K-map with don’t care conditions

Lecture-13

By
Bhagyalaxmi Behera
Asst. Professor (Dept. of ECE)

Topics to be discussed

- Boolean function Simplification using K-map with don’t care conditions
- HDL Description

DON'T-CARE CONDITIONS

- It often occurs that for certain input combinations, the value of output is unspecified either because the input combinations are invalid or because the precise value of the output is of no consequence. The combinations for which the values of the expression are not specified are called don't care combinations or optional combinations.

Examples:- In excess-3 code, 0000, 0001, 0010, 1101, 1110 and 1111 are unspecified and never occur. These are called don't cares. The four-bit binary code for the decimal digits has six combinations that are not used and consequently are considered to be unspecified.

- A don’t-care minterm is a combination of variables whose logical value is not specified.

- Such a minterm cannot be marked with a 1 in the map, because it would require that the function always be a 1 for such a combination. Likewise, putting a 0 on the square requires the function to be 0.

- To distinguish the don’t-care condition from 1’s and 0’s, an X is used. Thus, an X inside a square in the map indicates that we don’t care whether the value of 0 or 1 is assigned to *F for the particular minterm.*

- During the process of design, using an SOP map, each don't care is treated as 1 if it is helpful in map reduction, otherwise it is treated as 0 and left alone.
- During the process of design using a POS map, each don't care is treated as 0, if it is useful in map reduction, otherwise it is treated as 1 and left alone.
- In choosing adjacent squares to simplify the function in a map, the don't-care minterms may be assumed to be either 0 or 1.

- When simplifying the function, we can choose to include each don't-care minterm with either the 1's or the 0's, depending on which combination gives the simplest expression.

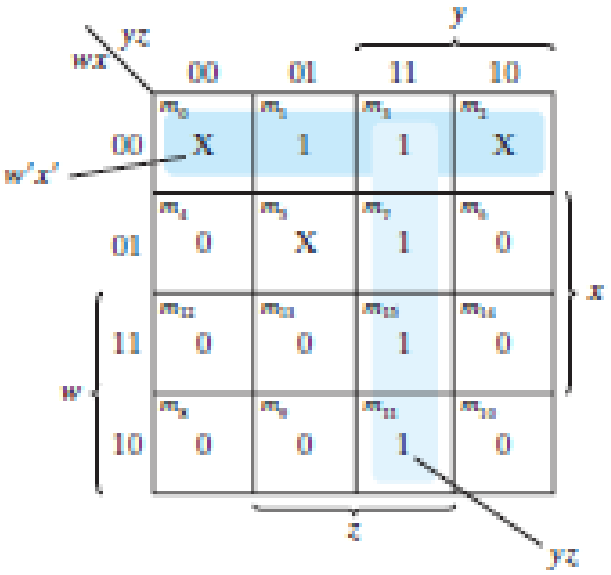
Example 1

Simplify the Boolean function

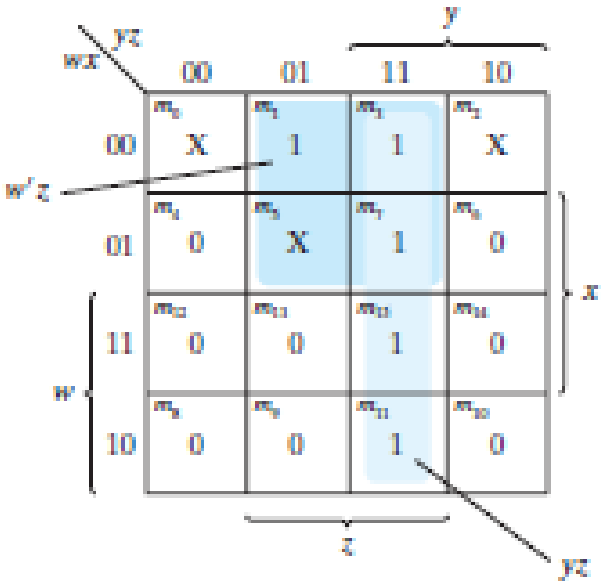
$$F(w, x, y, z) = \Sigma(1, 3, 7, 11, 15)$$

which has the don't-care conditions

$$d(w, x, y, z) = \Sigma(0, 2, 5)$$



(a) $F = yz + w'x'$



(b) $F = yz + w'z$

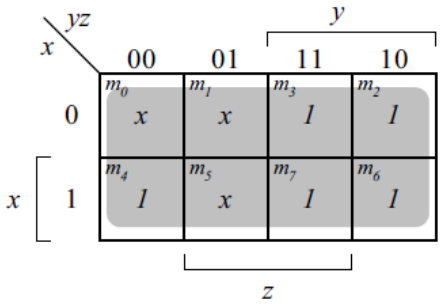
Either one of the preceding two expressions satisfies the conditions stated for this Example.

Example 2

Simplify the Boolean function F, together with don't-care conditions d, and then express the simplified function in sum-of-minterms forms.

$F(x, y, z) = \sum(2, 3, 4, 6, 7)$

$d(x, y, z) = \sum(0, 1, 5)$



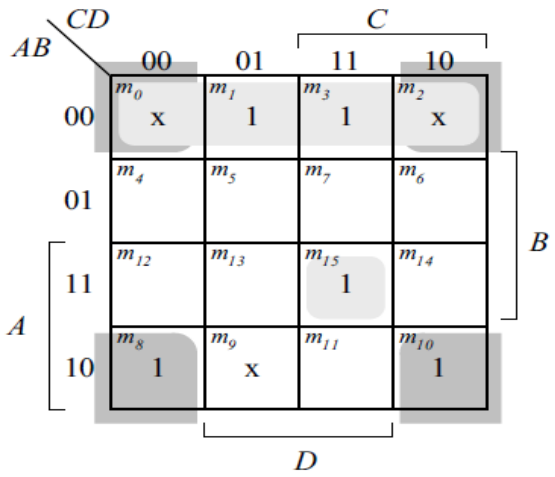
F=1

Example 3

Simplify the Boolean function F, together with don't-care conditions d, and then express the simplified function in sum-of-minterms forms.

$F(A, B, C, D) = \sum(1, 3, 8, 10, 15)$

$d(A, B, C, D) = \sum(0, 2, 9)$



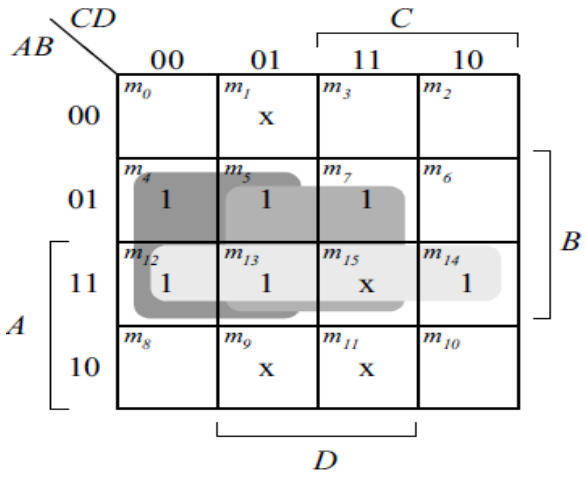
$F = B'D' + A'B' + ABCD$

Example 4

Simplify the Boolean function F, together with don't-care conditions d, and then express the simplified function in sum-of-minterms forms.

$F(A,B,C,D) = \sum(4, 5, 7, 12, 13, 14)$

$d(A,B,C,D) = \sum(1, 9, 11, 15)$



$F = BC' + BD + AB$

HDL Description of The Simplified Function
F = YZ+W'X' (From Example 1)

```
// Verilog model: Circuit with Boolean expressions
module Example_1 (F, W, X, Y, Z);
Output  F;
Input W, X, Y, Z;
assign F = (Y && Z ) || ((! W) && (! X) );
endmodule
```

HDL Description of The Simplified Function
F = (A'+B')(C'+D')(B'+D)

```
// Verilog model:with Boolean expressions
module Circuit (F, A,B,C,D);
Output  F;
Input  A, B, C, D;
assign F = ((! A) || (! B) ) && ((! C) || (! D) ) && ((! B) || D );
endmodule
```

HDL Description of The Simplified Function
(*From Example 3)*

```
module Example_3 (F, A, B, C, D);
Output  F;
Input A, B, C, D;
assign F = ((!B) && (!D)) || ((! A) && (! B) ) ||(A && B && C && D);
endmodule
```