Lecture 12 In-Class Worksheet

1 Learning Objectives

This worksheet is based on Lumetta course notes section 2.2. After completing this lesson, you will know how to:

- [S12-1] Manipulate Boolean expressions using Boolean identities.
- [S12-2] Find the minimal SOP and POS forms of a function with *don't care* values.

2 Boolean Identities

The following box lists several important identities of Boolean algebra.

Boolean Identities

$$1 + A = 1 \qquad 0 \cdot A = 0 \tag{1}$$
$$0 + A = A \qquad 1 \cdot A = A \tag{2}$$
$$A + A = A \qquad A \cdot A = A \tag{3}$$
$$A + \bar{A} = 1 \qquad A \cdot \bar{A} = 0 \tag{4}$$
$$A \cdot (B + C) = (A \cdot B) + (A \cdot C) \qquad A + (B \cdot C) = (A + B) \cdot (A + C) \tag{5}$$
$$\overline{A + B} = \bar{A} \cdot \bar{B} \qquad \overline{A \cdot B} = \bar{A} + \bar{B} \tag{6}$$
$$AB + \bar{A}C + BC = AB + \bar{A}C \qquad (A + B)(\bar{A} + C)(B + C) = (A + B)(\bar{A} + C) \tag{7}$$

Identity (7), called Consensus, can be derived from Identities (1), (2), (4), and (5) as shown below (however, it is useful to think of it as another identity):

$$\begin{aligned} AB + \bar{A}C + BC &= AB + \bar{A}C + (A + \bar{A})BC \\ &= AB + \bar{A}C + ABC + \bar{A}BC \\ &= AB(1 + C) + \bar{A}C(1 + B) \\ &= AB + \bar{A}C. \end{aligned}$$

We can use the identities above to transform Boolean expressions into a particular form. In the previous worksheet, we already saw how we can use DeMorgan's Laws (6) to transform a Boolean expression into one where negation is only applied to inputs, that is, an expression consisting of AND, OR, and literals only. We can also use these identities to simplify Boolean expressions.

Q1. Simplify the following Boolean expression when $A = 1$:

$$AB + \bar{A}(B + C) + (A + C)(\bar{A} + \bar{B})$$

Q2. Simplify the Boolean expression:

$$AB + \bar{A}(B + C) + (A + C)(\bar{A} + \bar{B})$$

(This is the same Boolean expression as Q1, but we are not assuming $A = 1$.)

3 Don't Care Values

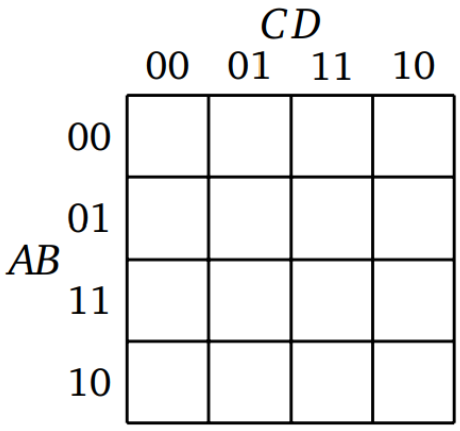
When design requirements allow some of the outputs of a Boolean function to be either 0 or 1, such a value is called a *don't care* and is denoted with an X in the truth table and Karnaugh map. The resulting Karnaugh map rules for SOP optimization are:

- **1.** Each loop must be rectangular.
- **2.** Each loop must enclose exactly 2^k values that are either *one* or *don't care*.
- **3a.** In a three-variable K-map, loops may wrap around left to right.
- **3b.** In a four-variable K-map, loops may wrap around top to bottom and left to right.
- **4.** Each loop must be as large as possible (subject to the above).
- **5.** Each *one* must be inside some loop.

For POS optimization, replace *one* with *zero* in the rules above.

Q3. Express the function U in minimal SOP form.

A	B	C	D	U
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	X
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	X



Q4. Express the function U in minimal POS forms.

A	B	C	D	U
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	X
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	X