# Karnaugh maps

Prof. Usagi

# How many "OR"s?

- For the truth table shown on the right, what's the minimum number of "OR" gates we need?

  A. 1
  B. 2
  C. 3
  D. 4
  E. 5

F(A, B, C) =

A′B′C′ + A′B′C + A′BC′ + A′BC + AB′C′ + ABC′

= A′B′(C′+C) + A′B(C′+C) + AC′(B′+B)

= A′B′ + A′B + AC′

= A′ + AC′ = A′(1+C′)+AC′ **Distributive Laws**

= A′ + A′C′ + AC′

= A′ + (A+A)C′

= A′ + C′

**How can I know this!!!**

| Input | | | Output |
|---|---|---|---|
| A | B | C | |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

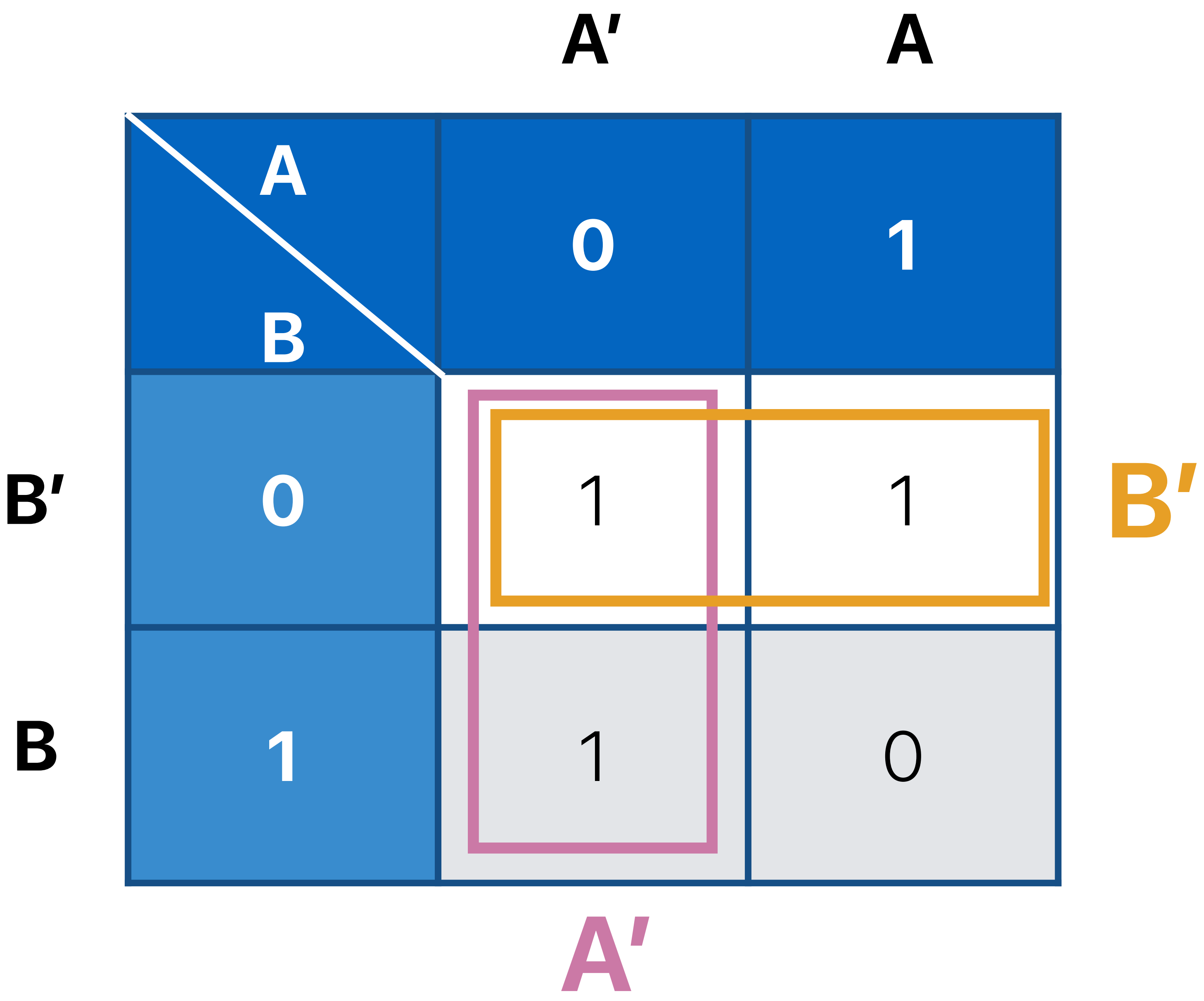
# Karnaugh maps

# Karnaugh maps

- Alternative to truth-tables to help visualize adjacencies
- Guide to applying the uniting theorem
- ON-set elements with only one variable changing value are adjacent unlike the linear truth-table

# 2-variable K-map example

| Input | | Output |
|---|---|---|
| A | B | |
| 0 | 0 | **1** |
| 0 | 1 | **1** |
| 1 | 0 | **1** |
| 1 | 1 | **0** |



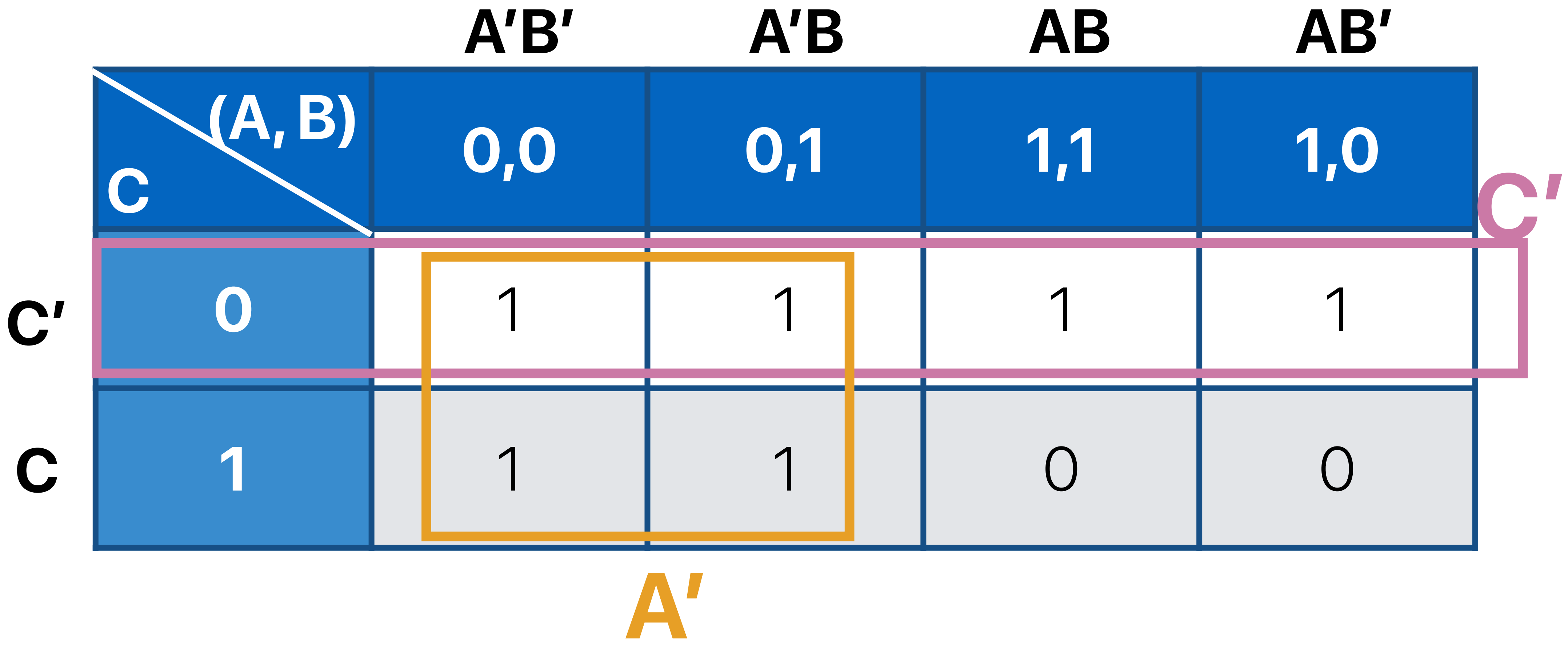
| | | | |
|---|---|---|---|
| | | A′ | A |
| | A \ B | 0 | 1 |
| B′ | 0 | 1 | 1 |
| B | 1 | 1 | 0 |

$$F(A, B) = A' + B'$$

# 3-variable K-map?

- Reduce to 2-variable K-map — 1 dimension will represent two variables
- Adjacent points should differ by only 1 bit
  - So we only change one variable in the neighboring column
  - 00, 01, 11, 10 — such numbering scheme is so-called **Gray–code**

| Input | | | Output |
|---|---|---|---|
| **A** | **B** | **C** | |
| 0 | 0 | 0 | **1** |
| 0 | 0 | 1 | **1** |
| 0 | 1 | 0 | **1** |
| 0 | 1 | 1 | **1** |
| 1 | 0 | 0 | **1** |
| 1 | 0 | 1 | **0** |
| 1 | 1 | 0 | **1** |
| 1 | 1 | 1 | **0** |

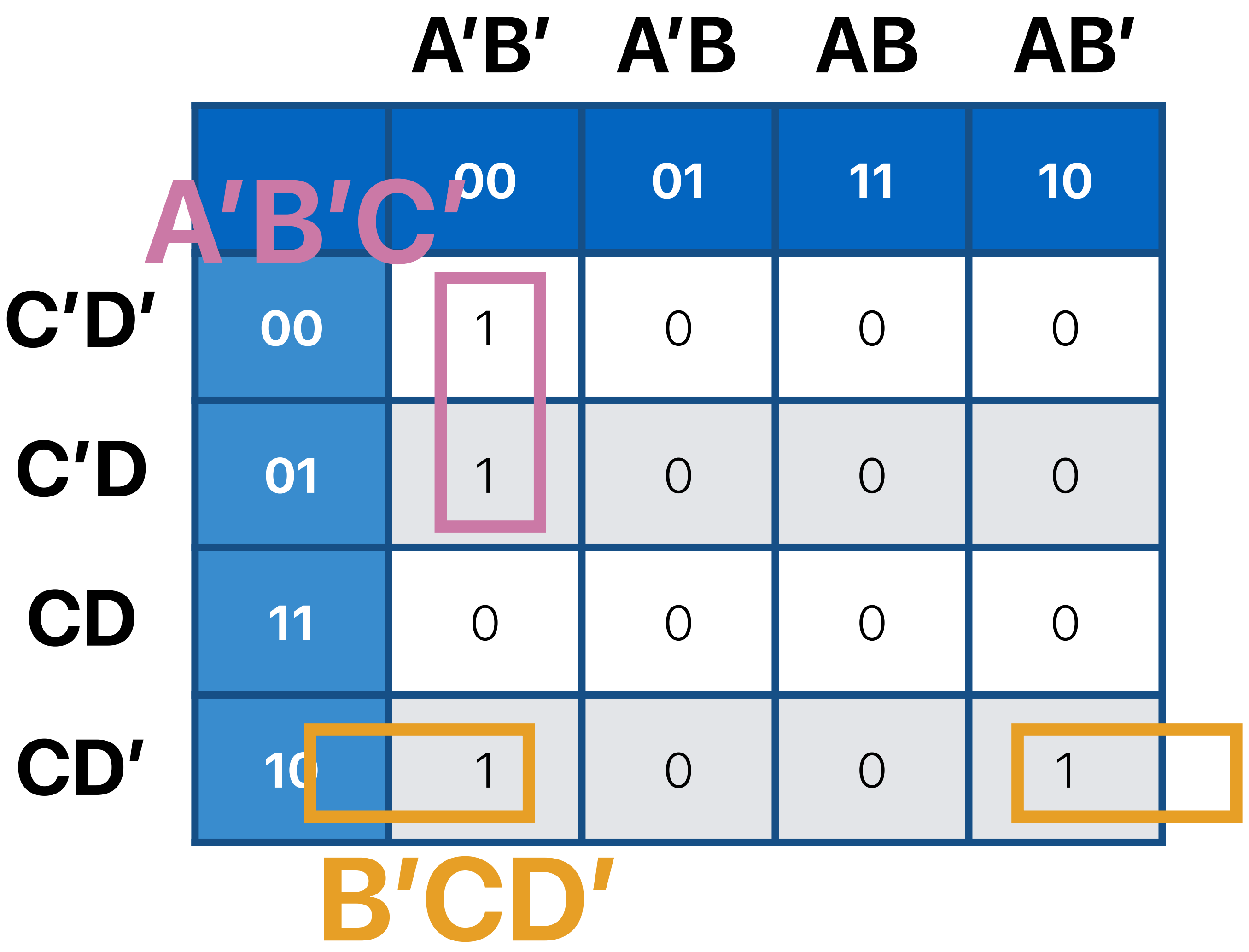


| | | **A′B′** | **A′B** | **AB** | **AB′** |
|---|---|---|---|---|---|
| | **(A, B)**<br>**C** | **0,0** | **0,1** | **1,1** | **1,0** |
| **C′** | **0** | 1 | 1 | 1 | 1 |
| **C** | **1** | 1 | 1 | 0 | 0 |

**$F(A, B, C) = A' + C'$**

# 4-variable K-map

- Reduce to 2-variable K-map — both dimensions will represent two variables
- Adjacent points should differ by only 1 bit
  - So we only change one variable in the neighboring column
  - Use Gray-coding — 00, 01, 11, 10

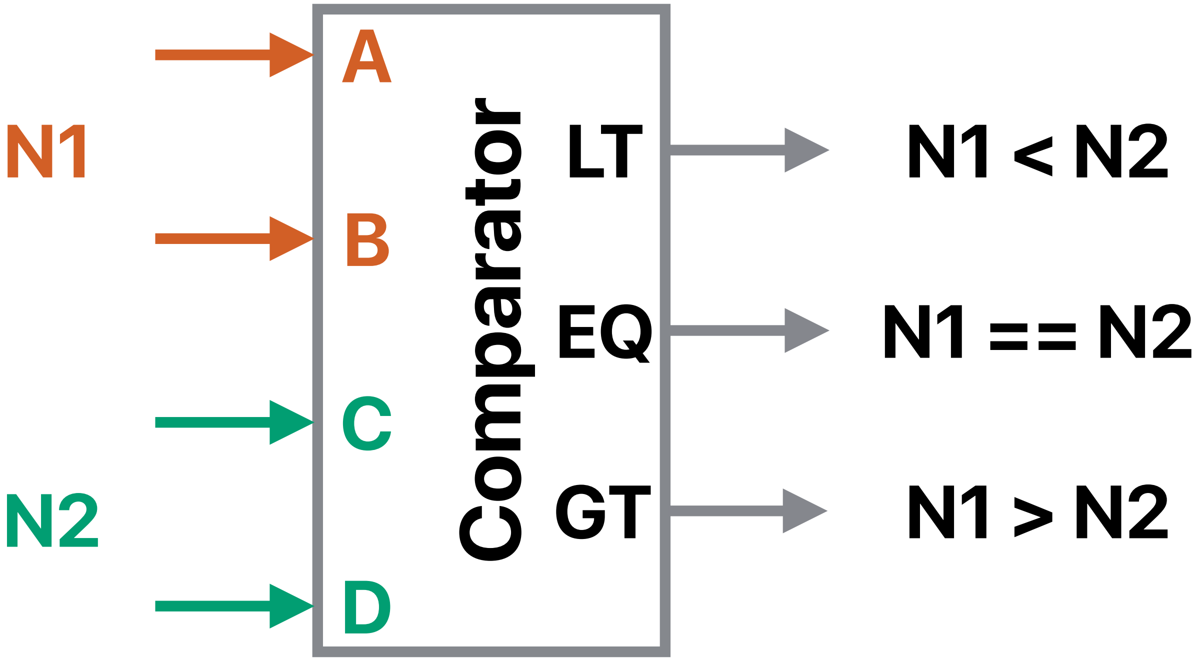


| | | A′B′ | A′B | AB | AB′ |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| C′D′ | 00 | 1 | 0 | 0 | 0 |
| C′D | 01 | 1 | 0 | 0 | 0 |
| CD | 11 | 0 | 0 | 0 | 0 |
| CD′ | 10 | 1 | 0 | 0 | 1 |

F(A, B, C) = A′B′C′+B′CD′

# Design Example: 2bit comparator

# Two-bit comparator



N1

A

B

N2

C

D

Comparator

LT → N1 < N2

EQ → N1 == N2

GT → N1 > N2

| Input | | | | Output | | |
|---|---|---|---|---|---|---|
| **A** | **B** | **C** | **D** | **LT** | **EQ** | **GT** |
| 0 | 0 | 0 | 0 | **0** | **1** | **0** |
| 0 | 0 | 0 | 1 | **1** | **0** | **0** |
| 0 | 0 | 1 | 0 | **1** | **0** | **0** |
| 0 | 0 | 1 | 1 | **1** | **0** | **0** |
| 0 | 1 | 0 | 0 | **0** | **0** | **1** |
| 0 | 1 | 0 | 1 | **0** | **1** | **0** |
| 0 | 1 | 1 | 0 | **1** | **0** | **0** |
| 0 | 1 | 1 | 1 | **1** | **0** | **0** |
| 1 | 0 | 0 | 0 | **0** | **0** | **1** |
| 1 | 0 | 0 | 1 | **0** | **0** | **1** |
| 1 | 0 | 1 | 0 | **0** | **1** | **0** |
| 1 | 0 | 1 | 1 | **1** | **0** | **0** |
| 1 | 1 | 0 | 0 | **0** | **0** | **1** |
| 1 | 1 | 0 | 1 | **0** | **0** | **1** |
| 1 | 1 | 1 | 0 | **0** | **0** | **1** |
| 1 | 1 | 1 | 1 | **0** | **1** | **0** |

We'll need a 4-variable Karnaugh map for each of the 3 output functions

# Don’t cares!

# Incompletely Specified Functions

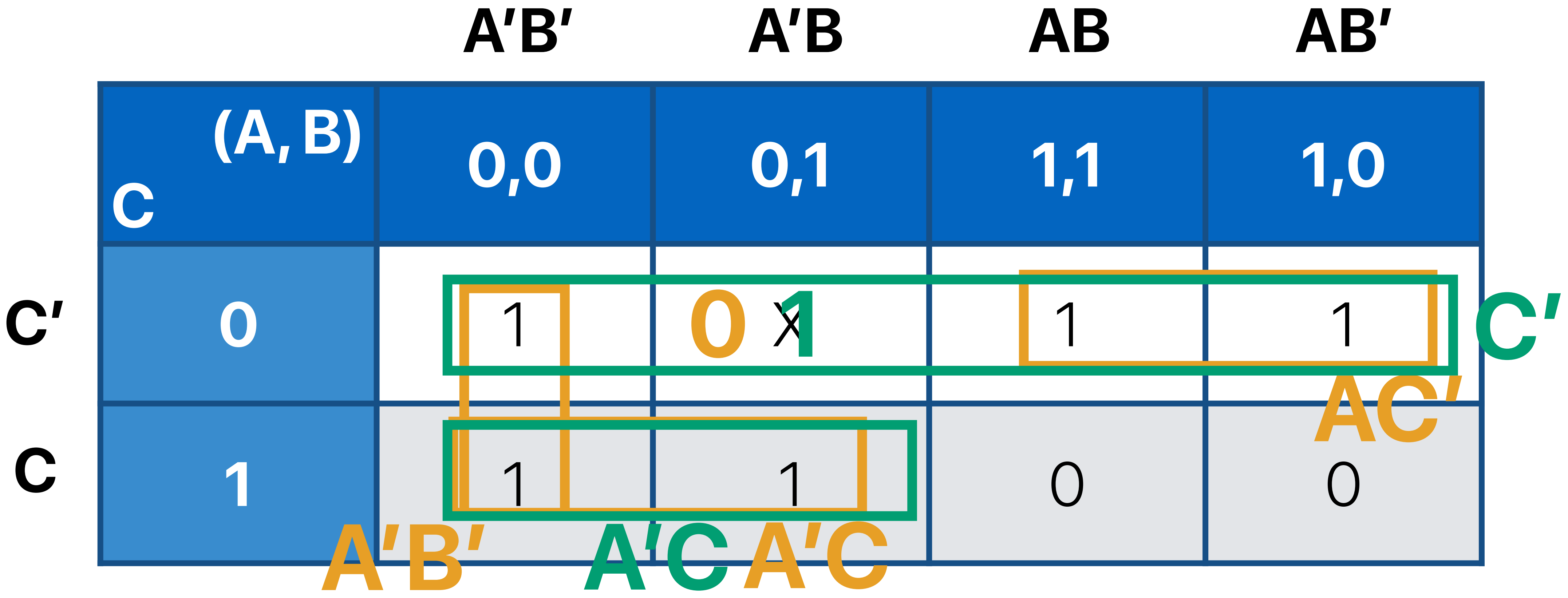
- Situations where the output of a function can be either 0 or 1 for a particular combination of inputs
- This is specified by a don’t care in the truth table
- This happens when
  - The input does not occur. e.g. Decimal numbers 0… 9 use 4 bits, so (1,1,1,1) does not occur.
  - The input may happen but we don’t care about the output. E.g. The output driving a seven segment display – we don’t care about illegal inputs (greater than 9)

**Don’t care**

| **A**<br>**B** | **0** | **1** |
|---|---|---|
| **0** | 0 | 0 |
| **1** | 1 | **X** |

# K-Map with “Don′t Care”s

You can treat “X” as either 0 or 1 — depending on which is more advantageous



| | | | | | |
|---|---|---|---|---|---|
| | | A′B′ | A′B | AB | AB′ |
| | (A, B)<br>C | 0,0 | 0,1 | 1,1 | 1,0 |
| C′ | 0 | 1 | X | 1 | 1 |
| C | 1 | 1 | 1 | 0 | 0 |

If we treat the “X” as 0?

F(A,B,C)=A′B′+A′C+AC′

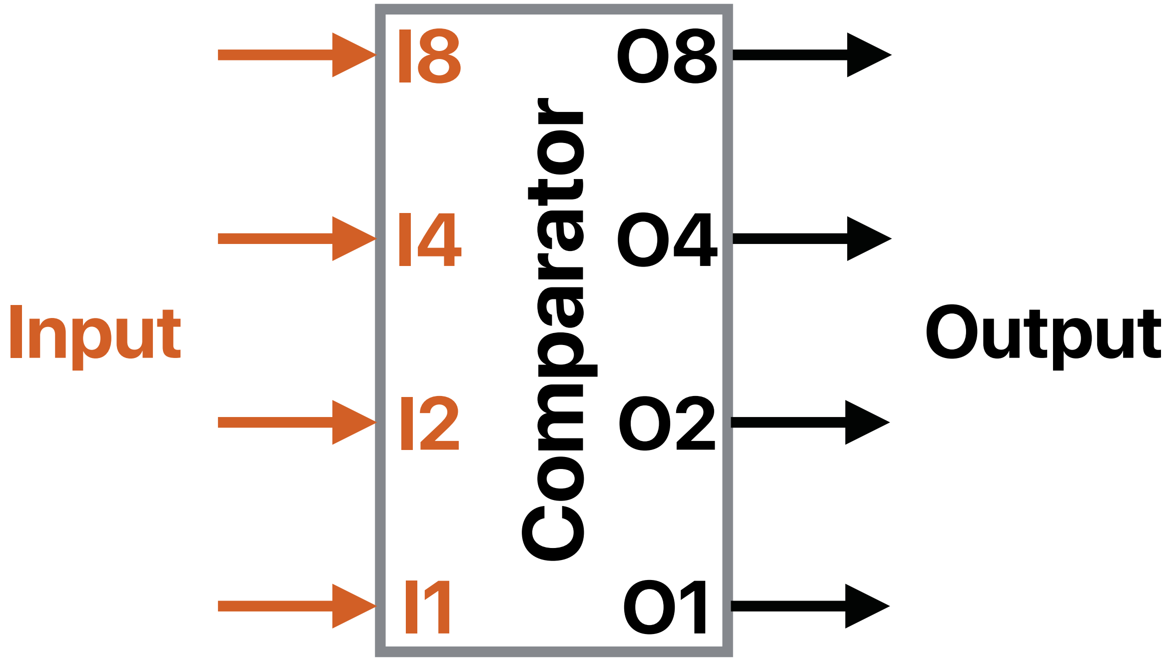
If we treat the “X” as 1?

F(A,B,C) = C′ + A′C

# Design examples — BCD + 1

# BCD+1 — Binary coded decimal + 1

- 0x0 — 1
- 0x1 — 2
- 0x2 — 3
- 0x3 — 4
- 0x4 — 5
- 0x5 — 6
- 0x6 — 7
- 0x7 — 8
- 0x8 — 9
- 0x9 — 0
- 0xA — 0xF — Don’t care



| Input | | | | Output | | | |
|---|---|---|---|---|---|---|---|
| **I8** | **I4** | **I2** | **I1** | **O8** | **O4** | **O2** | **O1** |
| 0 | 0 | 0 | 0 | **0** | **0** | **0** | **1** |
| 0 | 0 | 0 | 1 | **0** | **0** | **1** | **0** |
| 0 | 0 | 1 | 0 | **0** | **0** | **1** | **1** |
| 0 | 0 | 1 | 1 | **0** | **1** | **0** | **0** |
| 0 | 1 | 0 | 0 | **0** | **1** | **0** | **1** |
| 0 | 1 | 0 | 1 | **0** | **1** | **1** | **0** |
| 0 | 1 | 1 | 0 | **0** | **1** | **1** | **1** |
| 0 | 1 | 1 | 1 | **1** | **0** | **0** | **0** |
| 1 | 0 | 0 | 0 | **1** | **0** | **0** | **1** |
| 1 | 0 | 0 | 1 | **0** | **0** | **0** | **0** |
| 1 | 0 | 1 | 0 | **X** | **X** | **X** | **X** |
| 1 | 0 | 1 | 1 | **X** | **X** | **X** | **X** |
| 1 | 1 | 0 | 0 | **X** | **X** | **X** | **X** |
| 1 | 1 | 0 | 1 | **X** | **X** | **X** | **X** |
| 1 | 1 | 1 | 0 | **X** | **X** | **X** | **X** |
| 1 | 1 | 1 | 1 | **X** | **X** | **X** | **X** |

# K-maps

| Input | | | | Output | | | |
|---|---|---|---|---|---|---|---|
| I8 | I4 | I2 | I1 | O8 | O4 | O2 | O1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | X | X | X | X |
| 1 | 0 | 1 | 1 | X | X | X | X |
| 1 | 1 | 0 | 0 | X | X | X | X |
| 1 | 1 | 0 | 1 | X | X | X | X |
| 1 | 1 | 1 | 0 | X | X | X | X |
| 1 | 1 | 1 | 1 | X | X | X | X |

## O8

| | | I8′I4′ | I8′I4 | I8I4 | I8I4′ |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| I2′I1′ | 00 | 0 | 0 | X | 1 |
| I2′I1 | 01 | 0 | 0 | X | 0 |
| I2I1 | 11 | 0 | 1 | X | X |
| I2I1′ | 10 | 0 | 0 | X | X |

## O4

| | | I8′I4′ | I8′I4 | I8I4 | I8I4′ |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| I2′I1′ | 00 | 0 | 1 | X | 0 |
| I2′I1 | 01 | 0 | 1 | X | 0 |
| I2I1 | 11 | 1 | 0 | X | X |
| I2I1′ | 10 | 0 | 1 | X | X |

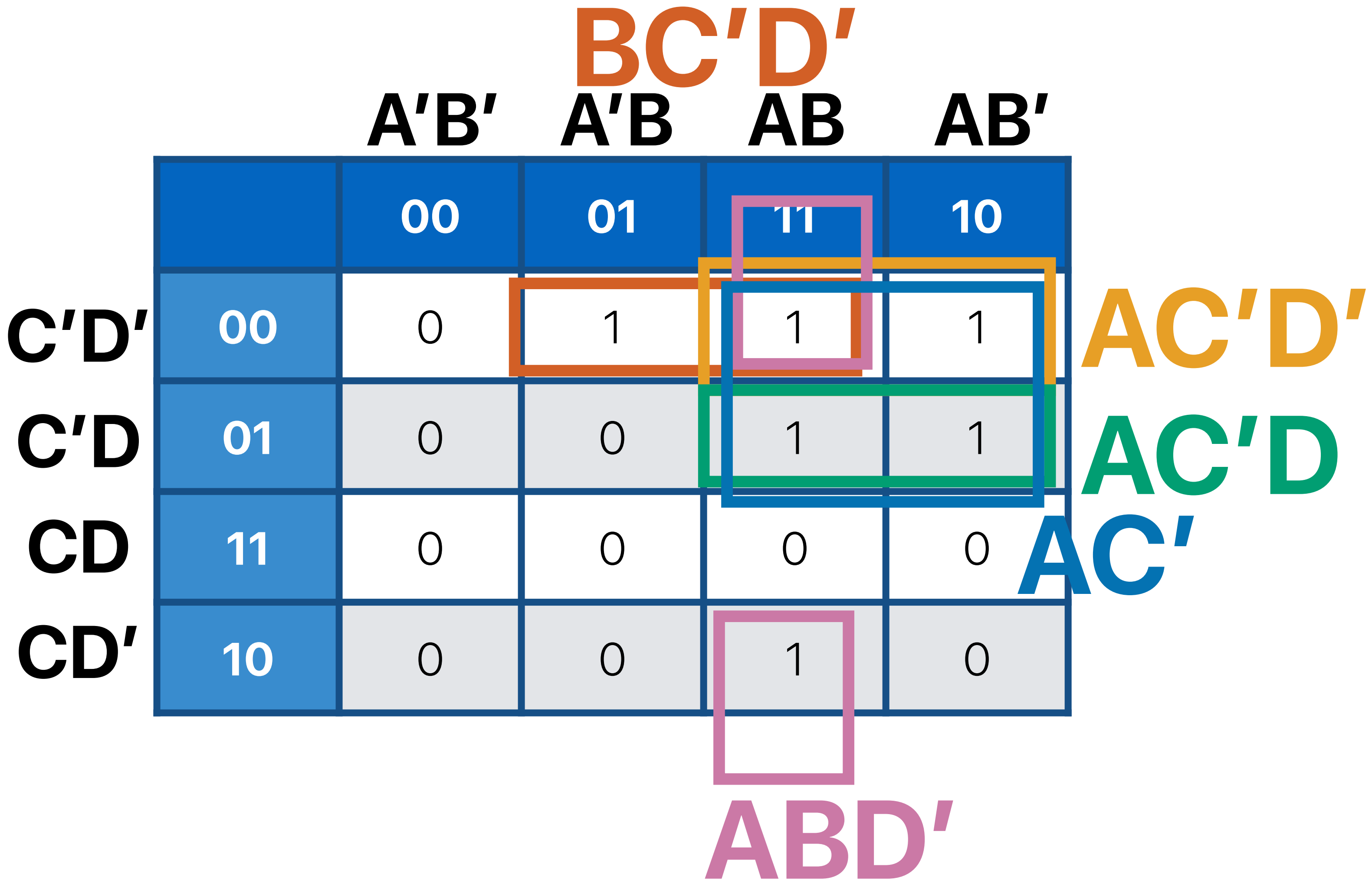
## O2

| | | I8′I4′ | I8′I4 | I8I4 | I8I4′ |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| I2′I1′ | 00 | 0 | 0 | X | 0 |
| I2′I1 | 01 | 1 | 1 | X | 0 |
| I2I1 | 11 | 0 | 0 | X | X |
| I2I1′ | 10 | 1 | 1 | X | X |

## O1

| | | I8′I4′ | I8′I4 | I8I4 | I8I4′ |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| I2′I1′ | 00 | 1 | 1 | X | 1 |
| I2′I1 | 01 | 0 | 0 | X | 0 |
| I2I1 | 11 | 0 | 0 | X | X |
| I2I1′ | 10 | 1 | 1 | X | X |

# Examples to illustrate terms



| | | A′B′ | A′B | AB | AB′ |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| C′D′ | 00 | 0 | 1 | 1 | 1 |
| C′D | 01 | 0 | 0 | 1 | 1 |
| CD | 11 | 0 | 0 | 0 | 0 |
| CD′ | 10 | 0 | 0 | 1 | 0 |

**5 prime implicants:**
**BC′D′, AC′D′, AC′D, AC′, ABD′**

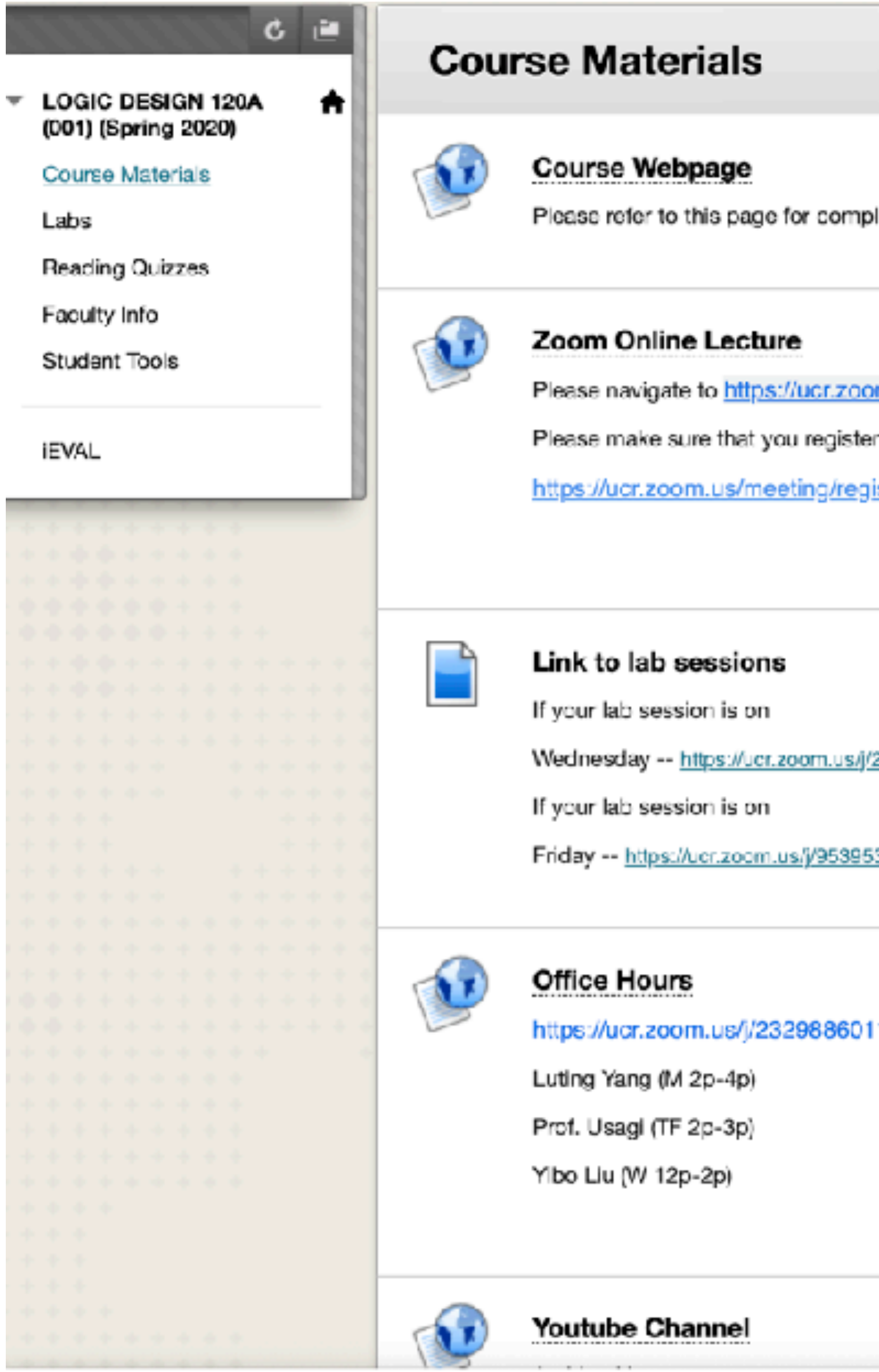
**minimum cover:**
**BC′D′ + AC′ + ABD′**

# Quine-McCluskey Algorithm

- Step 1: choose an element of the ON-set
- Step 2: find "maximal" groupings of 1s and Xs adjacent to that element
  - onsider top/bottom row, left/right column, and corner adjacencies
  - this forms prime implicants (number of elements always a power of 2)
  - Repeat Steps 1 and 2 to find all prime implicants
- Step 3: revisit the 1s in the K-map
  - if covered by single prime implicant, it is essential, and participates in final cover
  - 1s covered by essential prime implicant do not need to be revisited
- Step 4: if there remain 1s not covered by essential prime implicants
  - select the smallest number of prime implicants that cover the remaining 1s

# Announcement

- Please also register yourself to the following two
  - Please register to your corresponding lab sessions
    - The link is under iLearn > course materials
  - Please register to office hours
    - The link is also under iLearn > course materials
- Reading quiz 2 will be up tonight
  - Under iLearn > reading quizzes
- Lab 1 due 4/7
  - Submit through iLearn > Labs



## Course Materials

**Course Webpage**

Please refer to this page for compl

**Zoom Online Lecture**

Please navigate to https://ucr.zoo

Please make sure that you register

https://ucr.zoom.us/meeting/regi

**Link to lab sessions**

If your lab session is on

Wednesday -- https://ucr.zoom.us/j/2

If your lab session is on

Friday -- https://ucr.zoom.us/j/95395

**Office Hours**

https://ucr.zoom.us/j/232988601

Luting Yang (M 2p-4p)

Prof. Usagi (TF 2p-3p)

Yibo Liu (W 12p-2p)

**Youtube Channel**

Electrical
Computer Science
Engineering 120A

つづく