Example- A Boolean function is defined as-

$$F_2 = A'C' + A'B'C + A'BC + BC'$$

- F_2 has 4 terms and 10 literals
- First create K-Map directly from function.

a) For term $A'C'$, we fill-in each square that has $A = 0$ (i.e., A') AND $C = 0$ (i.e., C') (should be two squares),

A\BC	**00**	**01**	**11**	**10**
0	**1**	**-**	**-**	**1**
1	**-**	**-**	**-**	**-**

b) for term $A'B'C$, we fill-in the square that has $A = 0$ (i.e., A') AND $B = 0$ (i.e., B') AND $C = 1$ (i.e., C),

A\BC	**00**	**01**	**11**	**10**
0	**1**	**1**		**1**
1				

c) for term $A'BC$, we fill-in the square that has $A = 0$ (i.e., A') AND $B = 1$ (i.e., B) AND $C = 1$ (i.e., C), and

A\BC	**00**	**01**	**11**	**10**
0	**1**	**1**	**1**	**1**
1	**-**	**-**	**-**	**-**

d) for term BC', we also fill-in each square that has $B = 1$ (i.e., B) AND $C = 0$ (i.e., C') (should be two squares),

A\BC	**00**	**01**	**11**	**10**
0	**1**	**1**	**1**	**1**
1	**-**	**-**	**-**	**1**

- Next, use this K-Map to directly express F_2 in the canonical sum-of-minterms form. Put the applicable minterm into each square where there is a "1"-

A\BC	**00**	**01**	**11**	**10**
0	$m_0 = A'B'C'$	$m_1 = A'B'C$	$m_3 = A'BC$	$m_2 = A'BC'$
1	**-**	**-**	**-**	$m_6 = ABC'$

Therefore,

$$F_2 = m_0 + m_1 + m_2 + m_3 + m_6 = \sum(0,1,2,3,6)$$
$$\underline{= A'B'C' + A'B'C + A'BC + A'BC' + ABC'}$$

- Then, use this K-Map to directly express F_2 in simplified sum-of-products form.

A\BC	**00**	**01**	**11**	**10**
0	1	1	1	1
1	**-**	**-**	**-**	1

a) The first row has <u>four</u> adjacent squares which correspond to the single literal $A = 0$ (i.e., A').

b) The rightmost column has <u>two</u> adjacent squares which corresponds to the two literals $BC = 10$ (i.e., BC').

c) Therefore, the simplified sum-of-products form is-

$$\underline{F_2 = A' + BC'}$$

- Check these results using Truth Tables to show they agree.

$$F_2 = A'C' + A'B'C + A'BC + BC'$$

A	B	C	A'	B'	C'	$A'C'$	$A'B'C$	$A'BC$	BC'	F_2
0	0	0	1	1	1	1	0	0	0	**1**
0	0	1	1	1	0	0	1	0	0	**1**
0	1	0	1	0	1	1	0	0	1	**1**
0	1	1	1	0	0	0	0	1	0	**1**
1	0	0	0	1	1	0	0	0	0	**0**
1	0	1	0	1	0	0	0	0	0	**0**
1	1	0	0	0	1	0	0	0	1	**1**
1	1	1	0	0	0	0	0	0	0	**0**

$$F_2 = A'B'C' + A'B'C + A'BC + A'BC' + ABC'$$

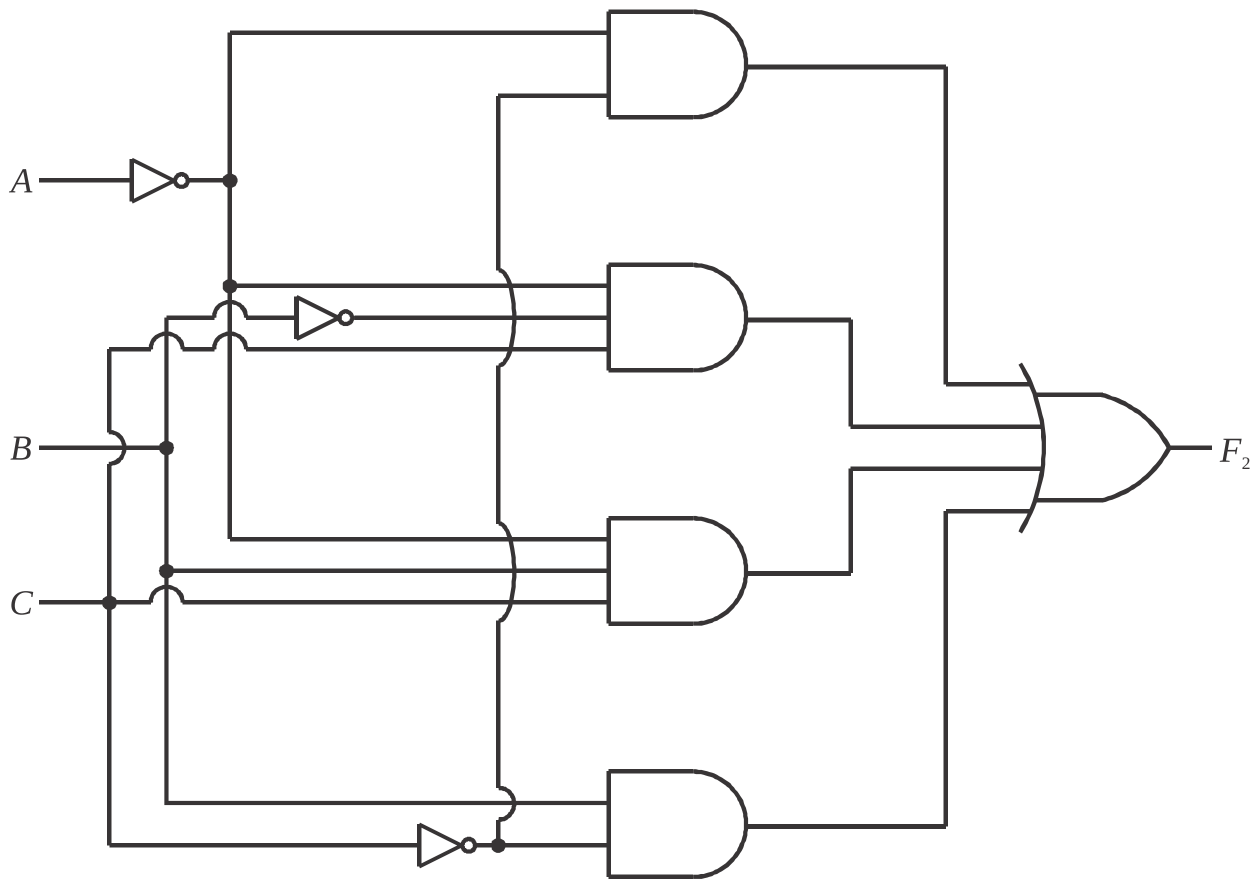
A	B	C	A'	B'	C'	$A'B'C'$	$A'B'C$	$A'BC$	$A'BC'$	ABC'	F_2
0	0	0	1	1	1	1	0	0	0	0	**1**
0	0	1	1	1	0	0	1	0	0	0	**1**
0	1	0	1	0	1	0	0	0	1	0	**1**
0	1	1	1	0	0	0	0	1	0	0	**1**
1	0	0	0	1	1	0	0	0	0	0	**0**
1	0	1	0	1	0	0	0	0	0	0	**0**
1	1	0	0	0	1	0	0	0	0	1	**1**
1	1	1	0	0	0	0	0	0	0	0	**0**

$$F_2 = A' + BC'$$

A	B	C	C'	A'	BC'	F_2
0	0	0	1	1	0	**1**
0	0	1	0	1	0	**1**
0	1	0	1	1	1	**1**
0	1	1	0	1	0	**1**
1	0	0	1	0	0	**0**
1	0	1	0	0	0	**0**
1	1	0	1	0	1	**1**
1	1	1	0	0	0	**0**

- The original F_2 has a logic-circuit diagram with 8 logic gates (three NOT, two 2-input AND, two 3-input AND, & a 4-input OR) and three inputs-

$$F_2 = A'C' + A'B'C + A'BC + BC'$$



- Note that the simplified F_2 has much simpler logic-circuit diagram with four simpler logic gates (2 NOT, a 2-input AND, & a 2-input OR) and 3 inputs.

$$F_2 = A' + BC'$$