# ECE6703J

## Computer-Aided Design of Integrated Circuits

Two-level Logic Synthesis

# Outline

- Introduction

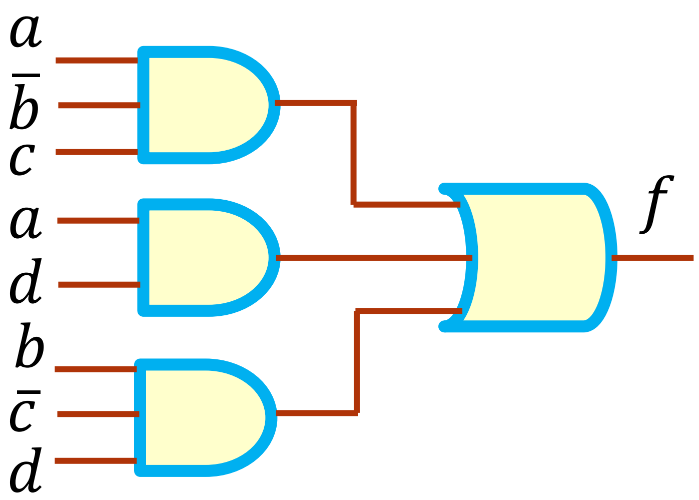
- Quine-McCluskey Algorithm
  - Phase I
  - Phase II

- Heuristic Method
  - Overview
  - Expand step

# Two-Level Logic Design

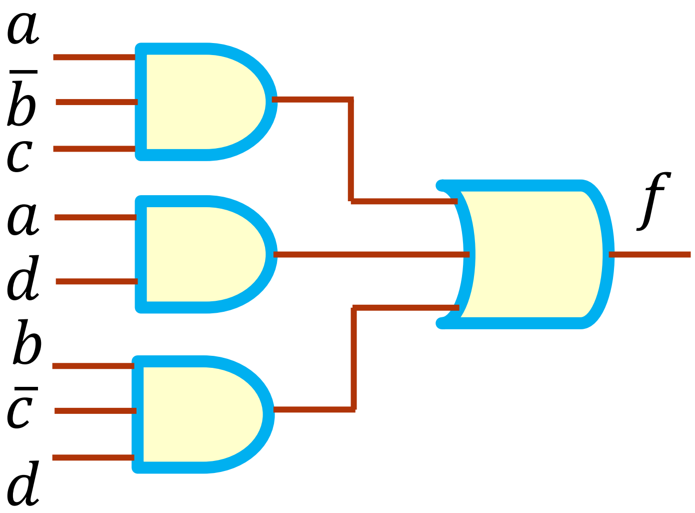
- A logic implementation based on **sum-of-products (SOP)** expression.
  - First level: **many** AND gates.
  - Second level: **one** "big" OR gate.

$$f = a\bar{b}c + ad + b\bar{c}d$$



# Two-Level Minimization

- Trying to find **minimal** SOP logic.
  - Want: **fewest input wires**.
  - These input wires called **literals**: a variable in true or complemented form. Thus, we want to minimize **number of literals**.



$$f = a\bar{b}c + ad + b\bar{c}d$$

**3 product terms, 8 literals**

Number of literals is a good metric for the complexity of SOP function.

# Two-Level Minimization

- None of the following methods you know from **fundamental digital design** is practical.
  - **Boolean algebra:** Hard with many variables. Can’t tell when have a good solution.
  - **Karnaugh maps**: Can only deal with up to 6 variables.

- We need a systematic computational method.

# Outline

- Introduction

- Quine-McCluskey Algorithm
  - Phase I
  - Phase II

- Heuristic Method
  - Overview
  - Expand step

# Quine-McCluskey Algorithm

- Solve the 2-level minimization problem
  - Find an SOP with the **minimal number of literals**.
  - Developed by Walliam Quine and Edward McCluskey in 1956.

- Basic idea: Eliminate as many literals as possible by systematically applying $xy + x\bar{y} = x$.
  - A "**computational**" version of Karnaugh map.

# Implicants and Prime Implicants

- **Implicant** of a Boolean function $F$: A **product term** $P$ that "**implies**" $F$.
  - I.e., whenever $P$ is 1, $F$ also takes value 1.
  - Example: $abc$ and $ab$ are implicants of $abc + ab\bar{c} + \bar{a}c$, but not $a$ (think about the circles in Karnaugh map)
- **Prime implicant**: an **implicant** that cannot be covered by a "**larger**" implicant.
  - "**Larger**" means have **fewer** literals.
  - Example: $ab$ is a prime implicant of $abc + ab\bar{c} + \bar{a}c$, while $abc$ is not.
  - Corresponds to a largest circle we can draw in Karnaugh maps.

Claim: an SOP with the **minimal number of literals** only contains **prime implicants**.

# Two Major Steps

1. Find all **prime implicants** of the function.
   - Just like finding all "largest" circles in Karnaugh map.

2. Use those prime implicants in a **prime implicant chart** to find a **minimal** set of prime implicants that covers the function.
   - Just like selecting a minimal number of "largest" circles in Karnaugh map to cover all 1's.

# Phase I: Find All Prime Implicants

- Group the minterms according to the number of 1s in the minterm.
  - This way we only have to compare minterms from **adjacent** groups.
- Example: $f(a,b,c,d) = \sum m(0,1,2,5,6,7,8,9,10,14)$

| | | |
|---|---|---|
| group 0 | 0 | 0000 |
| group 1 | 1 | 0001 |
| | 2 | 0010 |
| | 8 | 1000 |
| group 2 | 5 | 0101 |
| | 6 | 0110 |
| | 9 | 1001 |
| | 10 | 1010 |
| group 3 | 7 | 0111 |
| | 14 | 1110 |

# Phase I: Find All Prime Implicants

- Combining group 0 and group 1.
  - If two products can be combined, check mark them, and record the resulting products.

| | Column I | | | Column II | |
|---|---|---|---|---|---|
| group 0 | 0 | 0000 | ✓ | 0,1 | 000- |
| group 1 | 1 | 0001 | ✓ | | |
| | 2 | 0010 | | | |
| | 8 | 1000 | | | |
| group 2 | 5 | 0101 | | | |
| | 6 | 0110 | | | |
| | 9 | 1001 | | | |
| | 10 | 1010 | | | |
| group 3 | 7 | 0111 | | | |
| | 14 | 1110 | | | |

# Phase I: Find All Prime Implicants

- Combining group 0 and group 1.
    - If two products can be combined, check mark them, and record the resulting products.

Column I

| | | | |
|---|---|---|---|
| group 0 | 0 | 0000 | ✓ |
| group 1 | 1 | 0001 | ✓ |
| | 2 | 0010 | ✓ |
| | 8 | 1000 | |
| group 2 | 5 | 0101 | |
| | 6 | 0110 | |
| | 9 | 1001 | |
| | 10 | 1010 | |
| group 3 | 7 | 0111 | |
| | 14 | 1110 | |

Column II

| | |
|---|---|
| 0,1 | 000- |
| 0,2 | 00-0 |

# Phase I: Find All Prime Implicants

- Combining group 0 and group 1.
  - If two products can be combined, check mark them, and record the resulting products.

Column I

| | | | |
|---|---|---|---|
| group 0 | 0 | 0000 | ✓ |
| group 1 | 1 | 0001 | ✓ |
| | 2 | 0010 | ✓ |
| | 8 | 1000 | ✓ |
| group 2 | 5 | 0101 | |
| | 6 | 0110 | |
| | 9 | 1001 | |
| | 10 | 1010 | |
| group 3 | 7 | 0111 | |
| | 14 | 1110 | |

Column II

| | |
|---|---|
| 0,1 | 000- |
| 0,2 | 00-0 |
| 0,8 | -000 |

# Phase I: Find All Prime Implicants

- Question: Does it make sense to combine group 0 with group 2 or 3?
  - No. There are at least two bits that are different.

Column I

|  |  |  |  |
|---|---|---|---|
| group 0 | 0 | 0000 | ✓ |
| group 1 | 1 | 0001 | ✓ |
|  | 2 | 0010 | ✓ |
|  | 8 | 1000 | ✓ |
| group 2 | 5 | 0101 |  |
|  | 6 | 0110 |  |
|  | 9 | 1001 |  |
|  | 10 | 1010 |  |
| group 3 | 7 | 0111 |  |
|  | 14 | 1110 |  |

Column II

|  |  |
|---|---|
| 0,1 | 000- |
| 0,2 | 00-0 |
| 0,8 | -000 |

So, the next step is to combine group 1 with group 2.

# Phase I: Find All Prime Implicants

- Combining group 1 and group 2.
  - If two products can be combined, check mark them, and record the resulting products.

Column I

| | | | |
|---|---|---|---|
| group 0 | 0 | 0000 | ✓ |
| group 1 | 1 | 0001 | ✓ |
| | 2 | 0010 | ✓ |
| | 8 | 1000 | ✓ |
| group 2 | 5 | 0101 | ✓ |
| | 6 | 0110 | |
| | 9 | 1001 | |
| | 10 | 1010 | |
| group 3 | 7 | 0111 | |
| | 14 | 1110 | |

Column II

| | |
|---|---|
| 0,1 | 000- |
| 0,2 | 00-0 |
| 0,8 | -000 |
| 1,5 | 0-01 |

# Phase I: Find All Prime Implicants

- Combining group 1 and group 2.
  - If two products can be combined, check mark them, and record the resulting products.

Column I

| | | | |
|---|---|---|---|
| group 0 | 0 | 0000 | ✓ |
| group 1 | 1 | 0001 | ✓ |
| | 2 | 0010 | ✓ |
| | 8 | 1000 | ✓ |
| group 2 | 5 | 0101 | ✓ |
| | 6 | 0110 | |
| | 9 | 1001 | |
| | 10 | 1010 | |
| group 3 | 7 | 0111 | |
| | 14 | 1110 | |

Column II

| | |
|---|---|
| 0,1 | 000- |
| 0,2 | 00-0 |
| 0,8 | -000 |
| 1,5 | 0-01 |

No combining

# Phase I: Find All Prime Implicants

- Combining group 1 and group 2.
  - If two products can be combined, check mark them, and record the resulting products.

Column I

| | | | |
|---|---|---|---|
| group 0 | 0 | 0000 | ✓ |
| group 1 | 1 | 0001 | ✓ |
| | 2 | 0010 | ✓ |
| | 8 | 1000 | ✓ |
| group 2 | 5 | 0101 | ✓ |
| | 6 | 0110 | |
| | 9 | 1001 | ✓ |
| | 10 | 1010 | |
| group 3 | 7 | 0111 | |
| | 14 | 1110 | |

Column II

| | |
|---|---|
| 0,1 | 000- |
| 0,2 | 00-0 |
| 0,8 | -000 |
| 1,5 | 0-01 |
| 1,9 | -001 |

# Phase I: Find All Prime Implicants

- Combining group 1 and group 2.
  - If two products can be combined, check mark them, and record the resulting products.

Column I

| | | | |
|---|---|---|---|
| group 0 | 0 | 0000 | ✓ |
| group 1 | 1 | 0001 | ✓ |
| | 2 | 0010 | ✓ |
| | 8 | 1000 | ✓ |
| group 2 | 5 | 0101 | ✓ |
| | 6 | 0110 | |
| | 9 | 1001 | ✓ |
| | 10 | 1010 | |
| group 3 | 7 | 0111 | |
| | 14 | 1110 | |

Column II

| | |
|---|---|
| 0,1 | 000- |
| 0,2 | 00-0 |
| 0,8 | -000 |
| 1,5 | 0-01 |
| 1,9 | -001 |

**No combining**

# Phase I: Find All Prime Implicants

- Combining group 1 and group 2.
  - If two products can be combined, check mark them, and record the resulting products.

Column I

| | | | |
|---|---|---|---|
| group 0 | 0 | 0000 | ✓ |
| group 1 | 1 | 0001 | ✓ |
| | 2 | 0010 | ✓ |
| | 8 | 1000 | ✓ |
| group 2 | 5 | 0101 | ✓ No combining |
| | 6 | 0110 | |
| | 9 | 1001 | ✓ |
| | 10 | 1010 | |
| group 3 | 7 | 0111 | |
| | 14 | 1110 | |

• • •

Column II

| | |
|---|---|
| 0,1 | 000- |
| 0,2 | 00-0 |
| 0,8 | -000 |
| 1,5 | 0-01 |
| 1,9 | -001 |

# Phase I: Find All Prime Implicants

- Combining group 1 and group 2.
  - If two products can be combined, check mark them, and record the resulting products.

Column I

| | | | |
|---|---|---|---|
| group 0 | 0 | 0000 | ✓ |
| group 1 | 1 | 0001 | ✓ |
| | 2 | 0010 | ✓ |
| | 8 | 1000 | ✓ |
| group 2 | 5 | 0101 | ✓ |
| | 6 | 0110 | ✓ |
| | 9 | 1001 | ✓ |
| | 10 | 1010 | ✓ |
| group 3 | 7 | 0111 | |
| | 14 | 1110 | |

Column II

| | |
|---|---|
| 0,1 | 000- |
| 0,2 | 00-0 |
| 0,8 | -000 |
| 1,5 | 0-01 |
| 1,9 | -001 |
| 2,6 | 0-10 |
| 2,10 | -010 |
| 8,9 | 100- |
| 8,10 | 10-0 |

# Phase I: Find All Prime Implicants

- Combining group 2 and group 3.
  - If two products can be combined, check mark them, and record the resulting products.

Column I

| | | | |
|---|---|---|---|
| group 0 | 0 | 0000 | ✓ |
| group 1 | 1 | 0001 | ✓ |
| | 2 | 0010 | ✓ |
| | 8 | 1000 | ✓ |
| group 2 | 5 | 0101 | ✓ |
| | 6 | 0110 | ✓ |
| | 9 | 1001 | ✓ |
| | 10 | 1010 | ✓ |
| group 3 | 7 | 0111 | ✓ |
| | 14 | 1110 | |

Column II

| | |
|---|---|
| 0,1 | 000- |
| 0,2 | 00-0 |
| 0,8 | -000 |
| 1,5 | 0-01 |
| 1,9 | -001 |
| 2,6 | 0-10 |
| 2,10 | -010 |
| 8,9 | 100- |
| 8,10 | 10-0 |
| 5,7 | 01-1 |

• • •

# Phase I: Find All Prime Implicants

- Combining group 2 and group 3.

Column I

| | | | |
|---|---|---|---|
| group 0 | 0 | 0000 | ✓ |
| group 1 | 1 | 0001 | ✓ |
| | 2 | 0010 | ✓ |
| | 8 | 1000 | ✓ |
| group 2 | 5 | 0101 | ✓ |
| | 6 | 0110 | ✓ |
| | 9 | 1001 | ✓ |
| | 10 | 1010 | ✓ |
| group 3 | 7 | 0111 | ✓ |
| | 14 | 1110 | ✓ |

Column II

| | |
|---|---|
| 0,1 | 000- |
| 0,2 | 00-0 |
| 0,8 | -000 |
| 1,5 | 0-01 |
| 1,9 | -001 |
| 2,6 | 0-10 |
| 2,10 | -010 |
| 8,9 | 100- |
| 8,10 | 10-0 |
| 5,7 | 01-1 |
| 6,7 | 011- |
| 6,14 | -110 |
| 10,14 | 1-10 |

# Phase I: Find All Prime Implicants

- Divide column II into groups based on numbers of ones.

Column I

| | | |
|---|---|---|
| 0 | 0000 | ✓ |
| 1 | 0001 | ✓ |
| 2 | 0010 | ✓ |
| 8 | 1000 | ✓ |
| 5 | 0101 | ✓ |
| 6 | 0110 | ✓ |
| 9 | 1001 | ✓ |
| 10 | 1010 | ✓ |
| 7 | 0111 | ✓ |
| 14 | 1110 | ✓ |

Column II

| | | |
|---|---|---|
| group 0 | 0,1 | 000- |
| | 0,2 | 00-0 |
| | 0,8 | -000 |
| group 1 | 1,5 | 0-01 |
| | 1,9 | -001 |
| | 2,6 | 0-10 |
| | 2,10 | -010 |
| | 8,9 | 100- |
| | 8,10 | 10-0 |
| group 2 | 5,7 | 01-1 |
| | 6,7 | 011- |
| | 6,14 | -110 |
| | 10,14 | 1-10 |

# Phase I: Find All Prime Implicants

- Combining group 0 and group 1 of Column II.

**Column I**

| | | |
|---|---|---|
| 0 | 0000 | ✓ |
| 1 | 0001 | ✓ |
| 2 | 0010 | ✓ |
| 8 | 1000 | ✓ |
| 5 | 0101 | ✓ |
| 6 | 0110 | ✓ |
| 9 | 1001 | ✓ |
| 10 | 1010 | ✓ |
| 7 | 0111 | ✓ |
| 14 | 1110 | ✓ |

**Column II**

| | | |
|---|---|---|
| group 0 | 0,1 | 000- |
| | 0,2 | 00-0 |
| | 0,8 | -000 |
| group 1 | 1,5 | 0-01 |
| | 1,9 | -001 |
| | 2,6 | 0-10 |
| | 2,10 | -010 |
| | 8,9 | 100- |
| | 8,10 | 10-0 |
| group 2 | 5,7 | 01-1 |
| | 6,7 | 011- |
| | 6,14 | -110 |
| | 10,14 | 1-10 |

**Column III**

**No combining**

# Phase I: Find All Prime Implicants

- Combining group 0 and group 1 of Column II.

Column I

| | | |
|---|---|---|
| 0 | 0000 | ✓ |
| 1 | 0001 | ✓ |
| 2 | 0010 | ✓ |
| 8 | 1000 | ✓ |
| 5 | 0101 | ✓ |
| 6 | 0110 | ✓ |
| 9 | 1001 | ✓ |
| 10 | 1010 | ✓ |
| 7 | 0111 | ✓ |
| 14 | 1110 | ✓ |

Column II

| | | |
|---|---|---|
| group 0 | 0,1 | 000- |
| | 0,2 | 00-0 |
| | 0,8 | -000 |
| group 1 | 1,5 | 0-01 |
| | 1,9 | -001 |
| | 2,6 | 0-10 |
| | 2,10 | -010 |
| | 8,9 | 100- |
| | 8,10 | 10-0 |
| group 2 | 5,7 | 01-1 |
| | 6,7 | 011- |
| | 6,14 | -110 |
| | 10,14 | 1-10 |

Column III

**No combining**

# Phase I: Find All Prime Implicants

- Combining group 0 and group 1 of Column II.

Column I

| | | |
|---|---|---|
| 0 | 0000 | ✓ |
| 1 | 0001 | ✓ |
| 2 | 0010 | ✓ |
| 8 | 1000 | ✓ |
| 5 | 0101 | ✓ |
| 6 | 0110 | ✓ |
| 9 | 1001 | ✓ |
| 10 | 1010 | ✓ |
| 7 | 0111 | ✓ |
| 14 | 1110 | ✓ |

Column II

| | | |
|---|---|---|
| group 0 | 0,1 | 000- |
| | 0,2 | 00-0 |
| | 0,8 | -000 |
| group 1 | 1,5 | 0-01 |
| | 1,9 | -001 |
| | 2,6 | 0-10 |
| | 2,10 | -010 |
| | 8,9 | 100- |
| | 8,10 | 10-0 |
| group 2 | 5,7 | 01-1 |
| | 6,7 | 011- |
| | 6,14 | -110 |
| | 10,14 | 1-10 |

Column III

**No combining**

# Phase I: Find All Prime Implicants

- Combining group 0 and group 1 of Column II.

Column I

| | | |
|---|---|---|
| 0 | 0000 | ✓ |
| 1 | 0001 | ✓ |
| 2 | 0010 | ✓ |
| 8 | 1000 | ✓ |
| 5 | 0101 | ✓ |
| 6 | 0110 | ✓ |
| 9 | 1001 | ✓ |
| 10 | 1010 | ✓ |
| 7 | 0111 | ✓ |
| 14 | 1110 | ✓ |

Column II

| | | |
|---|---|---|
| group 0 | 0,1 | 000- |
| | 0,2 | 00-0 |
| | 0,8 | -000 |
| group 1 | 1,5 | 0-01 |
| | 1,9 | -001 |
| | 2,6 | 0-10 |
| | 2,10 | -010 |
| | 8,9 | 100- |
| | 8,10 | 10-0 |
| group 2 | 5,7 | 01-1 |
| | 6,7 | 011- |
| | 6,14 | -110 |
| | 10,14 | 1-10 |

Column III

**No combining**

# Phase I: Find All Prime Implicants

- Combining group 0 and group 1 of Column II.

Column I

| | | |
|---|---|---|
| 0 | 0000 | ✓ |
| 1 | 0001 | ✓ |
| 2 | 0010 | ✓ |
| 8 | 1000 | ✓ |
| 5 | 0101 | ✓ |
| 6 | 0110 | ✓ |
| 9 | 1001 | ✓ |
| 10 | 1010 | ✓ |
| 7 | 0111 | ✓ |
| 14 | 1110 | ✓ |

Column II

| | | | |
|---|---|---|---|
| group 0 | 0,1 | 000- | ✓ |
| | 0,2 | 00-0 | |
| | 0,8 | -000 | |
| group 1 | 1,5 | 0-01 | |
| | 1,9 | -001 | |
| | 2,6 | 0-10 | |
| | 2,10 | -010 | |
| | 8,9 | 100- | ✓ |
| | 8,10 | 10-0 | |
| group 2 | 5,7 | 01-1 | |
| | 6,7 | 011- | |
| | 6,14 | -110 | |
| | 10,14 | 1-10 | |

• • •

Column III

| | |
|---|---|
| 0,1,8,9 | -00- |

# Phase I: Find All Prime Implicants

- Combining group 0 and group 1 of Column II.

Column I

| | | |
|---|---|---|
| 0 | 0000 | ✓ |
| 1 | 0001 | ✓ |
| 2 | 0010 | ✓ |
| 8 | 1000 | ✓ |
| 5 | 0101 | ✓ |
| 6 | 0110 | ✓ |
| 9 | 1001 | ✓ |
| 10 | 1010 | ✓ |
| 7 | 0111 | ✓ |
| 14 | 1110 | ✓ |

Column II

| | | | |
|---|---|---|---|
| group 0 | 0,1 | 000- | ✓ |
| | 0,2 | 00-0 | ✓ |
| | 0,8 | -000 | ✓ |
| group 1 | 1,5 | 0-01 | |
| | 1,9 | -001 | ✓ |
| | 2,6 | 0-10 | |
| | 2,10 | -010 | ✓ |
| | 8,9 | 100- | ✓ |
| | 8,10 | 10-0 | ✓ |
| group 2 | 5,7 | 01-1 | |
| | 6,7 | 011- | |
| | 6,14 | -110 | |
| | 10,14 | 1-10 | |

Column III

| | |
|---|---|
| 0,1,8,9 | -00- |
| 0,2,8,10 | -0-0 |
| 0,8,1,9 | -00- |
| 0,8,2,10 | -0-0 |

# Phase I: Find All Prime Implicants

- Combining group 1 and group 2 of Column II.

Column I

| | | |
|---|---|---|
| 0 | 0000 | ✓ |
| 1 | 0001 | ✓ |
| 2 | 0010 | ✓ |
| 8 | 1000 | ✓ |
| 5 | 0101 | ✓ |
| 6 | 0110 | ✓ |
| 9 | 1001 | ✓ |
| 10 | 1010 | ✓ |
| 7 | 0111 | ✓ |
| 14 | 1110 | ✓ |

Column II

| | | | |
|---|---|---|---|
| group 0 | 0,1 | 000- | ✓ |
| | 0,2 | 00-0 | ✓ |
| | 0,8 | -000 | ✓ |
| group 1 | 1,5 | 0-01 | |
| | 1,9 | -001 | ✓ |
| | 2,6 | 0-10 | |
| | 2,10 | -010 | ✓ |
| | 8,9 | 100- | ✓ |
| | 8,10 | 10-0 | ✓ |
| group 2 | 5,7 | 01-1 | |
| | 6,7 | 011- | |
| | 6,14 | -110 | |
| | 10,14 | 1-10 | |

Column III

| | |
|---|---|
| 0,1,8,9 | -00- |
| 0,2,8,10 | -0-0 |
| 0,8,1,9 | -00- |
| 0,8,2,10 | -0-0 |

No combining

● ● ●

# Phase I: Find All Prime Implicants

- Combining group 1 and group 2 of Column II.

**Column I**

| | | |
|---|---|---|
| 0 | 0000 | ✓ |
| 1 | 0001 | ✓ |
| 2 | 0010 | ✓ |
| 8 | 1000 | ✓ |
| 5 | 0101 | ✓ |
| 6 | 0110 | ✓ |
| 9 | 1001 | ✓ |
| 10 | 1010 | ✓ |
| 7 | 0111 | ✓ |
| 14 | 1110 | ✓ |

**Column II**

| | | | |
|---|---|---|---|
| group 0 | 0,1 | 000- | ✓ |
| | 0,2 | 00-0 | ✓ |
| | 0,8 | -000 | ✓ |
| group 1 | 1,5 | 0-01 | |
| | 1,9 | -001 | ✓ |
| | 2,6 | 0-10 | ✓ |
| | 2,10 | -010 | ✓ |
| | 8,9 | 100- | ✓ |
| | 8,10 | 10-0 | ✓ |
| group 2 | 5,7 | 01-1 | |
| | 6,7 | 011- | |
| | 6,14 | -110 | ✓ |
| | 10,14 | 1-10 | ✓ |

**Column III**

| | |
|---|---|
| 0,1,8,9 | -00- |
| 0,2,8,10 | -0-0 |
| 0,8,1,9 | -00- |
| 0,8,2,10 | -0-0 |
| 2,6,10,14 | --10 |
| 2,10,6,14 | --10 |

# Phase I: Find All Prime Implicants

- Eliminate **repeated** combinations.

Column I

| | | |
|---|---|---|
| 0 | 0000 | ✓ |
| 1 | 0001 | ✓ |
| 2 | 0010 | ✓ |
| 8 | 1000 | ✓ |
| 5 | 0101 | ✓ |
| 6 | 0110 | ✓ |
| 9 | 1001 | ✓ |
| 10 | 1010 | ✓ |
| 7 | 0111 | ✓ |
| 14 | 1110 | ✓ |

Column II

| | | |
|---|---|---|
| 0,1 | 000- | ✓ |
| 0,2 | 00-0 | ✓ |
| 0,8 | -000 | ✓ |
| 1,5 | 0-01 | |
| 1,9 | -001 | ✓ |
| 2,6 | 0-10 | ✓ |
| 2,10 | -010 | ✓ |
| 8,9 | 100- | ✓ |
| 8,10 | 10-0 | ✓ |
| 5,7 | 01-1 | |
| 6,7 | 011- | |
| 6,14 | -110 | ✓ |
| 10,14 | 1-10 | ✓ |

Column III

| | |
|---|---|
| 0,1,8,9 | -00- |
| 0,2,8,10 | -0-0 |
| ~~0,8,1,9~~ | ~~-00-~~ |
| ~~0,8,2,10~~ | ~~-0-0~~ |
| 2,6,10,14 | --10 |
| ~~2,10,6,14~~ | ~~--10~~ |

# Phase I: Find All Prime Implicants

- Divide column III into groups based on numbers of ones.

Column I

| | | |
|---|---|---|
| 0 | 0000 | ✓ |
| 1 | 0001 | ✓ |
| 2 | 0010 | ✓ |
| 8 | 1000 | ✓ |
| 5 | 0101 | ✓ |
| 6 | 0110 | ✓ |
| 9 | 1001 | ✓ |
| 10 | 1010 | ✓ |
| 7 | 0111 | ✓ |
| 14 | 1110 | ✓ |

Column II

| | | |
|---|---|---|
| 0,1 | 000- | ✓ |
| 0,2 | 00-0 | ✓ |
| 0,8 | -000 | ✓ |
| 1,5 | 0-01 | |
| 1,9 | -001 | ✓ |
| 2,6 | 0-10 | ✓ |
| 2,10 | -010 | ✓ |
| 8,9 | 100- | ✓ |
| 8,10 | 10-0 | ✓ |
| 5,7 | 01-1 | |
| 6,7 | 011- | |
| 6,14 | -110 | ✓ |
| 10,14 | 1-10 | ✓ |

Column III

| | | |
|---|---|---|
| group 0 | 0,1,8,9 | -00- |
| | 0,2,8,10 | -0-0 |
| group 1 | 2,6,10,14 | --10 |

No more combinations in column III are possible, so we stop here.

# Phase I: Find All Prime Implicants

- **<u>Claim</u>**: those products that have NOT been check-marked are prime implicants, because they cannot be combined any more.

| Column I | | |
|---|---|---|
| 0 | 0000 | ✓ |
| 1 | 0001 | ✓ |
| 2 | 0010 | ✓ |
| 8 | 1000 | ✓ |
| 5 | 0101 | ✓ |
| 6 | 0110 | ✓ |
| 9 | 1001 | ✓ |
| 10 | 1010 | ✓ |
| 7 | 0111 | ✓ |
| 14 | 1110 | ✓ |

| Column II | | |
|---|---|---|
| 0,1 | 000- | ✓ |
| 0,2 | 00-0 | ✓ |
| 0,8 | -000 | ✓ |
| 1,5 | 0-01 | ← |
| 1,9 | -001 | ✓ |
| 2,6 | 0-10 | ✓ |
| 2,10 | -010 | ✓ |
| 8,9 | 100- | ✓ |
| 8,10 | 10-0 | ✓ |
| 5,7 | 01-1 | ← |
| 6,7 | 011- | ← |
| 6,14 | -110 | ✓ |
| 10,14 | 1-10 | ✓ |

| Column III | | |
|---|---|---|
| 0,1,8,9 | -00- | ← |
| 0,2,8,10 | -0-0 | ← |
| 2,6,10,14 | --10 | ← |

**Phase I Done!**

# Outline

- Introduction
- Quine-McCluskey Algorithm
  - Phase I
  - Phase II
- Heuristic Method
  - Overview
  - Expand step

# Phase II:
# Select A Minimum Set of Prime Implicants

- Build the **prime implicant chart**.
  - Rows: All prime implicants we obtained in Phase I.
  - Columns: All minterms in the original function.

Minterms

| Prime Implicants | 0 | 1 | 2 | 5 | 6 | 7 | 8 | 9 | 10 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|
| (0,1,8,9) | X | X |  |  |  |  | X | X |  |  |
| (0,2,8,10) | X |  | X |  |  |  | X |  | X |  |
| (2,6,10,14) |  |  | X |  | X |  |  |  | X | X |
| (1,5) |  | X |  | X |  |  |  |  |  |  |
| (5,7) |  |  |  | X |  | X |  |  |  |  |
| (6,7) |  |  |  |  | X | X |  |  |  |  |

Next step: find all **essential** prime implicants.

# Essential Prime Implicants

- **Essential prime implicants** are prime implicants that cover a minterm of the function that no other prime implicants are able to cover.
- How can we find essential prime implicants?
  - Look for minterm that is covered by only one prime implicant.
  - The covering prime implicant is essential.

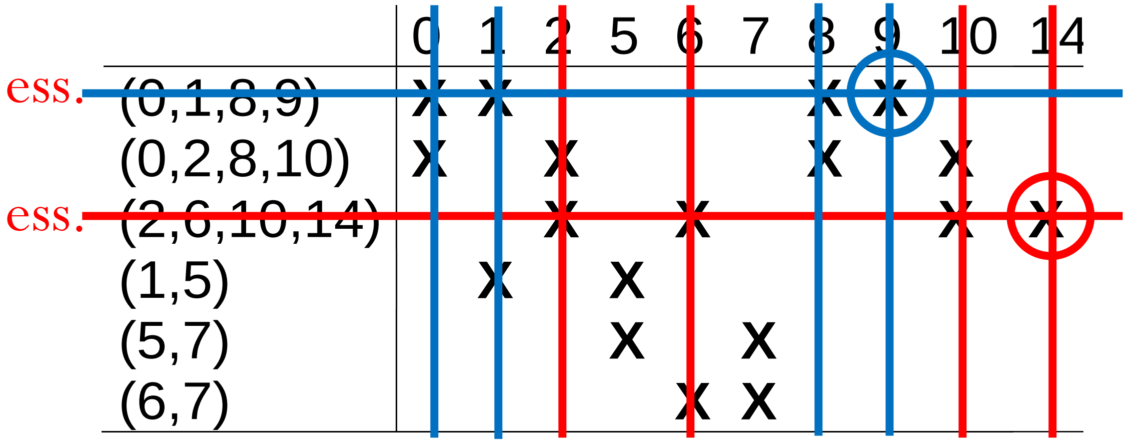
| | | 0 | 1 | 2 | 5 | 6 | 7 | 8 | 9 | 10 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ess. | (0,1,8,9) | X | X | | | | | X | X | | |
| | (0,2,8,10) | X | | X | | | | X | | X | |
| ess. | (2,6,10,14) | | | X | | X | | | | X | X |
| | (1,5) | | X | | X | | | | | | |
| | (5,7) | | | | X | | X | | | | |
| | (6,7) | | | | | X | X | | | | |

# Essential Prime Implicants

- Why **essential prime implicants**?
  - Claim: an SOP with the **minimal** number of literals must contain each essential prime implicant.

# Phase II: Select A Minimum Set of Prime Implicants

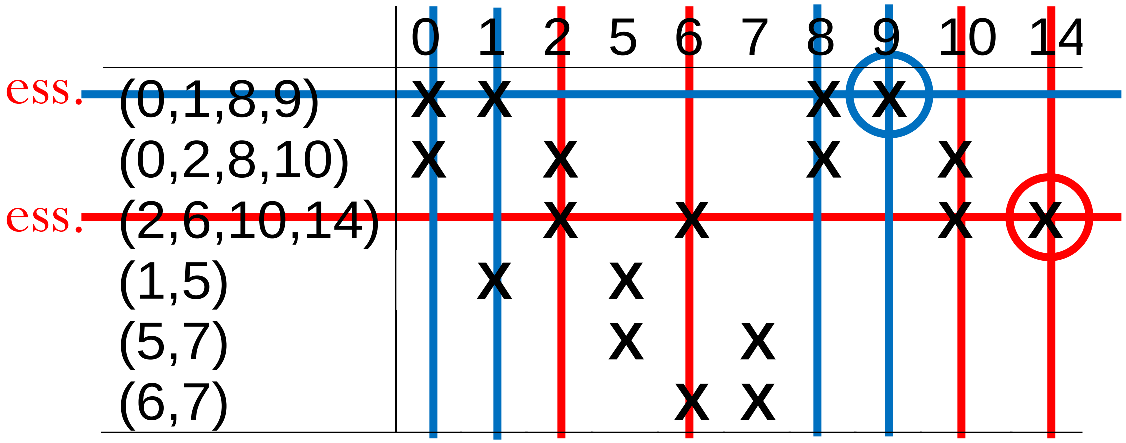
- Once a product term is included in the solution, all the minterms covered by that term are covered.
- Therefore we may now mark the covered minterms and only focus on prime implicants that cover some **not-yet-covered** minterms.



|  |  | 0 | 1 | 2 | 5 | 6 | 7 | 8 | 9 | 10 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ess. | (0,1,8,9) | X | X |  |  |  |  | X | X |  |  |
|  | (0,2,8,10) | X |  | X |  |  |  | X |  | X |  |
| ess. | (2,6,10,14) |  |  | X |  | X |  |  |  | X | X |
|  | (1,5) |  | X |  | X |  |  |  |  |  |  |
|  | (5,7) |  |  |  | X |  | X |  |  |  |  |
|  | (6,7) |  |  |  |  | X | X |  |  |  |  |

Minterms 5 and 7 haven't been covered. We only need to focus on prime implicants (1,5), (5,7), and (6,7).

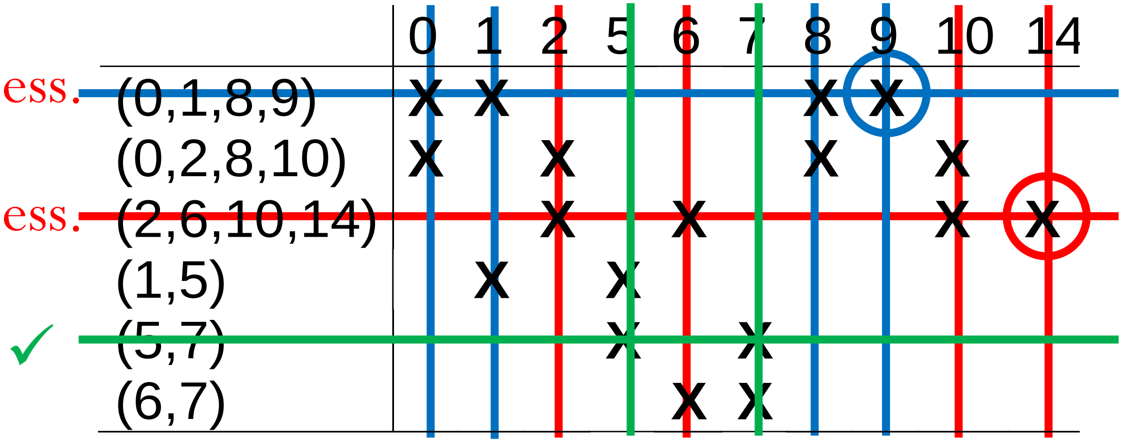
# Phase II:
# Select A Minimum Set of Prime Implicants



|  |  | 0 | 1 | 2 | 5 | 6 | 7 | 8 | 9 | 10 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ess. | (0,1,8,9) | X | X |  |  |  |  | X | X |  |  |
|  | (0,2,8,10) | X |  | X |  |  |  | X |  | X |  |
| ess. | (2,6,10,14) |  |  | X |  | X |  |  |  | X | X |
|  | (1,5) |  | X |  | X |  |  |  |  |  |  |
|  | (5,7) |  |  |  | X |  | X |  |  |  |  |
|  | (6,7) |  |  |  |  | X | X |  |  |  |  |

- Prime implicants (1,5), (5,7), and (6,7) are **non-essential** prime implicants.
    - Now we must choose **enough** non-essential prime implicants to cover the remaining minterms.
    - What strategy should we use?

# Phase II:
# Select A Minimum Set of Prime Implicants



| | | 0 | 1 | 2 | 5 | 6 | 7 | 8 | 9 | 10 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ess. | (0,1,8,9) | X | X | | | | | X | X | | |
| | (0,2,8,10) | X | | X | | | | X | | X | |
| ess. | (2,6,10,14) | | | X | | X | | | | X | X |
| | (1,5) | | X | | X | | | | | | |
| ✓ | (5,7) | | | | X | | X | | | | |
| | (6,7) | | | | | X | X | | | | |

**<u>Done</u>**: the minimal SOP is $f = \bar{b}\bar{c} + c\bar{d} + \bar{a}bd$

- Heuristics: choose the prime implicant that covers the most minterms.
  - Then mark covered minterms.
  - Repeat the above procedure if necessary.

Does this strategy always wok?

# Cyclic Prime Implicant Chart

- Example: $f(a,b,c) = \sum m(0,1,2,5,6,7)$

|   | abc |   |
|---|-----|---|
| 0 | 000 | ✓ |
| 1 | 001 | ✓ |
| 2 | 010 | ✓ |
| 5 | 101 | ✓ |
| 6 | 110 | ✓ |
| 7 | 111 | ✓ |

|     | abc |
|-----|-----|
| 0,1 | 00- |
| 0,2 | 0-0 |
| 1,5 | -01 |
| 2,6 | -10 |
| 5,7 | 1-1 |
| 6,7 | 11- |

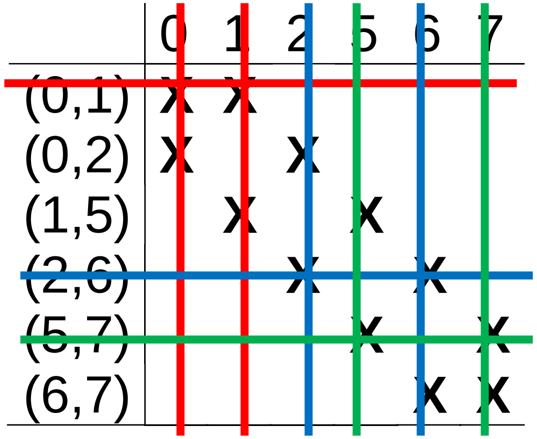
| Prime Implicants | Minterms |   |   |   |   |   |
|---|---|---|---|---|---|---|
|   | 0 | 1 | 2 | 5 | 6 | 7 |
| (0,1) | X | X |   |   |   |   |
| (0,2) | X |   | X |   |   |   |
| (1,5) |   | X |   | X |   |   |
| (2,6) |   |   | X |   | X |   |
| (5,7) |   |   |   | X |   | X |
| (6,7) |   |   |   |   | X | X |

- Are there any essential prime implicants? **No!**
- Further, all prime implicants cover the same number of minterms.
- Then, how shall we proceed? **By trial and error.**

# Cyclic Prime Implicant Chart

| | abc | |
|---|---|---|
| 0 | 000 | ✓ |
| 1 | 001 | ✓ |
| 2 | 010 | ✓ |
| 5 | 101 | ✓ |
| 6 | 110 | ✓ |
| 7 | 111 | ✓ |

| | abc |
|---|---|
| 0,1 | 00- |
| 0,2 | 0-0 |
| 1,5 | -01 |
| 2,6 | -10 |
| 5,7 | 1-1 |
| 6,7 | 11- |



| | 0 | 1 | 2 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| (0,1) | X | X | | | | |
| (0,2) | X | | X | | | |
| (1,5) | | X | | X | | |
| (2,6) | | | X | | X | |
| (5,7) | | | | X | | X |
| (6,7) | | | | | X | X |

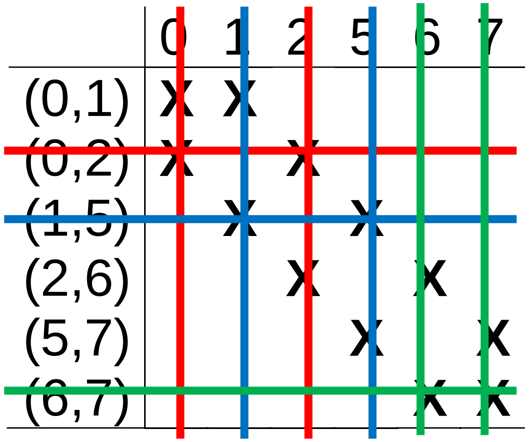
- Pick product (0,1).
- Now products (2,6), (5,7), and (6,7) cover the same most number of minterms.
  - Trial and error again: Pick product (2,6).
- Next: pick product (5,7)

**<u>Done</u>**: the minimal SOP is $f = \bar{a}\bar{b} + b\bar{c} + ac$

# Cyclic Prime Implicant Chart

|   | abc |   |
|---|---|---|
| 0 | 000 | ✓ |
| 1 | 001 | ✓ |
| 2 | 010 | ✓ |
| 5 | 101 | ✓ |
| 6 | 110 | ✓ |
| 7 | 111 | ✓ |

|   | abc |
|---|---|
| 0,1 | 00- |
| 0,2 | 0-0 |
| 1,5 | -01 |
| 2,6 | -10 |
| 5,7 | 1-1 |
| 6,7 | 11- |



|   | 0 | 1 | 2 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| (0,1) | X | X |   |   |   |   |
| (0,2) | X |   | X |   |   |   |
| (1,5) |   | X |   | X |   |   |
| (2,6) |   |   | X |   | X |   |
| (5,7) |   |   |   | X |   | X |
| (6,7) |   |   |   |   | X | X |

- Let’s try another set of prime implicants: Pick product (0,2).
- Now products (1,5), (5,7), and (6,7) cover the same most number of minterms.
  - Trial and error again: Pick product (1,5).
- Next: pick product (6,7)

**<u>Done</u>**: the minimal SOP is $f = \bar{a}\bar{c} + \bar{b}c + ab$

# Cyclic Prime Implicant Chart

$$f = \bar{a}\bar{b} + b\bar{c} + ac$$
$$f = \bar{a}\bar{c} + \bar{b}c + ab$$

- Which minimal form is better?
  - Both are best.

- Often we are interested in examining all minimal forms for a given function.
  - Thus we need an algorithm to do so.

# Petrick's Method

- Given: a prime implicant chart.
- Goal: determine all sum-of-products solutions that contains the **minimal number of product terms**.

- Step 1: Label all the rows in the chart with a Boolean variable

| | | 0 | 1 | 2 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| P1 | (0,1) | X | X | | | | |
| P2 | (0,2) | X | | X | | | |
| P3 | (1,5) | | X | | X | | |
| P4 | (2,6) | | | X | | X | |
| P5 | (5,7) | | | | X | | X |
| P6 | (6,7) | | | | | X | X |

Pi = 1 means the i-th prime implicant is selected

# Petrick's Method

- Step 2: We want to build a function on variables P1, P2, etc. to encode **all** prime implicant covers: for any Pi combination that lets function be 1, it corresponds to a valid cover

| | | 0 | 1 | 2 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| P1 | (0,1) | X | X | | | | |
| P2 | (0,2) | X | | X | | | |
| P3 | (1,5) | | X | | X | | |
| P4 | (2,6) | | | X | | X | |
| P5 | (5,7) | | | | X | | X |
| P6 | (6,7) | | | | | X | X |

- Note: The first column has an **X** in rows P1 and P2.
  - Therefore we must include one of these rows in order to cover minterm 0. Thus the following term must be in P:

$$P1+P2$$

# Petrick’s Method

- Step 2: We want to build a function on variables P1, P2, etc. to encode **all** prime implicant covers: for any Pi combination that lets function be 1, it corresponds to a valid cover

| | | 0 | 1 | 2 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| P1 | (0,1) | X | X | | | | |
| P2 | (0,2) | X | | X | | | |
| P3 | (1,5) | | X | | X | | |
| P4 | (2,6) | | | X | | X | |
| P5 | (5,7) | | | | X | | X |
| P6 | (6,7) | | | | | X | X |

- Note: The second column has an **X** in rows P1 and P3.
  - Therefore we must include one of these rows in order to cover minterm 1. Thus the following term must be in P:

P1+P3

• • •

# Petrick’s Method

| | | 0 | 1 | 2 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| P1 | (0,1) | X | X | | | | |
| P2 | (0,2) | X | | X | | | |
| P3 | (1,5) | | X | | X | | |
| P4 | (2,6) | | | X | | X | |
| P5 | (5,7) | | | | X | | X |
| P6 | (6,7) | | | | | X | X |

P=1 ↔ all minterms are covered

- The final function P is AND of all the sums:

$$P = (P1+P2)(P1+P3)(P2+P4)(P3+P5)(P4+P6)(P5+P6)$$

$$= [(P1+P2)(P1+P3)][(P2+P4)(P4+P6)][(P3+P5)(P5+P6)]$$

$$= (P1+P2\ P3)(P4+P2\ P6)(P5+P3\ P6)$$

$$= P1\ P4\ P5 + P1\ P3\ P4\ P6 + P1\ P2\ P5\ P6 + \cancel{P1\ P2\ P3\ P6}$$

$$+ P2\ P3\ P4\ P5 + \cancel{P2\ P3\ P4\ P6} + \cancel{P2\ P3\ P5\ P6} + P2\ P3\ P6$$

# Petrick’s Method

|  |  | 0 | 1 | 2 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| P1 | (0,1) | X | X |  |  |  |  |
| P2 | (0,2) | X |  | X |  |  |  |
| P3 | (1,5) |  | X |  | X |  |  |
| P4 | (2,6) |  |  | X |  | X |  |
| P5 | (5,7) |  |  |  | X |  | X |
| P6 | (6,7) |  |  |  |  | X | X |

$$P = P1\ P4\ P5 + P1\ P3\ P4\ P6 + P1\ P2\ P5\ P6$$
$$+ P2\ P3\ P4\ P5 + P2\ P3\ P6$$

- What does the above equation mean?
  - It says that to cover all the minterms we must include the terms in lines P1, P4, and P5, **or** we must include lines P1, P3, P4, and P6, **or** we must include lines P1, P2, P5, and P6, …

# Petrick’s Method

| | | 0 | 1 | 2 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| P1 | (0,1) | X | X | | | | |
| P2 | (0,2) | X | | X | | | |
| P3 | (1,5) | | X | | X | | |
| P4 | (2,6) | | | X | | X | |
| P5 | (5,7) | | | | X | | X |
| P6 | (6,7) | | | | | X | X |

P = P1 P4 P5 + P1 P3 P4 P6 + P1 P2 P5 P6
+ P2 P3 P4 P5 + P2 P3 P6

- What are the choices with the minimal number of products?
  - P1, P4, P5 $\Longrightarrow f = \bar{a}\bar{b} + b\bar{c} + ac$
  - **Or** P2, P3, P6 $\Longrightarrow f = \bar{a}\bar{c} + \bar{b}c + ab$

# Quine-McCluskey Algorithm

## Summary

- Two Major Steps:
  1. Find all **prime implicants** of the function.
  2. Use those prime implicants in a **prime implicant chart** to find a **minimal** set of prime implicants that covers the function.
     - **Essential prime implicants**
     - Petrick's method

- Quine-McCluskey algorithm gives **exact** minimal solution.
- … but it has **exponential** complexity on number of inputs.
  - Not practical for large problem.
  - How can we do better?

Heuristic Methods

# Outline

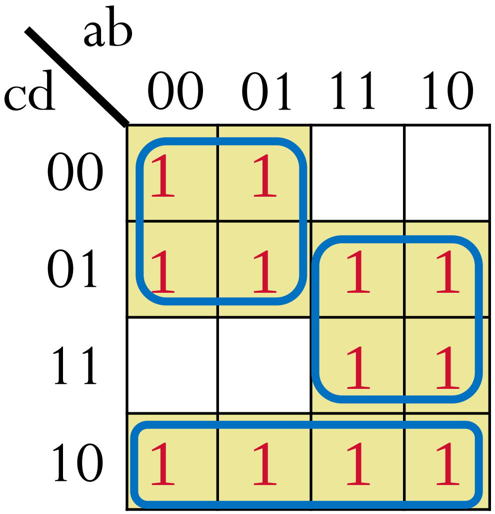
- Introduction
- Quine-McCluskey Algorithm
  - Phase I
  - Phase II
- Heuristic Method
  - Overview
  - Expand step

# Better Strategy

- **Big idea #1**: **Don’t** try for the **best**, **perfect** answer. Just get a **good** answer.

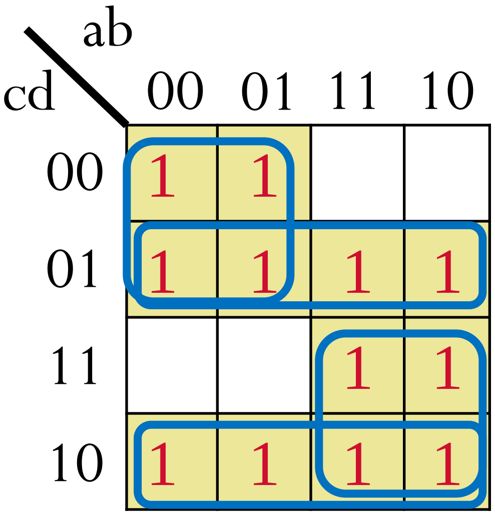
- **Big idea #2**: **Iterative improvement**. From one answer, **reshape** the solution to discover a (possibly better) answer. Continue until no more improvement.

# Example: Best vs "Good Enough"



| cd \ ab | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | | |
| 01 | 1 | 1 | 1 | 1 |
| 11 | | | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 |

**Best**

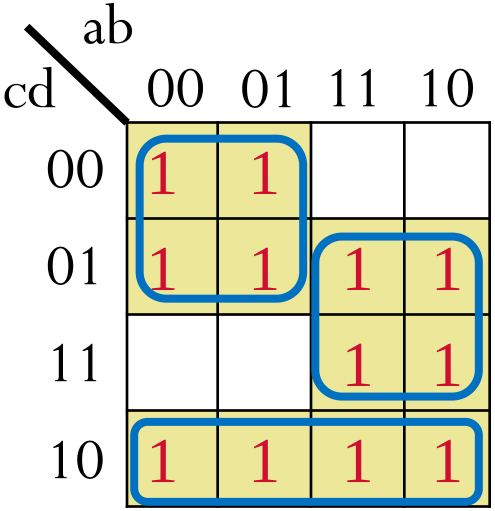


| cd \ ab | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | | |
| 01 | 1 | 1 | 1 | 1 |
| 11 | | | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 |

**Good**

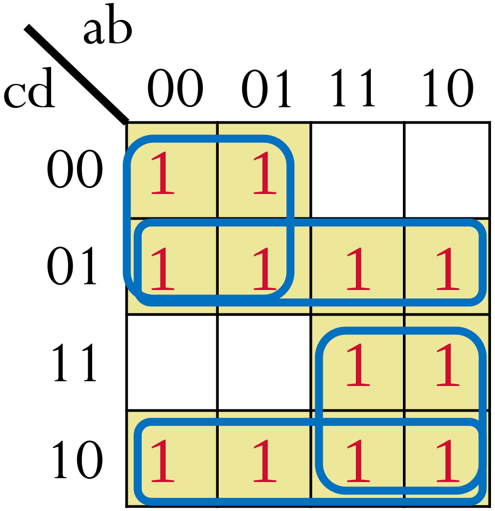
- Comparing the two solutions:
  - Both are made of product terms ("**cubes**") that are "**as big as possible**", i.e., **prime implicants**.
  - We **insist on** this, because best solution is composed of **cover of prime implicants**.

# Example: Best vs “Good Enough”



| cd \ ab | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | | |
| 01 | 1 | 1 | 1 | 1 |
| 11 | | | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 |

**Best**

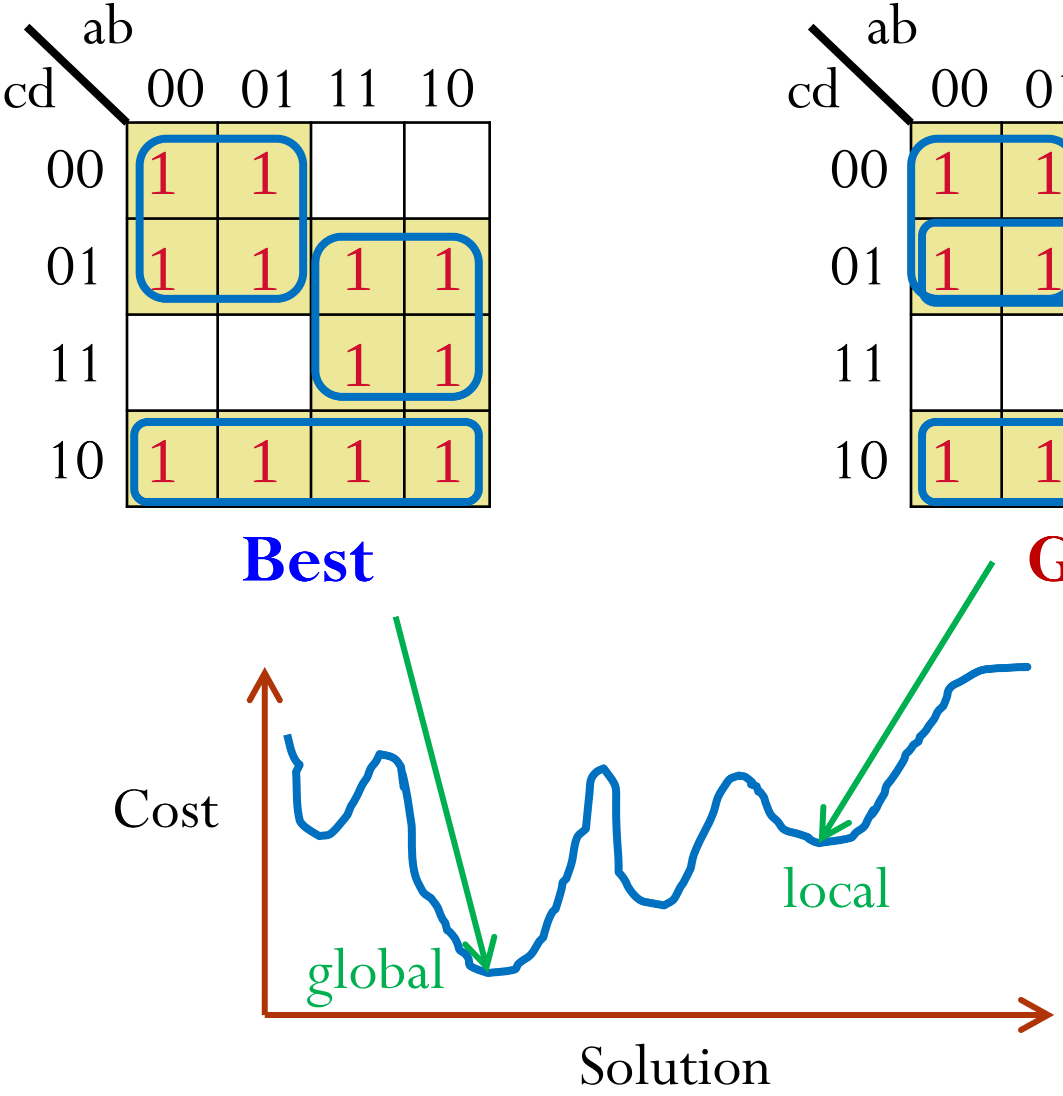


| cd \ ab | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | | |
| 01 | 1 | 1 | 1 | 1 |
| 11 | | | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 |

**Good**

- Note: Neither solution can be improved by **removing** a prime
  - Both solutions are “**irredundant**”.
  - We also **insist on** this.

# Example: Best vs “Good Enough”



| cd \ ab | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | | |
| 01 | 1 | 1 | 1 | 1 |
| 11 | | | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 |

**Best**

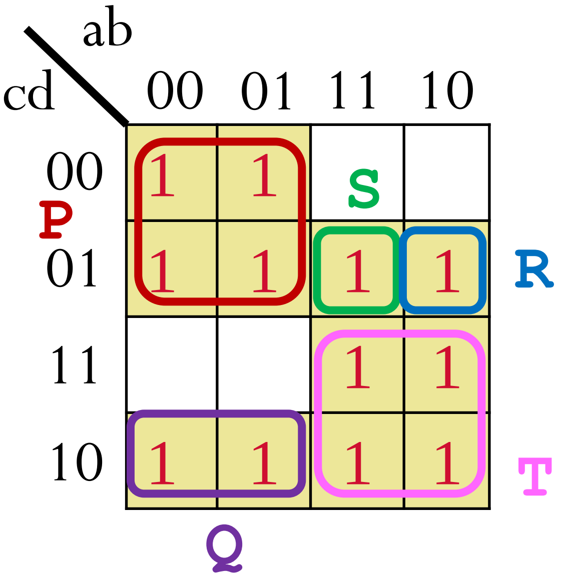
| cd \ ab | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | | |
| 01 | 1 | 1 | 1 | 1 |
| 11 | | | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 |

**Good**

# Heuristic Method: Example

Assume start with a cube list

| abcd | F | label |
|---|---|---|
| 0-0- | 1 | P |
| 0-10 | 1 | Q |
| 1001 | 1 | R |
| 1101 | 1 | S |
| 1-1- | 1 | T |

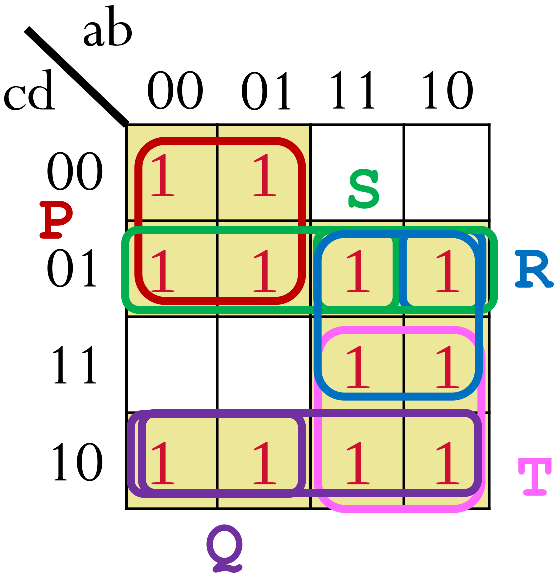


| cd \ ab | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | | |
| 01 | 1 | 1 | 1 | 1 |
| 11 | | | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 |

- Each row defines a **product** (**cube**)
  - It is an **implicant** that lets $F = 1$.
  - Might not be **prime**, but it surely covers all the 1’s.

# Next Step: Expand Each Cube to be Prime

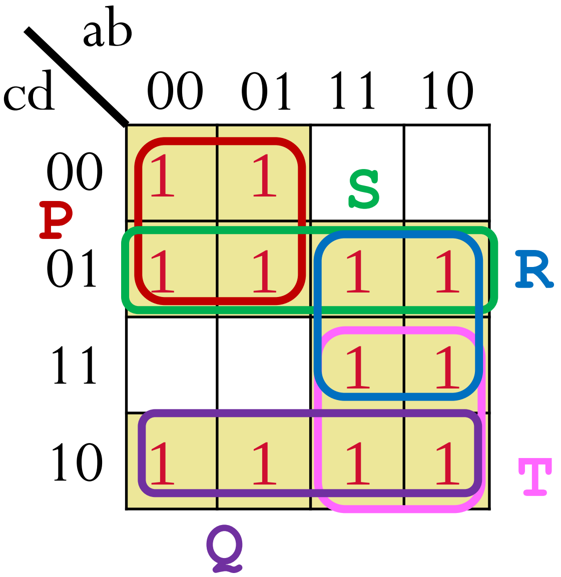
- "Expand" is a **heuristic**, done one cube at a time.
- Make each cube **as big as possible**.
    - 3 of our cubes have now been grown: Q, R, and S.
    - Might have **<u>different</u>** ways to do this for any specific cube…
        - E.g., cube S
- This new solution is a **prime cover**.
- But it might **not be best**, we need do something further.



| abcd | F | label |
|---|---|---|
| 0-0- | 1 | P |
| --10 | 1 | Q |
| 1--1 | 1 | R |
| --01 | 1 | S |
| 1-1- | 1 | T |

# Next Step: Remove Redundant Cubes

- "Irredundant" is a **heuristic**.
  - A cube is **redundant** if we can remove it, and all its 1's are **still** covered by **other** cubes in the rest of the cover.
  - Irredundant operation **<u>removes</u>** redundant cubes in our cover.
- <u>Question</u>: which cube is redundant?
- Assume we remove cube R
  - This new solution is a prime cover.
  - And it is technically "**minimal**" –cannot remove another cube without **uncovering** some 1s.
- But maybe we can still do better…

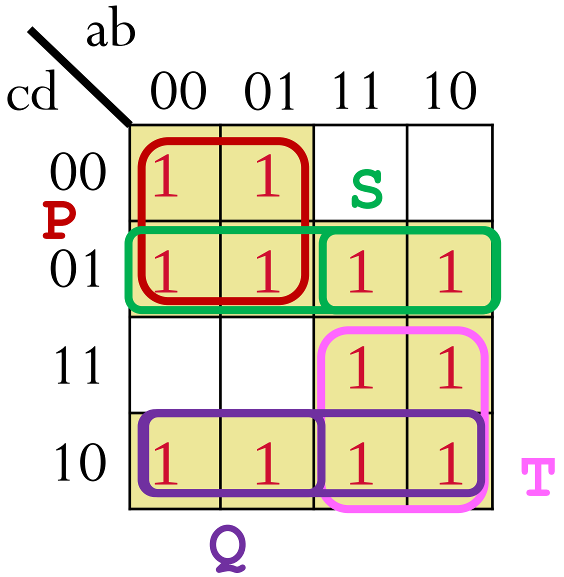


| cd \ ab | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 |  |  |
| 01 | 1 | 1 | 1 | 1 |
| 11 |  |  | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 |

| abcd | F | label |
|---|---|---|
| 0-0- | 1 | P |
| --10 | 1 | Q |
| --01 | 1 | S |
| 1-1- | 1 | T |

# Next Step: Reduce the Prime Cover

- "Reduce" is another **heuristic**.
  - Take each cube, "**shrink it**" as much as possible, but **do not uncover** any 1s.
    - In our example: shrink cubes Q and S.
  - These result cubes may **not** be prime; i.e. this is **not** necessarily a prime cover.
- Surprisingly, **essential** step!
  - This new solution has different **shape**.
  - **Big Idea**: When we expand it again, maybe we get a new, better solution.
  - So, maybe we can still do better…

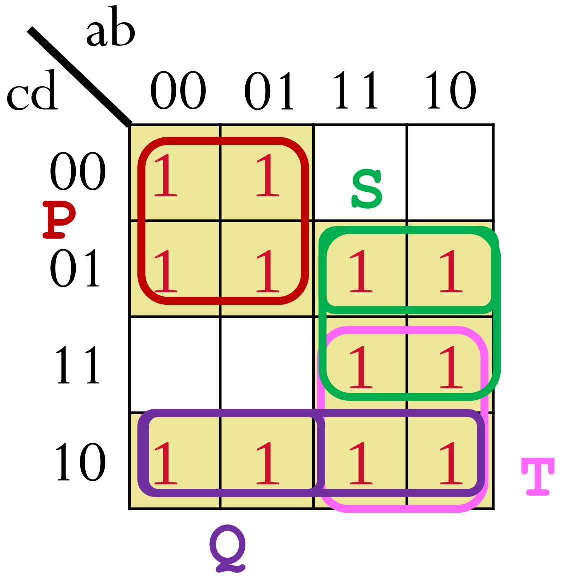


| cd \ ab | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | | |
| 01 | 1 | 1 | 1 | 1 |
| 11 | | | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 |

| abcd | F | label |
|---|---|---|
| 0-0- | 1 | P |
| 0-10 | 1 | Q |
| 1-01 | 1 | S |
| 1-1- | 1 | T |

# Next Step: Expand Cubes Again

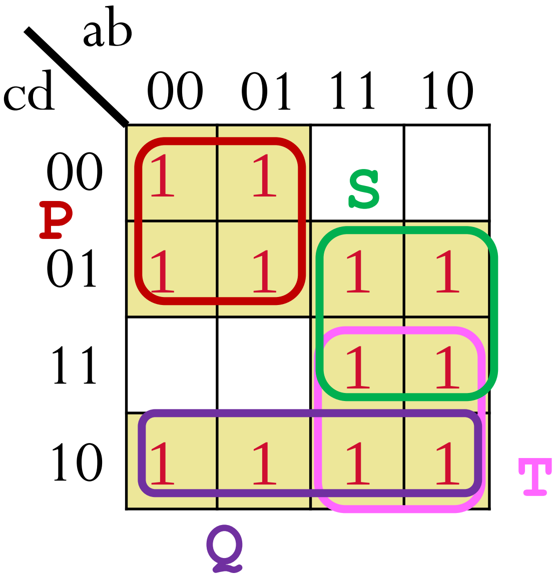
- Same "Expand" **heuristic**.
  - But it is starting from a **different cover**, so can get a **different** answer!
  - Take each cube and "**expand**" it to make it prime, and also…
  - …try to cover other cubes, to make them **redundant** (so we kill them later).
- In this example, we expand Q and S.



| abcd | F | label |
|---|---|---|
| 0-0- | 1 | P |
| --10 | 1 | Q |
| 1--1 | 1 | S |
| 1-1- | 1 | T |

# Next Step: Check Redundant Again

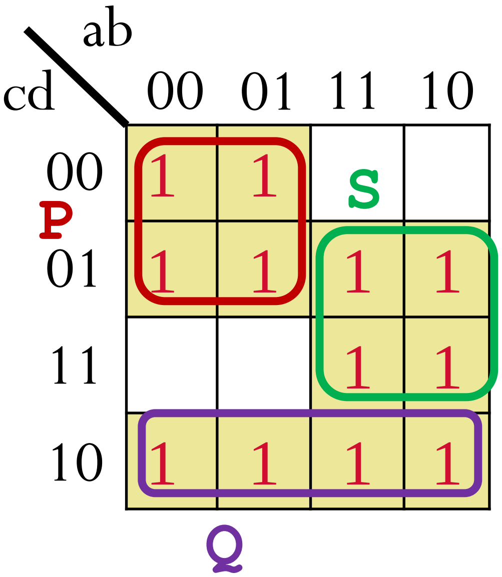
- Same “Irredundant” **heuristic**.
  - But it is starting from a **different cover**, so can get a **different** answer!
- In this example: we can kill another cube T, it is **redundant**.
  - After this, the cover is again **prime** and **irredundant**. Can’t remove anything to make it better (smaller).



| abcd | F | label |
|---|---|---|
| 0-0- | 1 | P |
| --10 | 1 | Q |
| 1--1 | 1 | S |

# This Result Is Really Good!

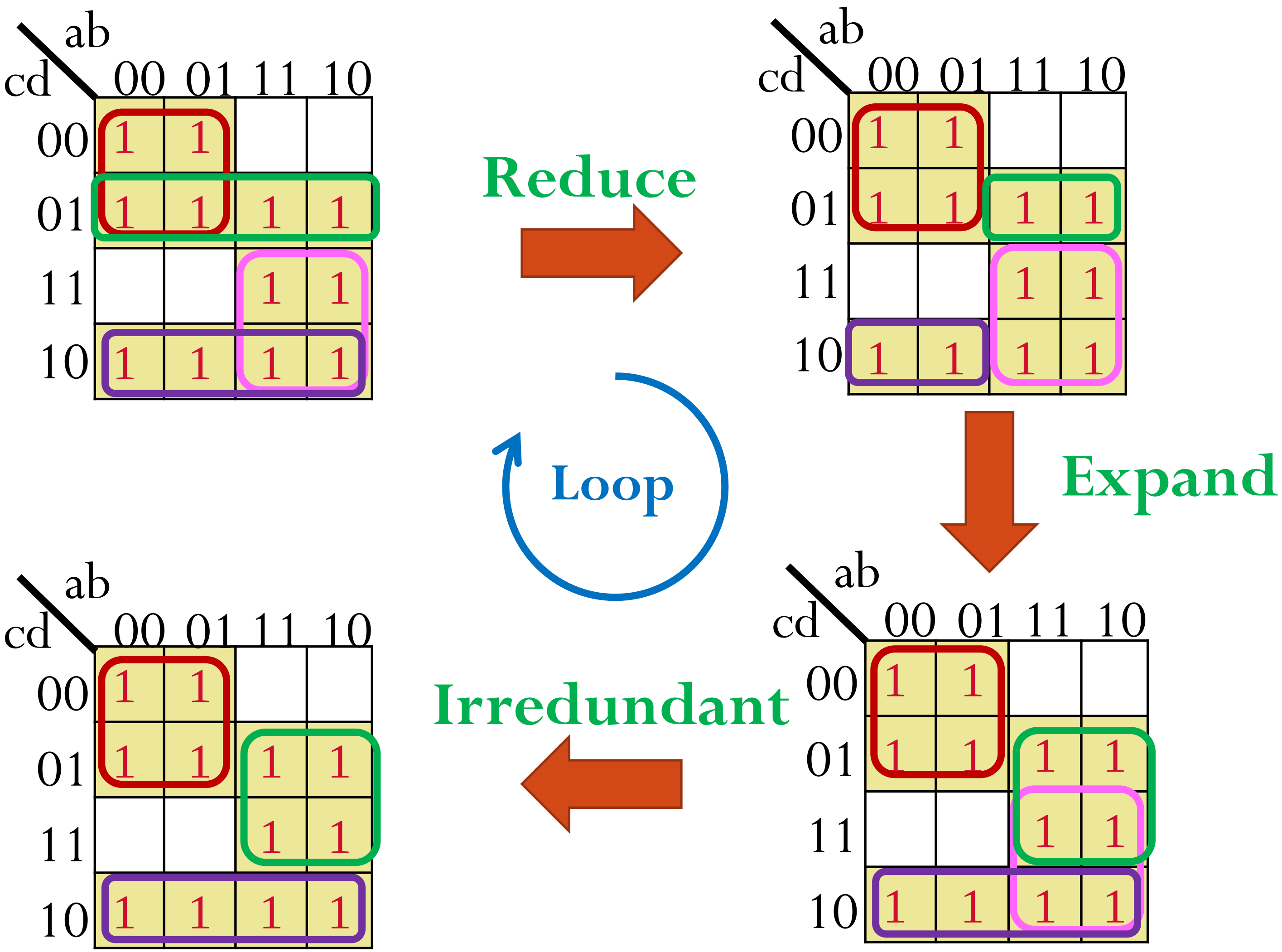
- Got lucky: this is the **BEST** answer.
  - This will **not** generally happen.
  - But we can guarantee a **prime, irredundant, and minimal** solution.
  - And it turns out in practice, that this **iterative** improvement that "**reshapes**" the cover produces **excellent solutions**.



| cd \ ab | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 |  |  |
| 01 | 1 | 1 | 1 | 1 |
| 11 |  |  | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 |

| abcd | F | label |
|---|---|---|
| 0-0- | 1 | P |
| --10 | 1 | Q |
| 1--1 | 1 | S |

# Famous: Reduce-Expand-Irredundant Loop



| cd \ ab | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | | |
| 01 | 1 | 1 | 1 | 1 |
| 11 | | | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 |

| cd \ ab | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | | |
| 01 | 1 | 1 | 1 | 1 |
| 11 | | | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 |

| cd \ ab | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | | |
| 01 | 1 | 1 | 1 | 1 |
| 11 | | | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 |

| cd \ ab | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | | |
| 01 | 1 | 1 | 1 | 1 |
| 11 | | | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 |

# Famous Tool: ESPRESSO

- For **two-level minimization**
- Started at IBM, finished at Berkeley.
- References:
  - Brayton, Hachtel, McMullen, Sangiovanni-Vincentelli, Logic Minimization Algorithms for VLSI Synthesis, Kluwer Academic Press, 1984.
  - Richard L. Rudell,, "Multiple-Valued Logic Minimization for PLA Synthesis", U. California Berkeley M.S. Thesis.

# Outline

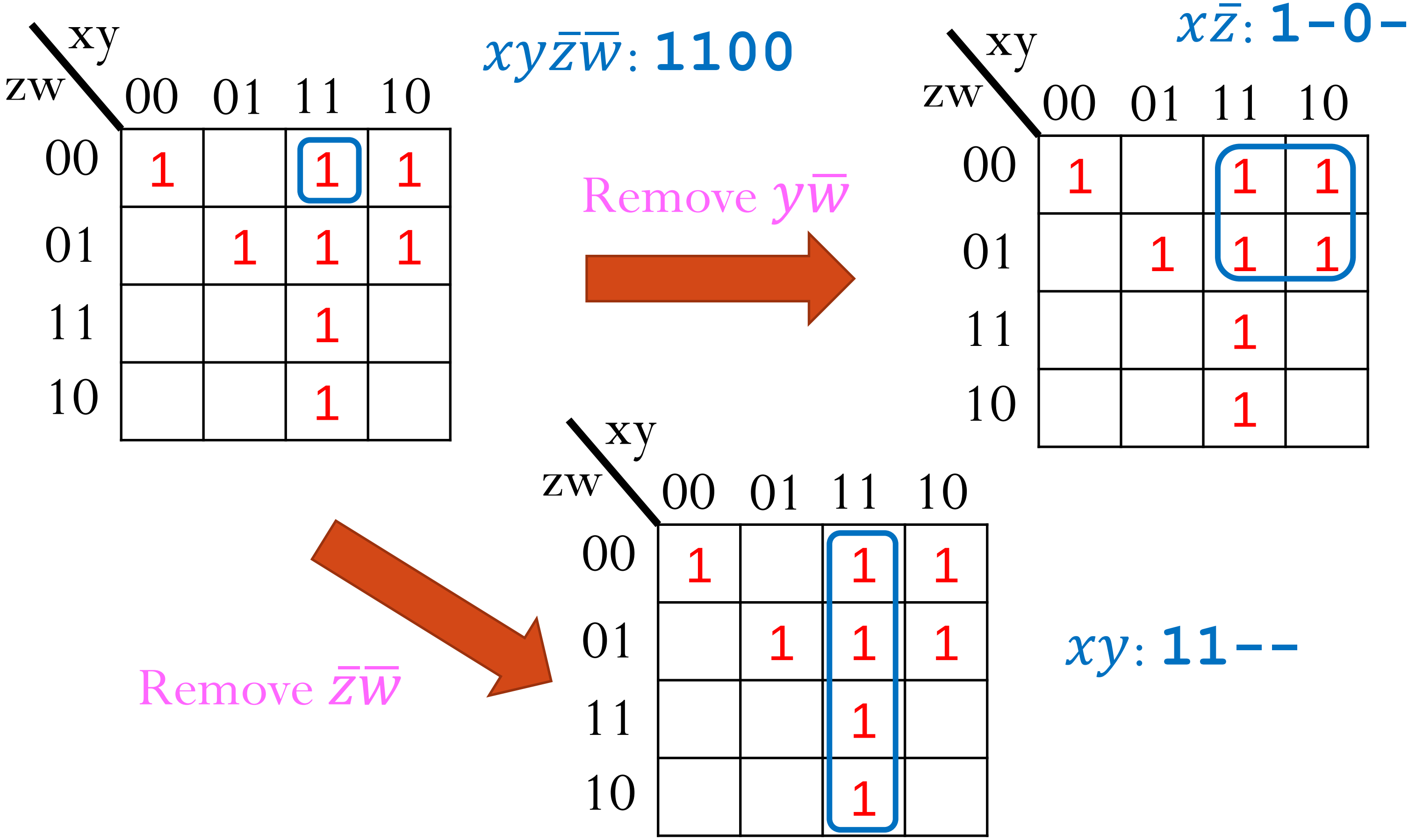
- Introduction

- Quine-McCluskey Algorithm
  - Phase I
  - Phase II

- Heuristic Method
  - Overview
  - Expand step

# Lets Look (Briefly) At One Step: Expand

- What does "expand a cube" mean?
  - **Remove variables** from cube.



$xy\bar{z}\bar{w}$: **1100**

| zw \ xy | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 |  | 1 | 1 |
| 01 |  | 1 | 1 | 1 |
| 11 |  |  | 1 |  |
| 10 |  |  | 1 |  |

Remove $y\bar{w}$

$x\bar{z}$: **1-0-**

| zw \ xy | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 |  | 1 | 1 |
| 01 |  | 1 | 1 | 1 |
| 11 |  |  | 1 |  |
| 10 |  |  | 1 |  |

Remove $\bar{z}\bar{w}$

| zw \ xy | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 |  | 1 | 1 |
| 01 |  | 1 | 1 | 1 |
| 11 |  |  | 1 |  |
| 10 |  |  | 1 |  |

$xy$: **11--**

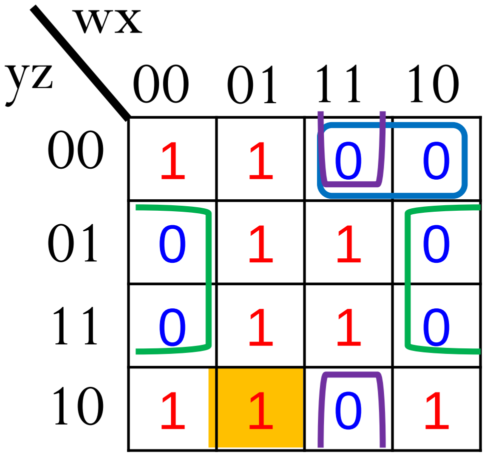
# Expand: Transform into a Covering Problem

- Here is the most basic **Covering Problem**:
  - Given matrix of R rows and C columns. Matrix has 1s and 0s in it. (Only show the 1s.)
  - Choose **smallest set of rows** so that, using only these rows, **every column** has **at least** a single 1 in them – i.e., every column is **"covered" by the selected rows**.
  - Very good **heuristics** to get decent, fast solutions

| | C1 | C2 | C3 | C4 | C5 |
|---|---|---|---|---|---|
| R1 | | 1 | | | |
| R2 | 1 | 1 | 1 | | |
| R3 | | | 1 | | 1 |
| R4 | | 1 | | 1 | 1 |

# Expand: the Blocking Matrix

- Expand = a **Covering Problem** on the **Blocking Matrix**
  - **First**: Given function F, build a **cube cover** of the **0s** in F (called the **OFF Set**).
  - **Why**: We need to know what our cube **cannot touch** when it expands.
  - **How**: **URP Complement** of the starting cover of the function! (this is exactly our Programming Assignment #2.)



| yz \ wx | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | 0 | 0 |
| 01 | 0 | 1 | 1 | 0 |
| 11 | 0 | 1 | 1 | 0 |
| 10 | 1 | 1 | 0 | 1 |

$$F = \bar{w}\bar{y}\bar{z} + xz + \bar{x}y\bar{z} + \bar{w}xy\bar{z}$$

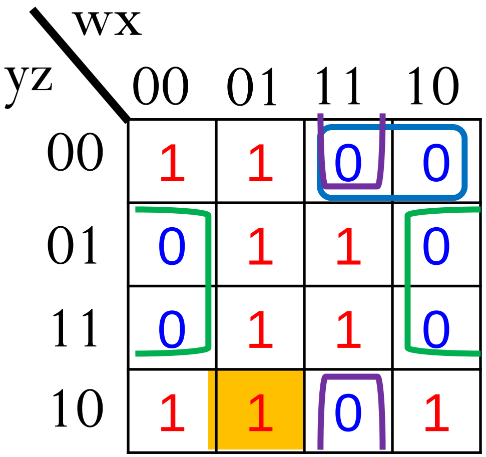
**URP Complement**

**OFF Set:** $\bar{F} = \bar{x}z + wx\bar{z} + w\bar{y}\bar{z}$

Assume we want to expand $\bar{w}xy\bar{z}$

# Next: Build the Blocking Matrix

- Blocking matrix is a binary matrix structured as follows:
  - One row for **each variable** in the cube you are trying to expand.
  - One column for **each cube** in the cover of the **OFF set**.
    - May have **<u>many</u>** columns.



| yz \ wx | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | 0 | 0 |
| 01 | 0 | 1 | 1 | 0 |
| 11 | 0 | 1 | 1 | 0 |
| 10 | 1 | 1 | 0 | 1 |

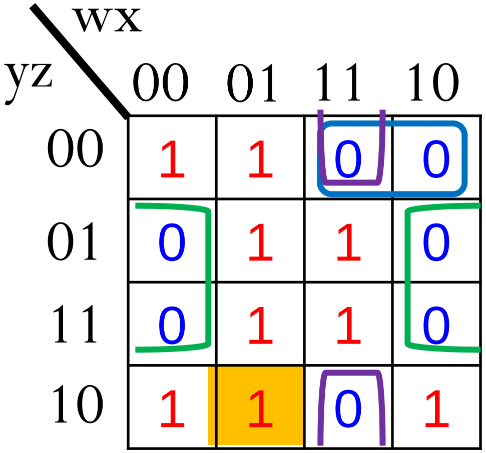
To expand $\bar{w}xy\bar{z}$

**OFF Set:** $\bar{F} = \bar{x}z + wx\bar{z} + w\bar{y}\bar{z}$

| | $\bar{x}z$ | $wx\bar{z}$ | $w\bar{y}\bar{z}$ |
|---|---|---|---|
| $\bar{w}$ | | | |
| $x$ | | | |
| $y$ | | | |
| $\bar{z}$ | | | |

# What to Fill in the Blocking Matrix?

- If the variable in the cube to be expanded (**row**) ≠ **polarity** of variable in the cube of the OFF cover (**column**), put a "**1**".
- If the variable in the cube to be expanded (**row**) = **polarity** of variable in the cube of the OFF cover (**column**), put a "**0**".
- If the variable in the cube to be expanded (**row**) **does not show** in the cube of the OFF cover (**column**), i.e., **don't care**, put a "**0**".



| yz \ wx | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | 0 | 0 |
| 01 | 0 | 1 | 1 | 0 |
| 11 | 0 | 1 | 1 | 0 |
| 10 | 1 | 1 | 0 | 1 |

To expand $\bar{w}xy\bar{z}$

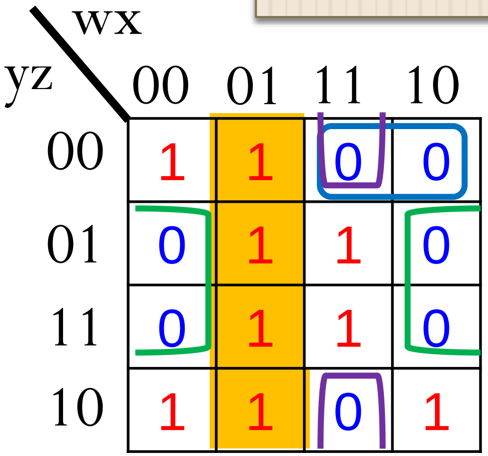
**OFF Set:** $\bar{F} = \bar{x}z + wx\bar{z} + w\bar{y}\bar{z}$

| | $\bar{x}z$ | $wx\bar{z}$ | $w\bar{y}\bar{z}$ |
|---|---|---|---|
| $\bar{w}$ | 0 | 1 | 1 |
| $x$ | 1 | 0 | 0 |
| $y$ | 0 | 0 | 1 |
| $\bar{z}$ | 1 | 0 | 0 |

# What does “1” in Blocking Matrix Mean?

- **Claim**: if all the row variables you have kept **cover each column**, then **product** of these variables is a **legal cube expansion**.
  - Why?

- For any cube $c \in \overline{F}$, these is a kept literal $l$ let $l \cdot c = 0$
- $\Pi(kept\ literal) \cdot \overline{F} = 0 \Rightarrow \Pi(kept\ literal) \subseteq F$



| yz \ wx | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | 0 | 0 |
| 01 | 0 | 1 | 1 | 0 |
| 11 | 0 | 1 | 1 | 0 |
| 10 | 1 | 1 | 0 | 1 |

To expand $\overline{w}xy\bar{z}$

**OFF Set:** $\bar{F} = \bar{x}z + wx\bar{z} + w\bar{y}\bar{z}$

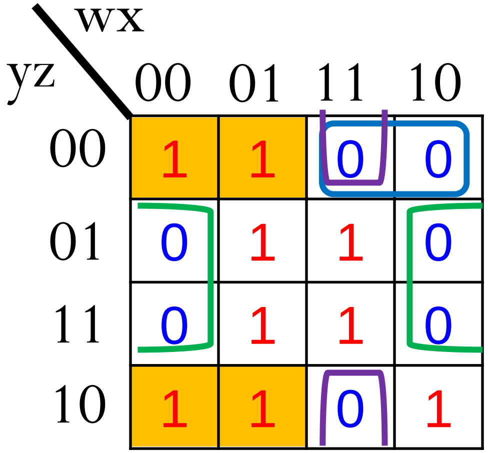
|  | $\bar{x}z$ | $wx\bar{z}$ | $w\bar{y}\bar{z}$ |
|---|---|---|---|
| $\overline{w}$ |  | 1 | 1 |
| $x$ | 1 |  |  |
| $y$ |  |  | 1 |
| $\bar{z}$ | 1 |  |  |

Keep rows $\overline{w}$ and $x$

**A cover!**

# What does “1” in Blocking Matrix Mean?

- **Claim**: if all the row variables you have kept **cover each column**, then **product** of these variables is a **legal cube expansion**.



| yz \ wx | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | 0 | 0 |
| 01 | 0 | 1 | 1 | 0 |
| 11 | 0 | 1 | 1 | 0 |
| 10 | 1 | 1 | 0 | 1 |

To expand $\bar{w}xy\bar{z}$

**OFF Set:** $\bar{F} = \bar{x}z + wx\bar{z} + w\bar{y}\bar{z}$

|  | $\bar{x}z$ | $wx\bar{z}$ | $w\bar{y}\bar{z}$ |
|---|---|---|---|
| $\bar{w}$ |  | 1 | 1 |
| $x$ | 1 |  |  |
| $y$ |  |  | 1 |
| $\bar{z}$ | 1 |  |  |

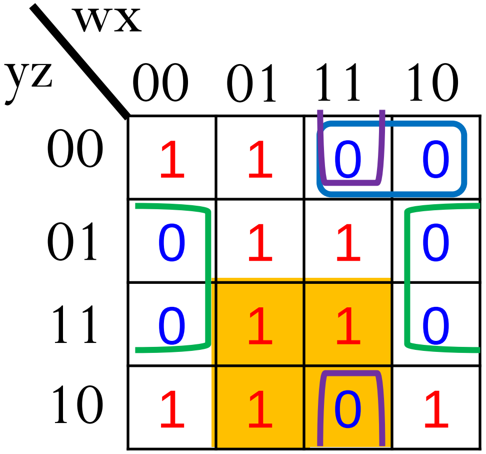
Another example: keep rows $\bar{w}$ and $\bar{z}$

**A cover!**

# What does “1” in Blocking Matrix Mean?

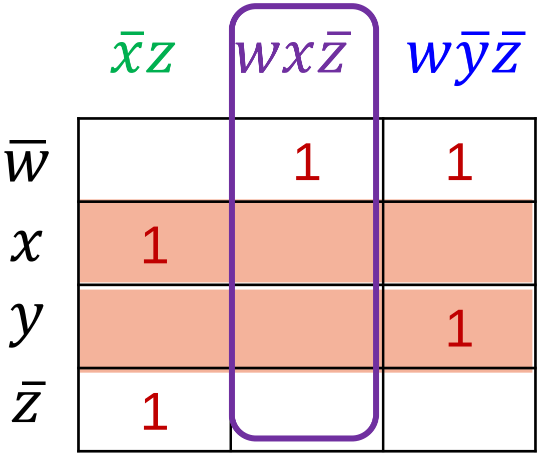
- **On the other hand**, if the set of row variables you have kept is **not** a **cover of all columns**, then **product** of these variables is **not** an **legal cube expansion**.
  - Because it may touch some cubes in the **OFF set**.

**OFF Set:** $\bar{F} = \bar{x}z + wx\bar{z} + w\bar{y}\bar{z}$



| yz \ wx | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | 0 | 0 |
| 01 | 0 | 1 | 1 | 0 |
| 11 | 0 | 1 | 1 | 0 |
| 10 | 1 | 1 | 0 | 1 |

To expand $\bar{w}xy\bar{z}$



|  | $\bar{x}z$ | $wx\bar{z}$ | $w\bar{y}\bar{z}$ |
|---|---|---|---|
| $\bar{w}$ |  | 1 | 1 |
| $x$ | 1 |  |  |
| $y$ |  |  | 1 |
| $\bar{z}$ | 1 |  |  |

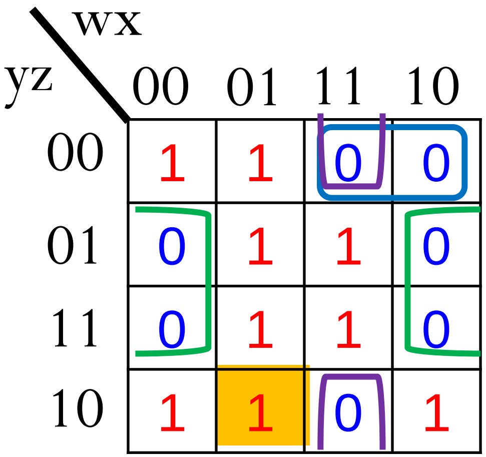
Keep rows $x$ and $y$

**Not a cover!**

Indeed, the expansion touches $wx\bar{z}$

# How to Expand to A Prime Implicant?

- **Keep as few variables as possible** from the initial cube, without touch any cubes in the **OFF set**.
- **Equivalent**: Find **smallest** set of rows that covers each column.
    - This is the **covering problem**. (Can be solved fast by heuristic.)



| yz \ wx | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | 0 | 0 |
| 01 | 0 | 1 | 1 | 0 |
| 11 | 0 | 1 | 1 | 0 |
| 10 | 1 | 1 | 0 | 1 |

To expand $\bar{w}xy\bar{z}$

**OFF Set:** $\bar{F} = \bar{x}z + wx\bar{z} + w\bar{y}\bar{z}$

|  | $\bar{x}z$ | $wx\bar{z}$ | $w\bar{y}\bar{z}$ |
|---|---|---|---|
| $\bar{w}$ |  | 1 | 1 |
| $x$ | 1 |  |  |
| $y$ |  |  | 1 |
| $\bar{z}$ | 1 |  |  |

Either

$\bar{w}$ and $x$,

or

$\bar{w}$ and $\bar{z}$

# ESPRESSO: Collection of Elegant Heuristics

- **Reduce-Expand-Irredundant loop**
  - **Reduce**: Rank cubes in a clever order and reduce them individually.
  - **Expand**: Rank cubes in the opposite of the above clever order and expand each individually as a covering problem.
  - **Irredundant**: A clever recursive algorithm + a clever covering problem.
  - And a bunch of other interesting steps we did not mention…

# Other Thing Can Do

- Minimize **several** functions at the **same time**.
  - Each function will be reduced to a 2-level form.
  - But some product terms (AND gates) will be **shared**.
  - This means: make this AND product once in hardware, connect its output to many OR gates to sum the product into other functions. Can save a lot of hardware this way.
- Handle conventional **Don’t Cares**.
  - Can specify a row of the truth table as being a “**Don’t Care**”.
  - Means the hardware can make a 1 or a 0 as output for this input – you don’t care.
  - Let algorithm choose 0 vs 1 output to make better, smaller circuit.

# How Well Does All This Work?

- Fabulous: Very **fast**, very **robust**
- Where does ESPRESSO spend its time?
  - Complement 14% (big if there are lots of cubes in cover)
  - Expand 29% (depends on of size of complement)
  - Irredundant 12%
  - Reduce 8%
  - Essentials 13% (some primes must be in answer; find them first)
  - Various optimizations 22% (special case optimizations)
- How fast?
  - Usually $< 5$ reduce-expand-irredundant iterations; often converges in just 1-2.
  - Thousands of cubes, tens of thousands of literals: $\ll 1$ CPU second.

# Summary

- 2-level logic synthesis uses **heuristics** to find good solutions.
  - Not "best", but instead "good enough".
  - Minimal (not minimum), prime, irredundant.
  - Famous idea: iterative improvement – **reduce-expand-irredundant loop**.
  - All done with PCN cube lists, covering matrices, and recursive ideas.
- But, not every piece of logic is implemented in 2-level form.
- Next: **Multi-level logic**