Below is a **step‐by‐step** walkthrough showing how to build and simplify a 3‐variable K‐map for the output . We assume our inputs are , and we have a truth table giving us for each combination of these inputs.

**1. Set Up the K‐Map**

1. **Determine K‐map size**:
   * With three inputs, the K‐map has cells.
2. **Label the rows and columns**:
   * Put on the rows (top row for , bottom row for ).
   * Use **Gray code** for the columns with in the order 00, 01, 11, 10.

Your blank K‐map will look like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | (X1X0) 00 | 01 | 11 | 10 |
| X2 = 0 |  |  |  |  |
| X2 = 1 |  |  |  |  |

**2. Fill the K‐Map Using the Truth Table**

1. **Go row by row** through your truth table for .
2. **Check**  for each row:
   * If , place a 1 in the K‐map cell that matches .
   * If , place a 0.
3. **Repeat until all 8 cells** are filled.

**Tip**: Each row in the truth table tells you exactly one cell in the K‐map to fill. The row where goes in the top‐left cell (, ), etc.

**3. Identify Groups of 1’s**

Once you’ve placed all the 1s and 0s:

1. **Look for adjacent 1‐cells**—in powers of two (1, 2, 4, 8). Cells are considered adjacent if they differ by only **one** input bit.
2. **Circle (or group) those 1s**. Larger groups mean simpler terms.

Common group sizes:

* Pairs (2 cells)
* Quads (4 cells)
* Octets (8 cells, if everything is 1)

**4. Extract the Simplified Expression**

For each group:

1. **Determine which inputs are constant** across that group (i.e., which input bits do *not* change).
2. **Form the product term** (AND) with those constant bits, using the original or complemented form (e.g., or ) as required.
3. **Sum** (OR) all the product terms if you have multiple groups.

That final sum of products gives you the simplified Boolean expression for .

For example, if *all* the 1s in the K‐map appear only in columns where , then must be complemented in the expression, leading to .

**Example Outcome**

Often, you’ll discover that depends on fewer variables than the full set . In many seven‐segment or decoder circuits, it might turn out that

meaning that is “1” whenever is “0,” regardless of or .

**That’s it!** By systematically labeling, filling, and grouping your K‐map, you get a straightforward simplification of . Once you see this process in action a few times, it becomes much more intuitive for any output column you might want to map.