Can we build any 2-input, single-1-output function using what we already know?
We can build (NOT, NOR, NAND) from CMOS gates, and combine them to build AND and OR.
Can we build arbitrary single-1-output functions from just these?

| A | B | f |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

***IF***

"(**A is FALSE**) **AND** (**B is TRUE**)"

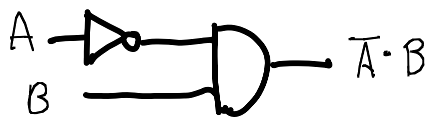
is a **TRUE** statement.

| A | B | $\bar{A}$ | $B \cdot \bar{A} = f$ |
|---|---|---|---|
| 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 |

FALSE, TRUE, false

THEN f is True

$f = \bar{A} \cdot B \equiv m_1$ (minterm)

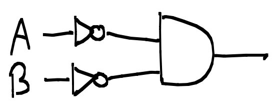



Our simple (single-1-output) functions are called "minterms". Our decomposition is called a "minterm expansion of f ".

## Minterms

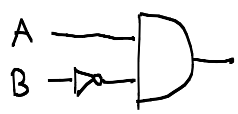
| A | B | |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

$\rightarrow \bar{A} \cdot \bar{B} \equiv m_0$



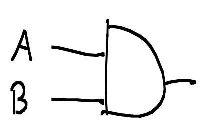

| A | B | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

$\rightarrow A \cdot \bar{B} \equiv m_2$




| A | B | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

$\rightarrow A \cdot B \equiv m_3$




We now have all possible simple 2-input functions, ie., our minterms m0, m1, m2, and m3.

And we know how to build each from the 2-input gates we already have and NOT.

So, we can now build ANY 2-input function! Just OR these as needed.

---

Can we do the same for 3-input functions?

| A | B | C | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 |

$\rightarrow m_1 = \bar{A}$ and $\bar{B}$ and $C$

A 3-input function?

We CAN build all 2-input functions.
BUT, WHAT is a 3-input function?
CAN we build from 2-input functions?

AND(X ,Y, Z)

is that the same as,

AND( AND(X, Y), Z) ???

| A | B | A · B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| (A·B) | C | (A·B)·C |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| B | C | B·C |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| A | (B·C) | A·(B·C) |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Are these the same?

Do they have exactly the same output values for exactly the same input values? Check ALL input values.

| A | B | C | (A·B)·C |
|---|---|---|---|
| 0 | 0 | 0 | ? |
| 0 | 0 | 1 | ? |
| 0 | 1 | 0 | ? |
| 0 | 1 | 1 | ? |
| 1 | 0 | 0 | ? |
| 1 | 0 | 1 | ? |
| 1 | 1 | 0 | ? |
| 1 | 1 | 1 | ? |

SAME?

| A | B | C | A·(B·C) |
|---|---|---|---|
| 0 | 0 | 0 | |
| 0 | 0 | 1 | |
| 0 | 1 | 0 | |
| 0 | 1 | 1 | |
| 1 | 0 | 0 | |
| 1 | 0 | 1 | |
| 1 | 1 | 0 | |
| 1 | 1 | 1 | |

# Basic Algebraic Properties

## equivalent functions

a logical constant = TRUE for all inputs.

$1 \cdot P = P$

$0 \cdot P = 0$

$P \cdot P = P$

$P \cdot \overline{P} = 0$

$A \cdot B = B \cdot A$

$(AB)C = A(BC)$

Proof

| 1 | P | 1·P |
|---|---|---|
| 1 | 0 | 0 |
| 1 | 1 | 1 |

same output

| 1 | P | P |
|---|---|---|
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Proof

| P | P·P |
|---|---|
| 0 | 0 |
| 1 | 1 |

| P | P |
|---|---|
| 0 | 0 |
| 1 | 1 |

Proof

| $P$ | $\overline{P}$ | $P \cdot \overline{P}$ |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 0 | 0 |

always 0

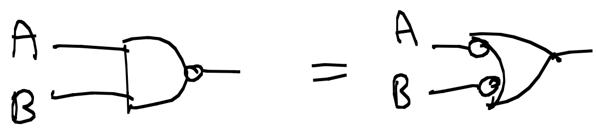
Handy tricks for (1) proving algebraic properties or doing algebraic operations, and (2) converting logic circuits from one type of logic gates to another. (Aside: can you prove Duality?)

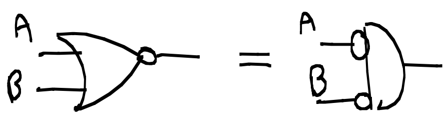
## DeMorgan's Laws

$\overline{AB} = \bar{A} + \bar{B}$

| A | B | A·B | $\overline{AB}$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

| $\bar{A}$ | $\bar{B}$ | $\bar{A}+\bar{B}$ |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |




$\overline{A+B} = \bar{A} \cdot \bar{B}$ DUAL

## Duality

· → +
+ → ·
1 → 0
0 → 1

starting with an identity, gives an identity.

$1 \cdot P = P$
$0 + P = P$ (duality)

$0 \cdot P = 0$
$1 + P = 1$ (duality)

$(A+B)C = A \cdot C + B \cdot C$
$(A \cdot B) + C = (A+C) \cdot (B+C)$ (dual)

Build any k-input function? Use k-ary minterm expansion.

Build k-ary minterms from 2-input gates we already have?

| A | B | C | f(A, B, C) | $m_0$ | $m_3$ | $m_4$ | $m_6$ |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

$f = m_0 + m_3 + m_4 + m_6$ — minterm expansion (prove '+' associative)

(Recall: $P + 0 = P$)

$\Rightarrow m_0 + 0 + 0 + 0 = m_0$

any row is similar, 1 non-zero OR'd w/ zeroes.

What sort of functions are these minterms?

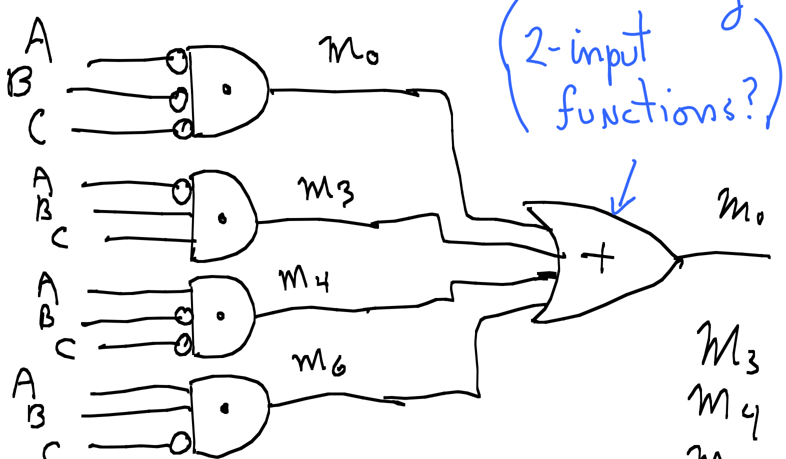
if A is F and B is F and C is F

| A | B | C | $m_0$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| ⋮ | ⋮ | ⋮ | ⋮ |
| 1 | 1 | 1 | 0 |

$$m_0(A,B,C) = \bar{A}\cdot\bar{B}\cdot\bar{C}$$

| $\bar{A}\cdot\bar{B}\cdot\bar{C}$ | |
|---|---|
| $\bar{0}\cdot\bar{0}\cdot\bar{0}$ | = 1 |
| $\bar{0}\cdot\bar{0}\cdot\bar{T}$ | = 0 |
| $\bar{0}\cdot\bar{T}\cdot\bar{0}$ | = 0 |
| $\bar{0}\cdot\bar{T}\cdot\bar{T}$ | = 0 |
| $\bar{T}\cdot\bar{0}\cdot\bar{0}$ | = 0 |
| $\bar{T}\cdot\bar{0}\cdot\bar{T}$ | = 0 |
| $\bar{T}\cdot\bar{T}\cdot\bar{0}$ | = 0 |
| $\bar{T}\cdot\bar{T}\cdot\bar{T}$ | = 0 |

yes, it matches, i.e. $\bar{A}\bar{B}\bar{C}$ really is correct expression for $m_0$.




Build using (2-input functions?)

$$m_0 + m_3 + m_4 + m_6 = f$$

$$m_3 = \bar{A}BC$$
$$m_4 = A\bar{B}\bar{C}$$
$$m_6 = AB\bar{C}$$

Complete Logic set → (OR, AND, NOT)

MAXTERM EXPANSION

| A | B | C | f |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

6 minterms, but only 2 zeroes?

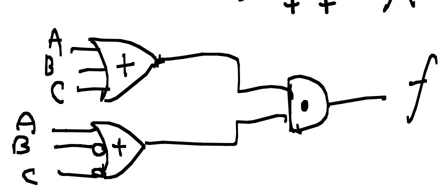
| $\bar{f}$ | |
|---|---|
| 1 | → $m_0$ |
| 0 | |
| 0 | |
| 1 | → $m_3$ |
| 0 | |
| 0 | |
| 0 | |
| 0 | |

2 minterms

$$\bar{f} = m_0 + m_3 = \bar{A}\bar{B}\bar{C} + \bar{A}BC$$

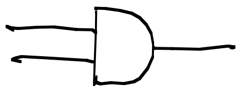
$$f = \bar{\bar{f}} = \overline{\bar{A}\bar{B}\bar{C} + \bar{A}BC} \xrightarrow{\text{de Morgan}} (\overline{\bar{A}+\bar{B}+\bar{C}})(\overline{\bar{A}+B+C})$$

$$= (A+B+C)(A+\bar{B}+\bar{C})$$

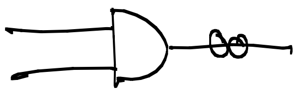


# Pushing Bubbles

implement using NOR logic



$= f$



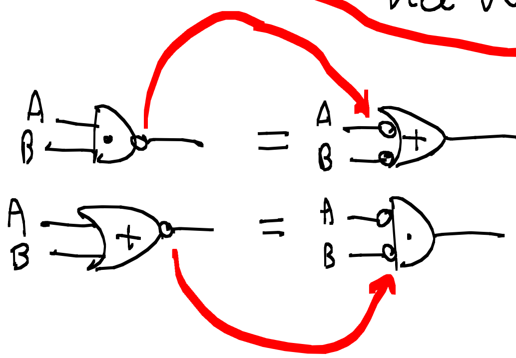
$= \bar{\bar{f}} = f$

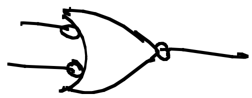
Bubble goes from output side to all inputs, or vice versa

## DeMorgan's Laws

$\overline{AB} = \bar{A} + \bar{B} \Rightarrow$

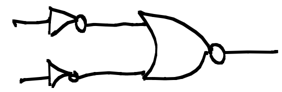
$\overline{A+B} = \bar{A} \cdot \bar{B} \Rightarrow$



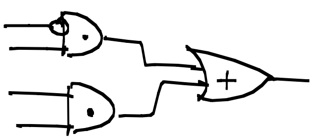
$= f$

equivalent circuit in NOR logic

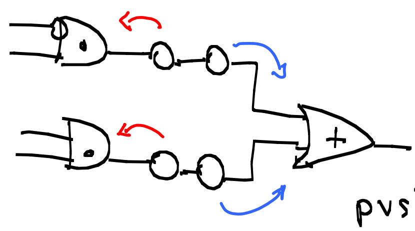



---

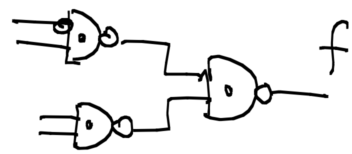
## AND-OR → NAND-NAND



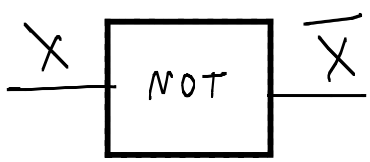

Add bubble pairs



push bubbles

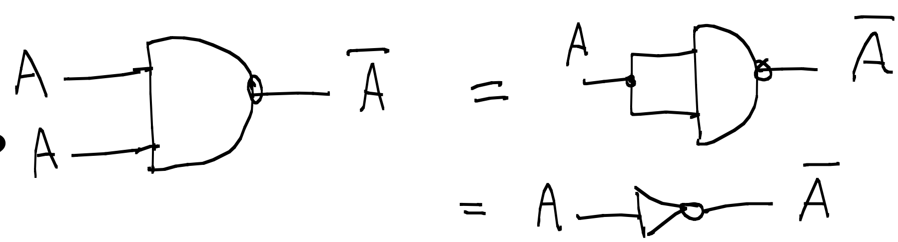



NAND — NAND is complete logic family, if we have NOT.



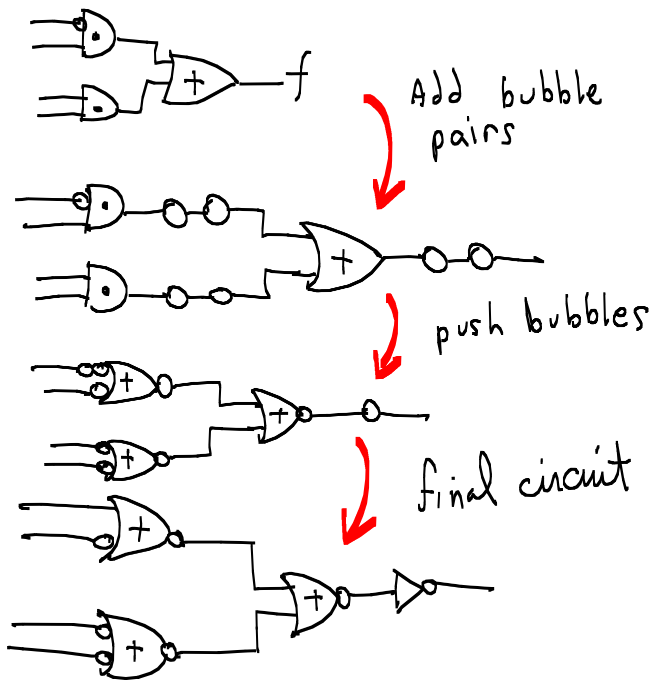

is there a NAND circuit for this function?

| A | ~~B~~ A | NAND |
| --- | --- | --- |
| 0 | 0 | 1 |
| ~~0~~ | ~~1~~ | ~~1~~ |
| ~~1~~ | ~~0~~ | ~~1~~ |
| 1 | 1 | 0 |

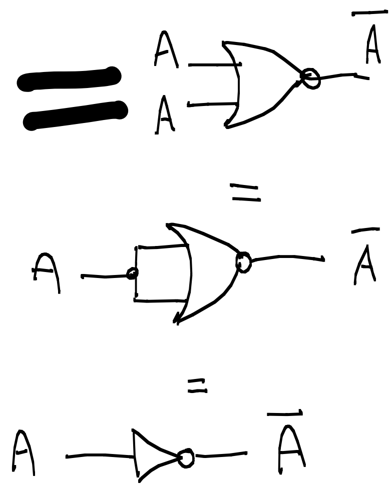


# NOR–NOR is complete, if we have NOT.

AND–OR ➜ NOR–NOR




| A | ~~B~~ A | NOR |
| --- | --- | --- |
| 0 | 0 | 1 |
| ~~0~~ | ~~1~~ | ~~0~~ |
| 1 | 1 | 0 |

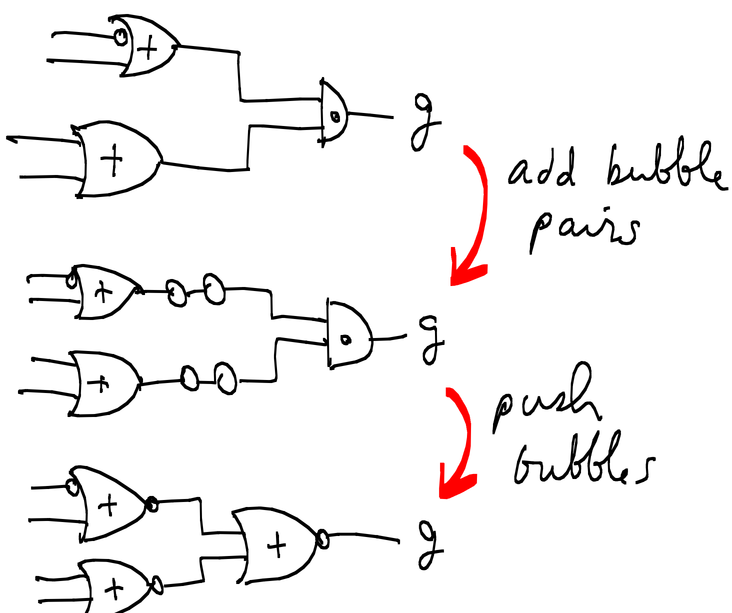



We can push bubbles to get a NOR–NOR circuit from an AND–OR circuit.
But, is there an easier way?

Is there an easier way to Get a NOR–NOR circuit?

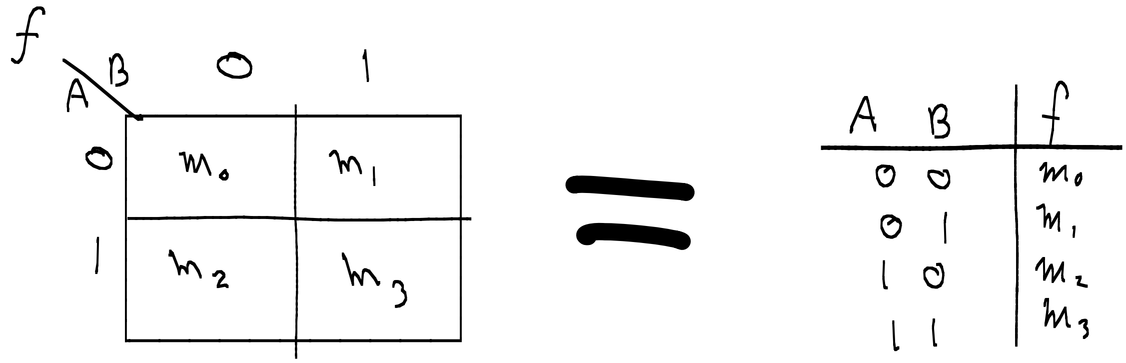
yes, use a maxterm expansion

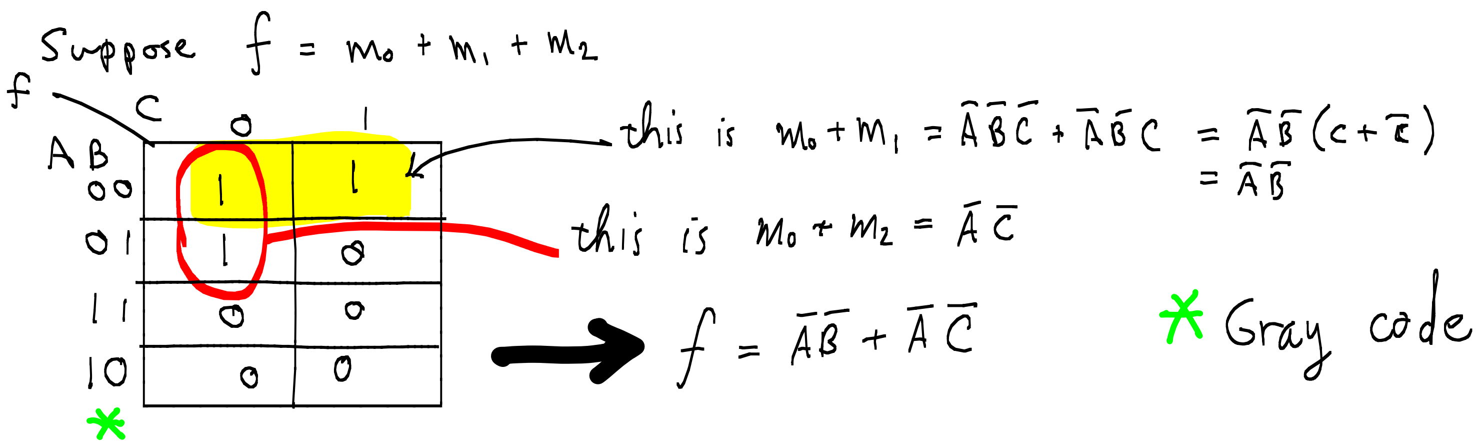
OR–AND maxterm circuit

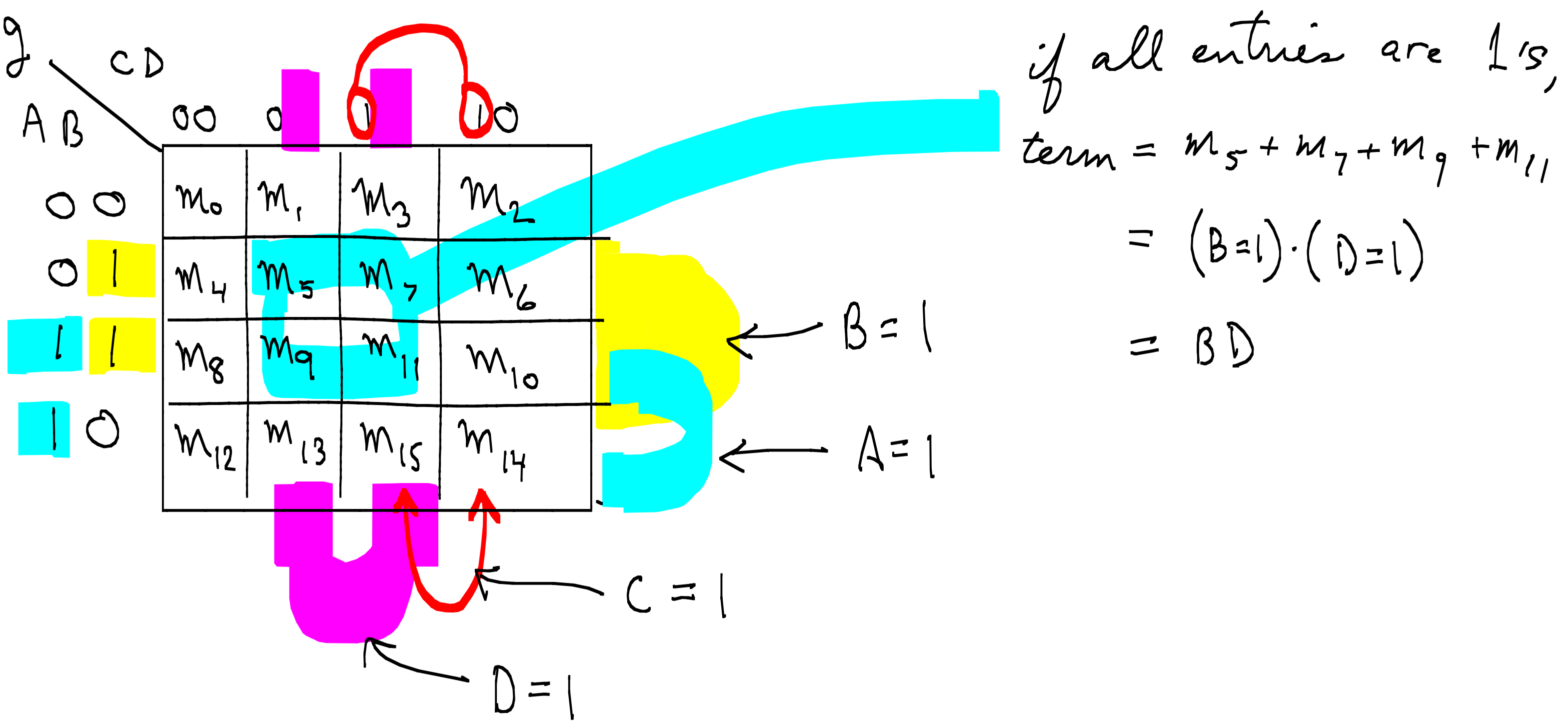



NOR–NOR maxterm circuit

Karnaugh Maps are truth tables that allow us to find simpler logic expressions (and circuits). This is something that can also be done algebraically, but is usually harder that way. Is there a general procedure for minimizing circuits? (Is it computable?)




Find a collection of terms that:
1. covers all ones
2. has the fewest terms possible.

Term = a region where one or more variables are "don't cares": the output does not change when the variables change value, those variables make no difference. The values of the remaining variables define the term.

So, what can we say about mult-bit symbols as output from our Boolean functions? Deal with each bit of output symbol individually.

| | $f$ | $f_1 f_0$ |
|---|---|---|
| 00 | → | 01 |
| 01 | → | 10 |
| 10 | → | 11 |
| 11 | → | 00 |

$\Longrightarrow$

| | | $f_1$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

| | | $f_0$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

are these 2-bit symbols or 2 1-bit symbols? Could be either:

$f: \{0,1\} \times \{0,1\} \to \{00, 01, 10, 11\}$ ← 4 symbols

(each $\{0,1\}$: 2 symbols)

$\{A, B\} \times \{\Theta, \phi\} \to \{@, \&, *, \$\}$

$S_1 \times S_2 \to S_3$

OR

$f: \{00, 01, 10, 11\} \to \{00, 01, 10, 11\}$

(4 symbols → 4 symbols)

$\{@, \&, *, \$\} \to \{@, \&, *, \$\}$

$S_3 \to S_3$

| | $g$ | |
|---|---|---|
| 000 | → | 00 |
| 001 | → | 01 |
| 010 | → | 01 |
| 011 | → | 10 |
| 100 | → | 01 |
| 101 | → | 10 |
| 110 | → | 10 |
| 111 | → | 11 |

In general, we can now handle any functions with m-bit input and n-bit output. We can implement in hardware:

===> ANY finite symbol set of size 2^k by using {0, 1}^k multi-bit symbols
===> ANY finite combination of finite sets
===> ANY functions mapping from any combination to any combination