Below is a step-by-step look at what the slide is showing and how one would analyze it. The core design problem is:

> **Given a 3-bit input** $(X_2X_1X_0)$, **produce a 3-bit output** $(Y_2Y_1Y_0)$ **which equals "input 2."**

Because we are working in only 3 bits, the outputs for small inputs become negative in 2's-complement form. For example, "0 2" (decimal) is binary = 110 if interpreted as -2 in 3-bit 2's-complement.

---

1. Construct the Truth Table

List out all eight input combinations $(X_2X_1X_0)$ from 0 to 7 in decimal, then compute (input – 2) in 3-bit 2's-complement:

Decimal	$X_2X_1X_0$	**"input** – 2"	$Y_2Y_1Y_0$ (3-bit 2's-complement)
0	000	-2	110
1	001	-1	111
2	010	0	000
3	011	1	001
4	100	2	010
5	101	3	011
6	110	4	100
7	111	5	101

In other words:

- input = 0 → output = 110
- input = 1 → output = 111
- input = 2 → output = 000
- . . . and so on.

---

2. Fill the K-Map(s)

A **Karnaugh map** (K-map) for a 3-bit input is a 2×4 or 4×2 grid in **Gray code** order.
One common layout (columns = X_2X_1, row = X_0) is:

```
          X2X1
            00    01    11    10
X0
0           ?     ?     ?     ?
1           ?     ?     ?     ?
```

- **Columns** go 00 → 01 → 11 → 10 so that each step flips only one bit.
- **Rows** go $X_0 = 0$ then $X_0 = 1$.

Because the output itself is **3 bits** (Y_2, Y_1, Y_0), you actually fill **three** separate K-maps (one per output bit).

(a) K-Map for Y_2

From the truth table:

$$Y_2 = 1 \quad \text{when input decimal} \in \{0, 1, 6, 7\}$$

That corresponds to $(X_2X_1X_0) = 000, 001, 110, 111$.

Placing 1's in those K-map cells and 0 in the rest, you find that Y_2 is 1 in two adjacent columns (00 and 11) for **both** $X_0 = 0$ and $X_0 = 1$. The largest group covers those entire two columns, yielding a simplified expression:

$$Y_2 = (X_2' \wedge X_1') \ \vee \ (X_2 \wedge X_1)$$

which is the "equivalence" function $X_2 \leftrightarrow X_1$ (i.e. $Y_2 = 1$ exactly when X_2 and X_1 match).

(b) K-Map for Y_1

From the table:

$$Y_1 = 1 \quad \text{when input decimal} \in \{0, 1, 4, 5\}.$$

In terms of $(X_2X_1X_0)$, that is $000, 001, 100, 101$. Notice that those all have $X_1 = 0$. Hence:

$$Y_1 = X_1' \quad (\text{i.e. } Y_1 = \neg X_1).$$

(c) K-Map for Y_0

From the table:

$$Y_0 = 1 \quad \text{when input decimal} \in \{1, 3, 5, 7\}.$$

Those are exactly the cases where $X_0 = 1$. Hence:

$$Y_0 = X_0.$$

---

3. Final Simplified Logic

Putting it all together:

$$\begin{aligned} &Y_2 = X_2' X_1' + X_2 X_1 \quad \text{(equivalence of } X_2 \text{ and } X_1\text{)}, \\ &Y_1 = X_1', \\ &Y_0 = X_0. \end{aligned}$$

If you wire up a 3-bit combinational circuit whose outputs follow those three equations, it implements "input − 2" under 3-bit 2's-complement arithmetic.

Takeaways

1. **K-Map Setup**
   - Label rows/columns in Gray code so adjacent cells differ by only one input bit.
2. **Filling Outputs**
   - For each possible input, compute "input−2" in 3-bit 2's-complement and mark the corresponding 1/0 in the map for each output bit.
3. **Grouping and Simplifying**
   - Combine adjacent 1-cells to get minimal SOP (Sum of Products). Here, we see the neat patterns:
     - Y_2 "equivalence" of X_2 and X_1.

- Y_1 is simply $\overline{X_1}$.
- Y_0 is simply X_0.

Hence, the slide is illustrating how to go from the truth table for “input minus 2” to a Karnaugh map, and then derive the simplest Boolean expressions for each output bit.