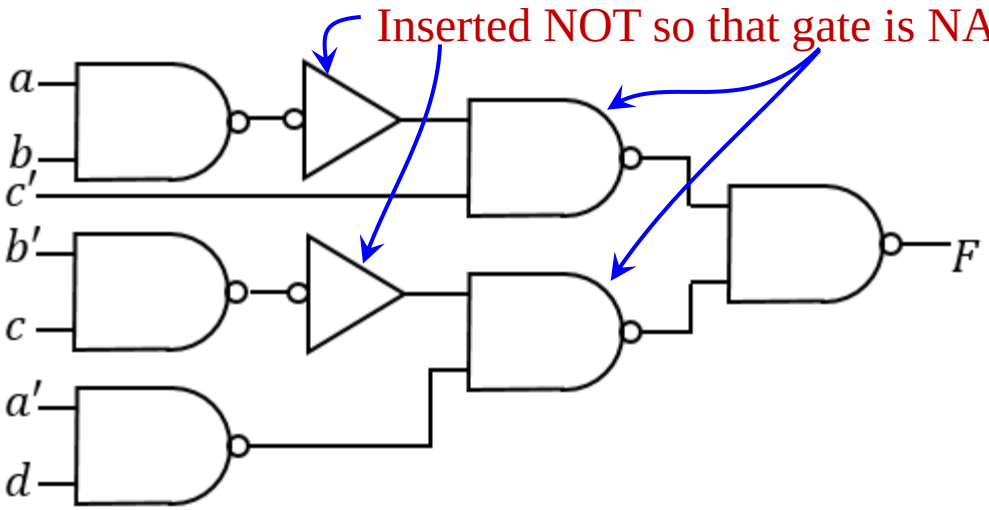
Limited Fan-In Designs

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

Both of these are NAND →

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

Redrawn with regular NAND symbols and showing Inverters (NOT)



6 NAND
2 Inverters
8 total gates

4 levels counting inverters

14 inputs

Limited Fan-In Designs

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

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

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: still AND/OR

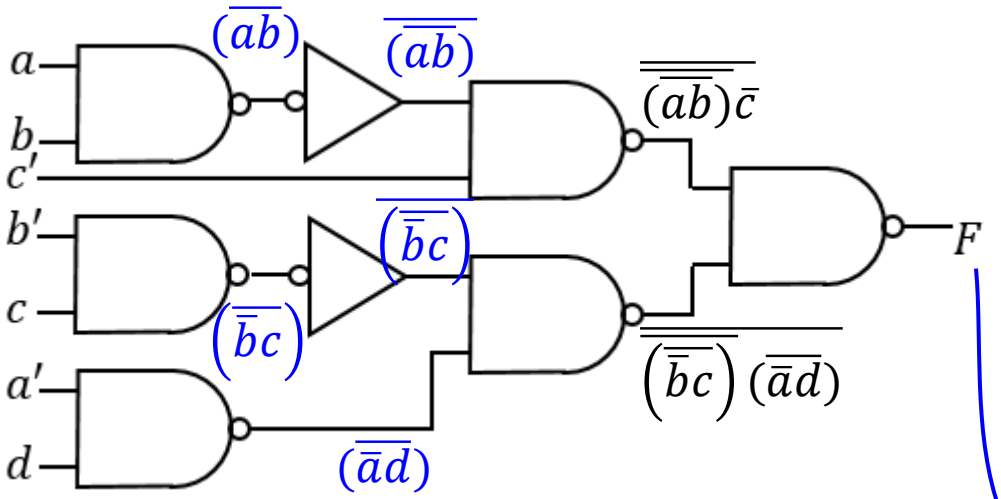
$$F = \overline{\left[\overline{(ab)\bar{c}}\right]\left[(\bar{b}c)\overline{\overline{(a + \bar{d})}}\right]}$$

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

NAND

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

All NAND/NOT



$$\overline{\text{AND}} = \text{NAND}$$

$$\overline{\overline{\text{AND}}} = \overline{\text{NAND}}$$

Multiple Outputs

- Digital design problems often require realization of circuits to generate more than one function from the same set of inputs
- Each output could be designed separately using already learned techniques
- But use of some gates in common between two or more functions may yield a more economical circuit

Multiple Outputs

Example: Three separate K-maps were used to find the following reduced functions for three outputs from four input variables

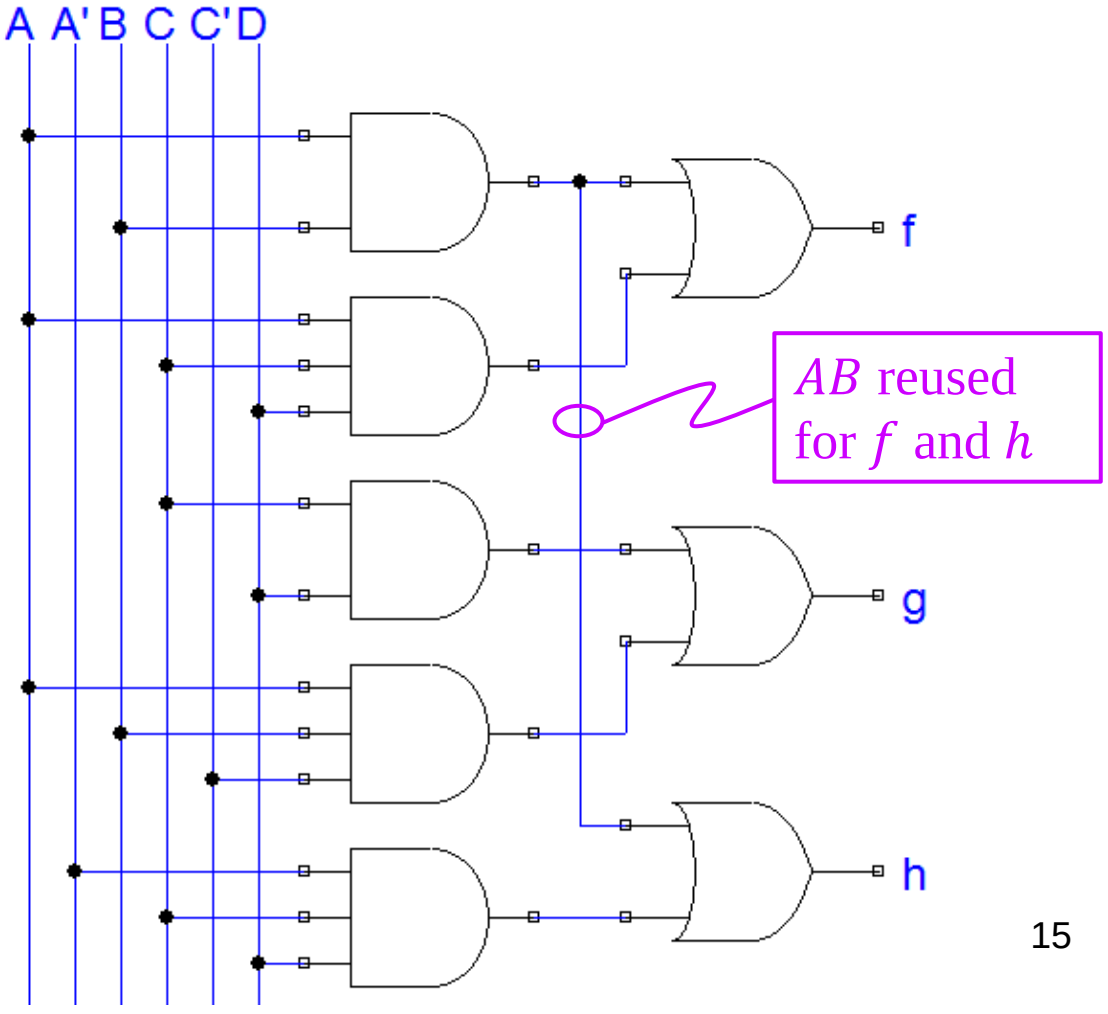
$f = \boxed{AB} + ACD$

$g = CD + ABC'$

$h = \boxed{AB} + A'CD$

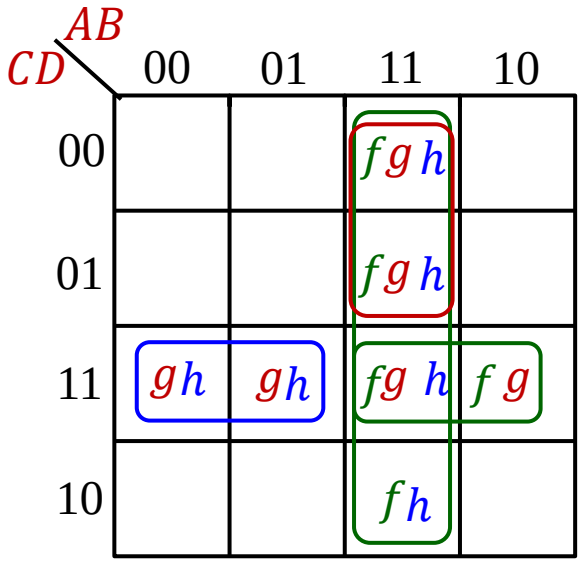
8 gates (5 products, 3 sums)
19 inputs

But it is possible to be more economical, by sharing products that are not minimal



Multiple Outputs

$f = AB + ACD$ $g = CD + ABC'$ $h = AB + A'CD$



f	g	h
	ABC'	
ACD	ACD	
	$A'CD$	$A'CD$
AB		AB

$f = AB + ACD$ $g = ACD + ABC' + A'CD$ $h = AB + A'CD$

7 gates (4 products, 3 sums) - one fewer gate
18 inputs - one fewer input

Multiple Outputs

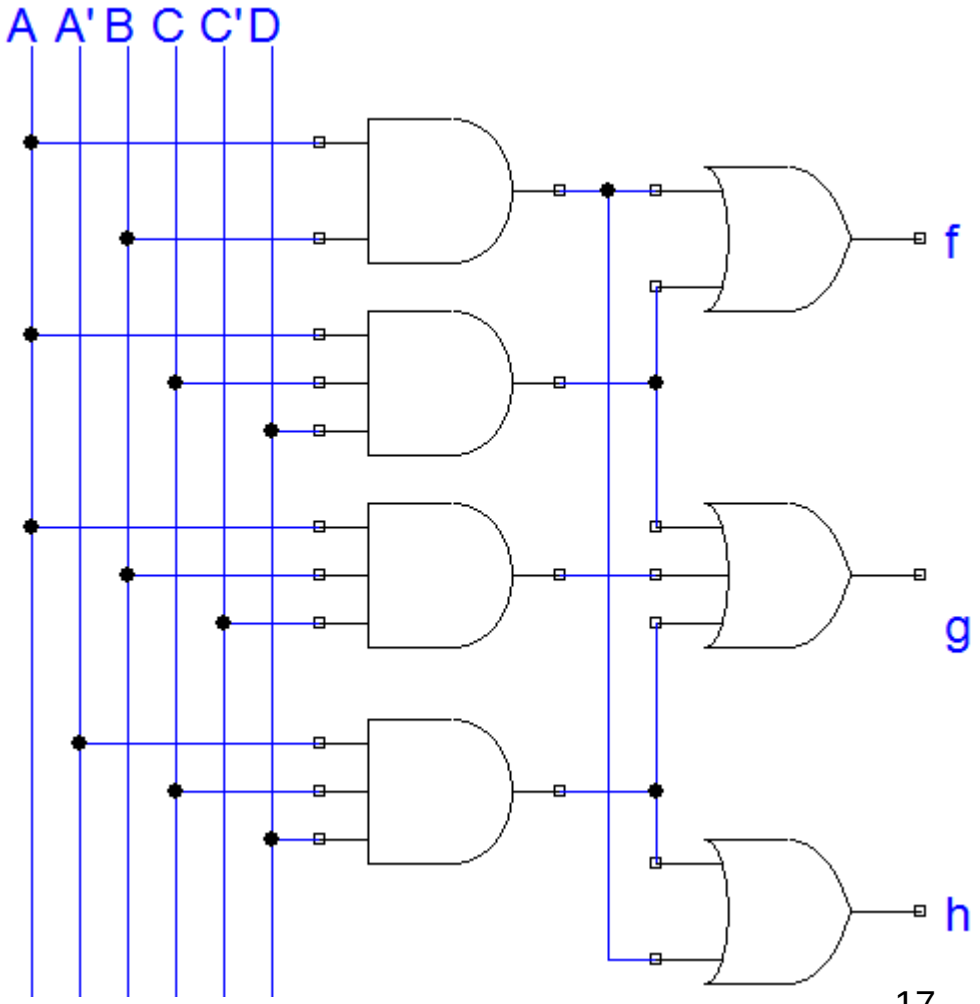
$f = AB + ACD$ $g = ACD + ABC' + A'CD$ $h = AB + A'CD$

7 gates (4 products, 3 sums)
18 inputs

- Three-input OR for *g*
- With re-use of two 3-input ANDs
- Allowed elimination of one AND gate

When designing multiple output circuits

1. First try to minimize number of gates
2. If more than one solution have same number of gates, choose one with fewest inputs

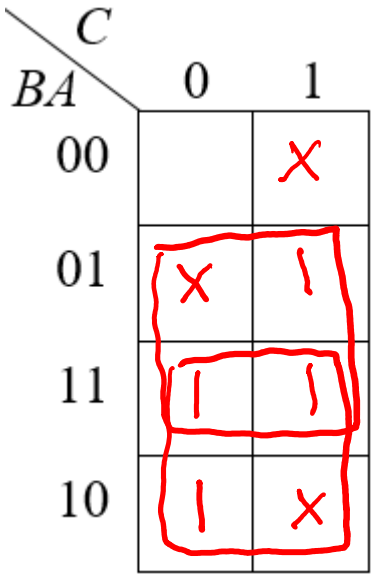


Multiple Outputs

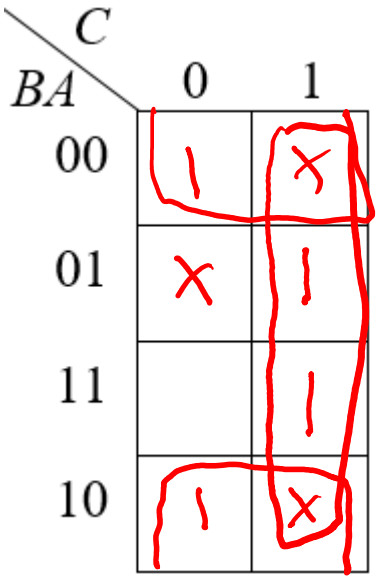
Example from final exam from autumn semester 2019

- The problem was design of a 3-bit counter (topic for later this semester)
- A large part of problem was developing this truth table (later this semester)
- Remainder of problem was designing circuit to produce T_C, T_B, and T_A with fewest number of gates

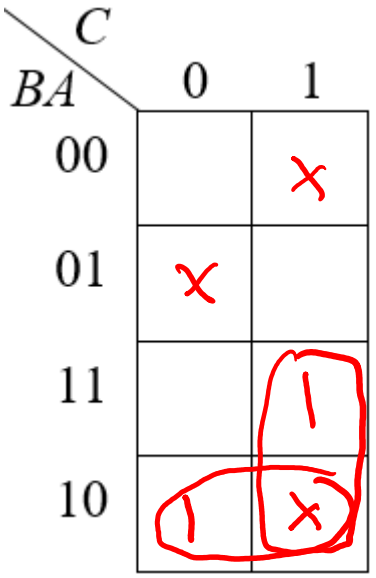
C	B	A	T_C	T_B	T_A
0	0	0	0	1	0
0	0	1	X	X	X
0	1	0	1	1	1
0	1	1	1	0	0
1	0	0	X	X	X
1	0	1	1	1	0
1	1	0	X	X	X
1	1	1	1	1	1



$T_C = B + A$



$T_B = C + A'$

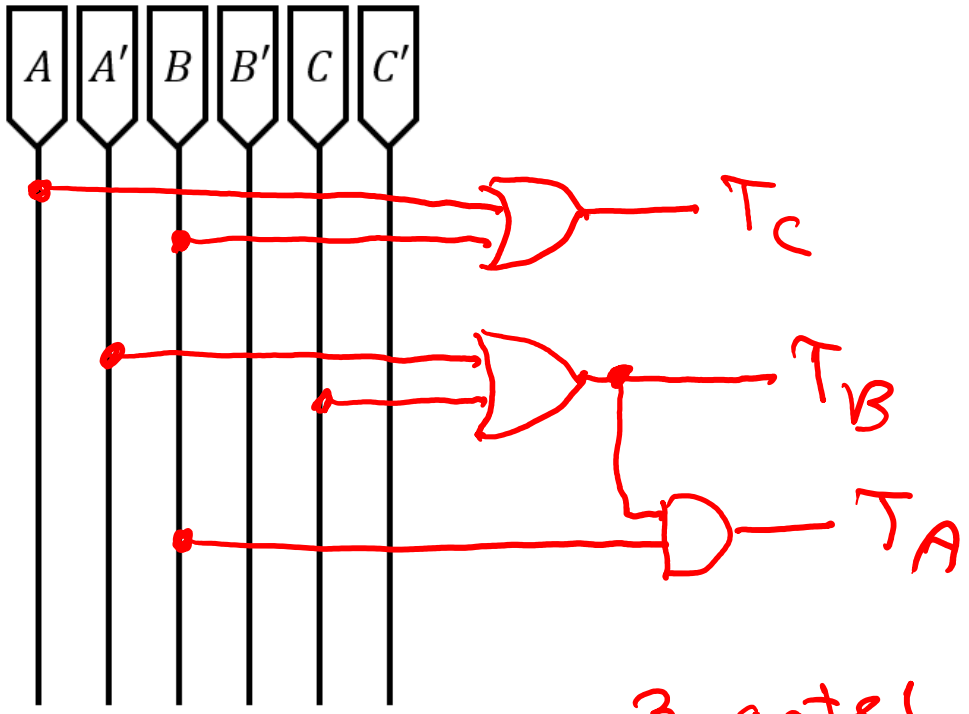


$T_A = A'B + BC$

Multiple Outputs

ECE2060

C	B	A	T_C	T_B	T_A
0	0	0	0	1	0
0	0	1	X	X	X
0	1	0	1	1	1
0	1	1	1	0	0
1	0	0	X	X	X
1	0	1	1	1	0
1	1	0	X	X	X
1	1	1	1	1	1



3 gates
6 inputs

$$T_C = A + B$$

$$T_B = A' + C$$

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

5 gates, 10 inputs

Essential Prime Implicants
Single Output

Procedure

1. Choose a minterm (a 1 – not an X) that has not yet been covered
2. Find all 1's and X's adjacent to that minterm (Check the n adjacent squares on an n-variable K-map)
3. If a single prime implicant covers the minterm and all adjacent 1's and X's then that term is an essential prime implicant → include it
4. Repeat steps 1, 2, and 3 until all essential prime implicants have been included
5. For the remaining 1's
   - Find a minimum set of prime implicants that covers them
   - If there is more than one such set, choose a set with a minimum number of literals (minimize number of gate inputs)