What if we have 5 input variables?

\[ V = 0 \]

\[ V = 1 \]
Example with 5 variables

\[ F = \sum_{V,W,X,Y,Z} (0,1,2,3,4,7,15,16,20,23,29,31) \]

What if we have 6 input variables?

Two most-significant bits (UV) are used to select which of the 4-variable maps is being used and the WXYZ bits are used to select the entry in the 4-variable Karnaugh map.
Six input variables — another view

Two most-significant bits (UV) are used to select which of the 4-variable maps is being used and the WXYZ bits are used to select the entry in the 4-variable Karnaugh map.

Definitions

- **Minimal sum** of \( F \) — a sum-of-products such that
  - No sum-of-products for \( F \) has fewer product terms
  - Any sum-of-products with the same # of product terms has \( \geq \) literals

- **Prime Implicant** of \( F \) — a normal product term \( P \) that implies \( F \), s.t. if any variable removed from \( P \) then \( P^* \) doesn’t imply \( F \)

- **Complete sum** — the sum of all prime implicants of \( F \)

- **Distinguished 1-cell** — an input combination covered by only one prime implicant

- **Essential prime implicant** — a prime implicant that covers \( \geq 1 \) distinguished 1-cells → it must be included in the minimal sum!
Example of Prime Implicants

As seen, both prime implicants must be included in the minimal sum to cover all of the 1-cells.

Another example

Essential prime implicants (circled with thicker lines)
Algorithm for minimum \textit{sum-of-product} from K-maps

1. Circle all prime implicants.
2. Identify and select the essential prime implicants for cover.
3. Cover the 1-cells not covered by essential prime implicants.

\textit{Minimum product-of-sum}

1. Represent the ones of \( F' \) in the K-map.
2. Find the minimum SoP of \( F' \).
3. Complement the obtained expression by applying DeMorgan theorem.

---

Example: All Prime Implicants Essential

\[ F = \sum_{W,X,Y,Z} (2,3,4,5,6,7,11,13,15) \]

\[ F = W'X + W'Y + XZ + YZ \]

\( \rightarrow \) all prime implicants are included in the minimal sum
Few or None Essential Prime Implicants

• IF there are no essential prime implicants, or essential prime implicants do not cover all 1-cells,

• THEN select nonessential prime implicants to form a complete minimum-cost cover

• Selection Heuristics explained next …

Too Few Essential Prime Implicants

Select nonessential prime implicant $W' \cdot Z$ over $X' \cdot Y \cdot Z$ because it has fewer inputs \to lower cost
More Definitions

(for more complex cases of too few essential prime implicants)

- **Eclipse**: Given two prime implicants P, Q
  
P eclipses Q if P covers \( \geq 1 \)-cells covered by Q

- **Secondary essential prime implicant**: eclipses other prime implicants

---

Too Few Essential Prime Implicants

(2)

\[ F = \sum_{W,X,Y,Z}(2,6,7,9,13,15) \]

\[ F = W^*Y^*Z + W^*YZ + XYZ \]

\(X \cdot Y \cdot Z\) eclipses the other two prime implicants, therefore it is a secondary essential prime implicant → must be included in the minimal sum
No Essential Prime Implicants

No distinguished 1-cells $\Rightarrow$ No essential prime implicants!

Branching Heuristic

1. Starting w/ any 1-cell, arbitrarily select one prime implicant covering it

2. Include this p.i. as if it were essential and find a tentative minimal sum-of-products

3. Repeat the process, for all other prime implicants covering the starting 1-cell
   - Generates a different tentative minimal sum for each starting point

4. Finally, compare all tentative minimal sums and select the truly minimal
**Branching Heuristic Example**

Two minimal sums:

\[ F = W \cdot X \cdot Z + W \cdot Y \cdot Z + X \cdot Y \cdot Z \]

\[ F = X \cdot Y \cdot Z + W \cdot X \cdot Z + W \cdot Y \cdot Z \]

---

**Incompletely Specified Functions**

(“Don’t-Care” Input Combinations)

*Don’t-cares*: output doesn’t matter for such input combinations (never occur in normal operation).

Example: Detect the prime numbers to ten, input is always a BCD digit.

\[ F = \Sigma_{N_2,N_1,N_0} (1,2,3,5,7) + d(10,11,12,13,14,15) \]

(don’t cares, a.k.a. \(d\)-set)
Modified procedure for circling sets of 1’s  
(prime implicants)

• Allow d’s to be included when circling sets of 1’s, to make the sets as large as possible.
  • This reduces the number of variables in the corresponding prime implicants.
  • Two such prime implicants ($N_2 \cdot N_0$ and $N_2' \cdot N_1$) appear in the example.

• Do not circle any sets that contain only d’s.
  • Including the corresponding product term in the function would unnecessarily increase its cost.
  • Two such product terms ($N_3 \cdot N_2$ and $N_3 \cdot N_1$) are circled in the example.

• Reminder: As usual, do not circle any 0’s

Incompletely Specified Functions  
("Don’t-Care" Input Combinations)

*Don’t-cares:* output doesn’t matter for such input combinations (never occur in normal operation).  
Example: Detect the prime numbers to ten, input is always a BCD digit.

\[
F = \sum N_3, N_2, N_0 (1,2,3,5,7) + d(10,11,12,13,15) \\
(don't cares, a.k.a. d-set)
\]

\[
F = N_3', N_0 + N_2', N_1
\]
Don't Cares and Product-of-Sums Minimization

For Product-of-Sums (PoS) minimization, apply the same technique as for Sum-of-Products (SoP) and the principle of duality.

\[ F = \sum_{W,X,Y,Z} (4,5,13,15) + d(2,3,7,9,14) \]

SoP Minimization and Inverting the Result

\[ F = (X')' \cdot (W \cdot Z')' \cdot (Y \cdot Z')' = X \cdot (W+Z) \cdot (W+Y') \]

(using DeMorgan's theorem)
### Lots of Possibilities

- Can follow a “dual” procedure to find minimal products-of-sums (OR-AND realization)
- Can modify procedure to handle don’t-care input combinations.
- Can draw Karnaugh maps with up to six variables.

### Real-World Logic Design

- Lots more than 6 inputs --can’t use Karnaugh maps
- Design correctness more important than gate minimization
  - Use "higher-level language" to specify logic operations
- Use programs to manipulate logic expressions and minimize logic
- ABEL — developed for PLDs
- VHDL, Verilog — developed for ASICs