EE 2720

Handout #16

EE 2720

• Karnaugh maps continued.

Some definitions follow first:

• Definition: Any single 1 or any group of 1's which can be combined together on a Karnaugh map of a function F represents a product term which is called an implicant of F.

• Definition: A product term implicant is called a prime implicant if it cannot be combined with another term to eliminate a variable. All of the prime implicants of a function F can be easily obtained from a Karnaugh map as follows: A single 1 on a Karnaugh map represents a prime implicant if it is not adjacent to any other 1s. Two adjacent 1s on a Karnaugh map combined form a prime implicant if they are not contained in a group of four 1s. four adjacent 1s combined form a prime implicant if they are not contained in a group of eight 1s, etc.

An example that clarifies the above follows:

Example 1: Consider the logic function F shown by the Karnaugh map of figure 1 below:

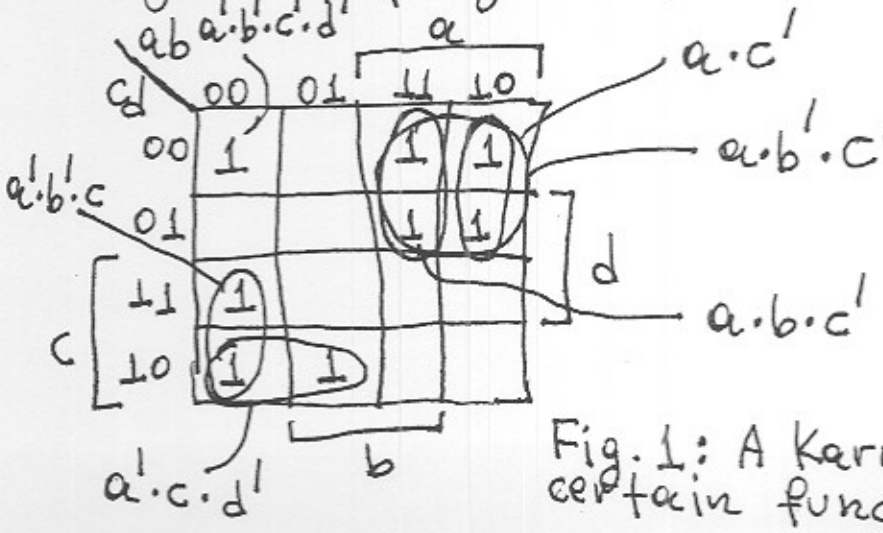


Fig. 1: A Karnaugh map for a certain function F.

Fig. 1 of previous page shows some implicants and some prime implicants. In fig. 1, the product terms $a' \cdot b' \cdot c$, $a' \cdot c \cdot d'$ and $a \cdot c'$ are prime implicants because they cannot be combined with other terms to eliminate a variable. On the other hand, $a' \cdot b' \cdot c' \cdot d'$ is not a prime implicant because it can be combined with $a' \cdot b' \cdot c \cdot d'$ or $a \cdot b' \cdot c' \cdot d'$. Neither $a \cdot b \cdot c'$ nor $a \cdot b' \cdot c'$ is a prime implicant because these terms can be combined together to form $a \cdot c'$.

An important theorem follows:

• Prime Implicant Theorem: A minimal sum is a sum of prime implicants; (see page 15 of handout #15 for the definition of minimal sum). In other words, in order to find a minimal sum, we need not consider any product terms that are not prime implicants.

Note: This theorem can be proved by contradiction. The proof can be found in pages 226–227 of the text. Read it if you want. You are not responsible for it.

Note: The sum of all prime implicants of a logic function is called the complete sum. Although the complete sum is always a legitimate way to realize a logic function, it is not always minimal; (it is not always going to give us a minimal sum). In other words, the minimal sum consists of some (but not necessarily all) of the prime implicants of a function.

Example 2: Consider the logic function $F = \sum_{w,x,y,z}(1,3,4,5,9,11,12,13,14,15)$. The karnaugh map for this function is shown in figure 2 on the next page.

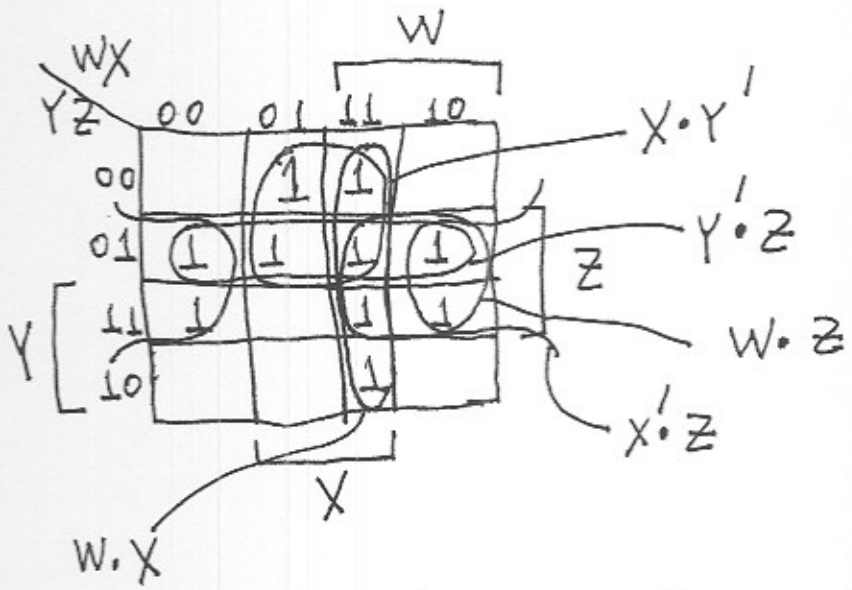


Figure 2: Karnaugh map for the logic function $F=\Sigma_{W,X,Y,Z}(1,3,4,5,9,11,12,13,14,15)$.

Here, the above function F has five prime implicants They are: $X\cdot Y'$, $Y'\cdot Z$, $W\cdot Z$, $X'\cdot Z$ and $W\cdot X$. However, the minimal sum includes only three of them which are $X\cdot Y'$, $X'\cdot Z$ and $W\cdot X$. So minimal sum for F is:

$$F=X\cdot Y'+X'\cdot Z+W\cdot X \quad (1)$$

Question: How can we systematically determine which prime implicants to include and which to leave out?

Answer: We need one more definition here, before we answer the above question.

- Definition: An essential prime implicant of a logic function is a prime implicant where all its minterms (contained in this prime implicant) are covered by only one prime implicant.
- Important note: Every essential prime implicant mus

be included in the minimal sum for the logic function. So, just include all the essential prime implicants in the minimal sum; (only the essential ones).

An example that clarifies the above follows:

Example 3: Consider the following logic function shown in the Karnaugh map of figure 3 below:

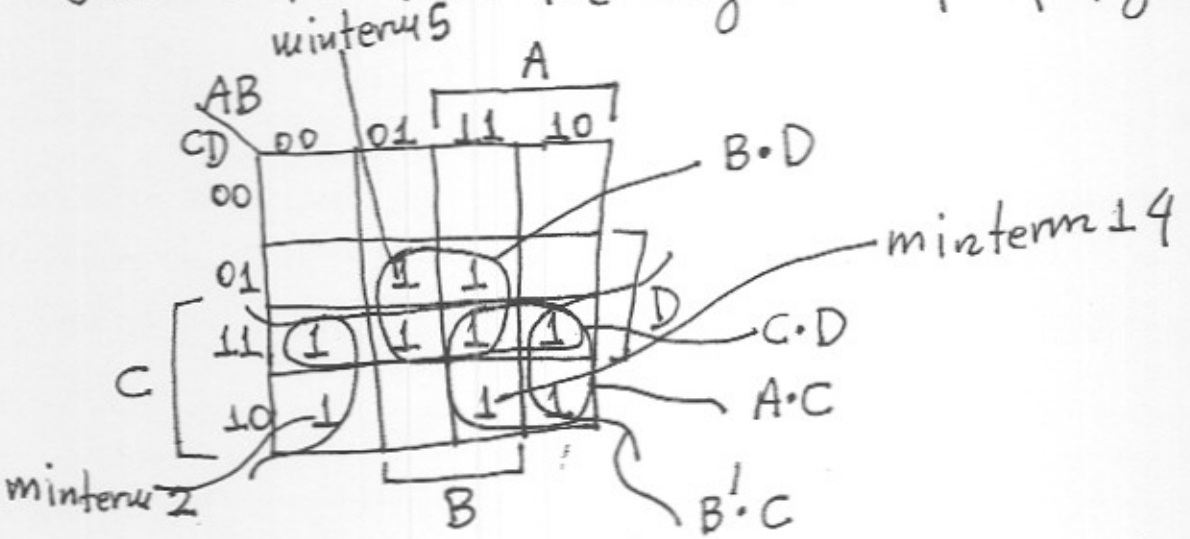


Figure 3: Karnaugh map for a certain logic function F.

In the above example, $B'\cdot C$ is an essential prime implicant because minterm 2 is not covered by any other prime implicant except $B'\cdot C$. However, $C\cdot D$ is not essential prime implicant because each of the minterms in $C\cdot D$ can be covered by another prime implicant. $B\cdot D$ is an essential prime implicant because minterm 5 is not covered by another prime implicant except $B\cdot D$. Likewise, $A\cdot C$ is an essential prime implicant because minterm 14 is not covered by any other prime implicant except $A\cdot C$. Therefore, the essential prime implicants for the above logic function F are $B'\cdot C$, $B\cdot D$ and $A\cdot C$ and the minimal sum for F is

$$F = B'\cdot C + B\cdot D + A\cdot C \qquad (2)$$

As seen, the nonessential prime implicant C·D is not needed and is not included in the minimal sum of eq.(2) of the previous page.

• Simplifying Products of Sums:

Once we know how to simplify sums of products, (that is what we just did), it is very easy to simplify products of sums. (using a karnaugh map). If we are given a logic function F and we are asked to provide a simplified product-of-sums expression for F, we can compute F', simplify F' by combining its 1's using a karnaugh map, provide a simplified sum-of-products expression for F' (mean; the 1's of F will become 0's of F' and the 0's of F will become 1's of F' and then compute F=(F')'. This will give us the simplified product-of-sums expression for F. The above can be done in one step by combining the 0's of F (instead of combining its 1's). This will be explained later. An example follows that explains the above issue:

Example 4: Find a simplified product-of-sums expression for the logic function $F=\Sigma_{x,y,z}(1,3,5,6,7)$.

Answer: Figure 4 below shows the karnaugh map for F of this example:

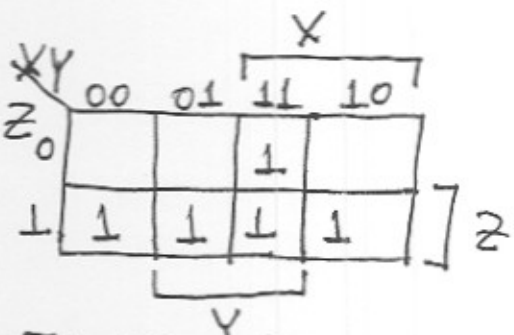


Fig 4: karnaugh map for $F=\Sigma_{x,y,z}(1,3,5,6,7)$.

Figure 5 on next page shows a karnaugh map for F'.

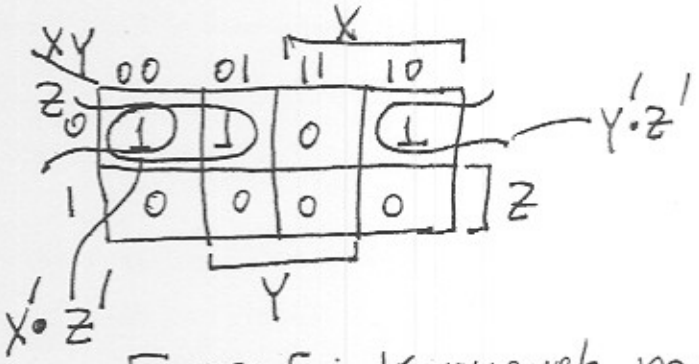


Figure 5: Karnaugh map for F'.

From the above karnaugh map of fig. 5, one can get the following simplified sum-of-products expression for F':

$$F' = X' \cdot Z' + Y' \cdot Z' \quad (3).$$

Complementing both sides of eq. (3) above we get:

$$(F')' = F = (X' \cdot Z' + Y' \cdot Z')' = (X+Z) \cdot (Y+Z) \quad (4).$$

The above eq. (4) is the simplified product-of-sums expression for F.

As we said earlier, the above can be done in one step by combining the 0's of F instead of combining its 1's. The rules are the following:

- 2^i cells containing zeroes can be combined to form a sum term containing $n-i$ literals, where n is the number of variables in the function.
- A set of 2^i cells containing zeroes may be combined if there are i variables of the logic function that take on all 2^i possible combinations within that set, while the remaining $n-i$ variables have the same value throughout that set. The corresponding sum term has $n-i$ literals, where a variable is complemented if it appears as 1 in all of the cells, and uncomplemented if it appears as 0.

Let's now present the previous example (example 4.1 again) by just combining the zeroes of F (doing it in one step). A karnaugh map that shows the zeroes of F is shown in fig. 6 on next page.

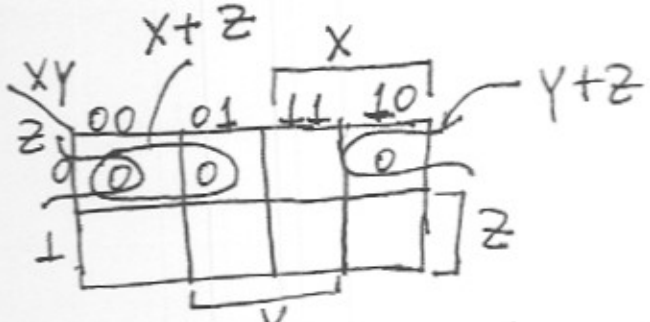


Figure 6: Karnaugh map for F of example 4. Here we shade the zeroes of F and we combine zeroes (not ones)

Some explanations follow; (this is the last time I provide detailed explanations).

- We combine cells 0 and 2. Within that set of cells, variable Y takes on all possible values (0 and 1 I mean), while variables X and Z have the same value throughout the set of cells 0 and 2. The resulting sum term as a result of combining cells 0 and 2 will have 2 literals. The participating variables in the sum term will be X and Z. Both variables X and Z will be uncomplemented because they both appear as 0's in both cells 0 and 2. Therefore, the resulting sum term, (result of combining cells 0 and 2 I mean) will be X+Z.
- We combine cells 0 and 4. Within that set of cells, variable X takes on all possible values (0 and 1 I mean), while variables Y and Z have the same value throughout the set of cells 0 and 4. The resulting sum term as a result of combining cells 0 and 4 will have 2 literals. The participating variables in the sum term will be Y and Z. Both variables Y and Z will be uncomplemented because they both appear as 0's in both cells 0 and 4. Therefore, the resulting sum term, (result of combining cells 0 and 4 I mean) will be Y+Z.
- We now covered (took into consideration) all 0's in the Karnaugh map for the function F, so we are ready to provide a simplified product-of-sums expression for F. We have:

F = (result of combining cells 0 and 2) $\overset{\text{AND}}{\downarrow}\cdot$ (result of combining cells 0 and 4) $= (X+Z)\cdot(Y+Z)$ (5).

Note: From eq. (5) above, one can easily get the minimized OR-AND logic circuit for F which is shown in figure 7 below

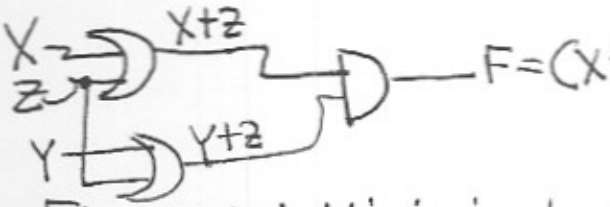


Figure 7: Minimized OR-AND logic circuit for F of example 4.

Note: You can easily apply the graphical approach (bubbles etc) to transform the circuit of fig. 7 into an equivalent circuit that relies only on NOR gates. Do it if you want. We have done it many times

One more example of providing simplified product-of-sums expression for a logic function follows. Again, here, we do the simplification of F in one step by combining the 0s of F.

Example 5: Find a simplified product-of-sums expression for the logic function

$F = \Sigma_{W,X,Y,Z}(1,4,5,6,7,9,14,15)$.

Answer: A karnaugh map that shows the 0s of F is shown below:

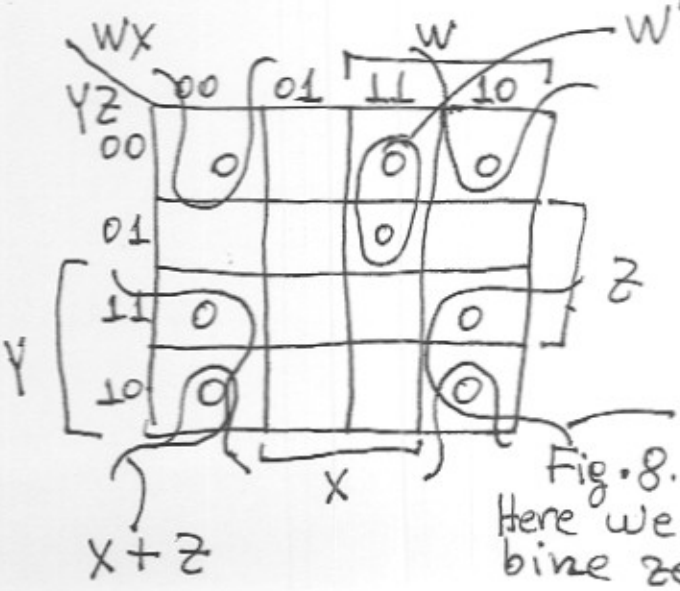


Fig. 8. karnaugh map for F of example 5. Here we show the zeroes of F and we combine zeroes.

From the karnaugh map of fig. 8 of the previous page, we can easily get the following simplified product-of-sums expression for F shown in eq. (6) below:

$$F = (X+Z)\cdot(X+Y')\cdot(W'+X'+Y) \quad (6)$$

Note: From eq. (6) above, one can very easily get a minimized OR-AND logic circuit for F. Do it if you want. We have done it many times.

A definition follows:

Definition: A minimal product of a logic function F is a product-of-sums expression for F that has the fewest possible sum terms (number of first-level gates and second-level gate inputs) and, within that constraint the fewest possible literals (number of first-level gate inputs).

- Don't-Care Input Combinations (or simply don't cares).

Sometimes the specification of a combinational circuit is such that its output doesn't matter for certain input combinations, called don't cares. This can be true when the outputs don't matter when these input combinations occur, or because these input combinations never occur. For example, suppose we want to build a prime number detector whose 4-bit input $A = (a_3 a_2 a_1 a_0)_2$ is a BCD digit; (recall that BCD digits are 0, 1, ..., 9 or in binary 0000, 0001, ..., 1001). The output of this detector F will be 1 if the input BCD digit is a prime number. Since the only prime BCD digits are 1, 2, 3, 5, 7, F can be written as

$$F = \Sigma_{a_3,a_2,a_1,a_0}(1,2,3,5,7) + d(10,11,12,13,14,15)$$

CT.

In eq. G) of the previous page, the d(...) list specifies the don't-care input combinations for the function, also known as the d-set. Function F must be 1 for input combinations 1, 2, 3, 5, 7 (in binary 0001, 0010, 0011, 0101, 0111), F can have any values for inputs in the d-set (10, 11, 12, 13, 14, 15); (recall that input combinations 10, 11, 12, 13, 14, 15 will never occur because they are not BCD digits so we don't care what the value of the output is for these input combinations), and F must be 0 for all other input combinations

- When we use a karnaugh map to simplify a function that contains don't cares, the don't cares ~~are denoted by~~ (don't-care input combinations I mean) are denoted by d's on the karnaugh map; (some books use X's).
- If you want to find a simplified sum-of-products expression for a function F that contains don't-care input combinations using a karnaugh map, combine cells containing 1's together with cells containing d's if this is going to be to your benefit. And what I mean here, is if it is going to give you product terms with less number of literals; (remember that the more cells you combine, the less literals your product terms will have which implies a cheaper circuit realization or circuit with ~~gates~~ first-level gates with fewer inputs). We can combine 1's and d's because d's are don't cares for these input combs (and)

nations our function can take any value, so 1 is OK; (that is what we actually do). Do not combine cells containing only d's. This will give you unnecessary product terms which means extra first-level gates (AND gates in this case) that you do not need.

- If you want to find a simplified product-of-sums expression for a function F that contains don't-care input combinations using a Karnaugh map, combine cells containing 0s together with cells containing d's if this is going to be to your benefit. Again, what I mean here is if this is going to give you sum terms with less number of literals. We can combine 0's and d's because d's are don't cares and for these input combinations our function can take any value, so 0 is OK; (that is what we actually do). Again, do not combine cells containing only d's because this will give you unnecessary terms which means extra gates that you do not need.

Example 6: Using a Karnaugh map, find a simplified sum-of-products expression for the prime BCD-digit detector described earlier in this handout.

Answer: As we already said, the output of this detector is:

$$F = \Sigma_{a_3, a_2, a_1, a_0}(1, 2, 3, 5, 7) + d(10, 11, 12, 13, 14, 15) \quad (8)$$

A Karnaugh map that shows the 1's and the d's of F is shown in figure 9 on the next page.

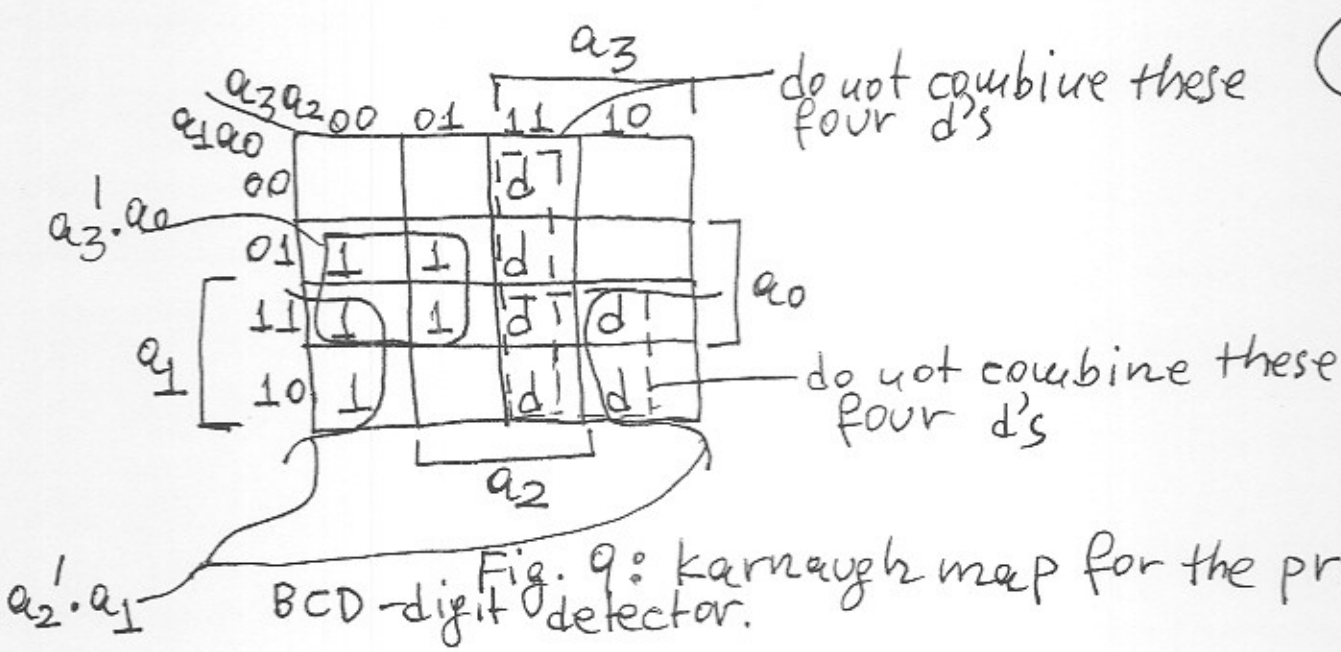


Fig. 9: karnaugh map for the prime BCD-digit detector.

From the karnaugh map of fig. 9 above, we can easily get the following simplified sum-of-products expression for F shown in eq. (9) below:

$$F = a_3'.a_0 + a_2'.a_1 \quad (9).$$

From the above eq. (9) we can very easily get the minimized AND-OR logic circuit for the BCD-digit detector shown in figure 10 below: (Prime)

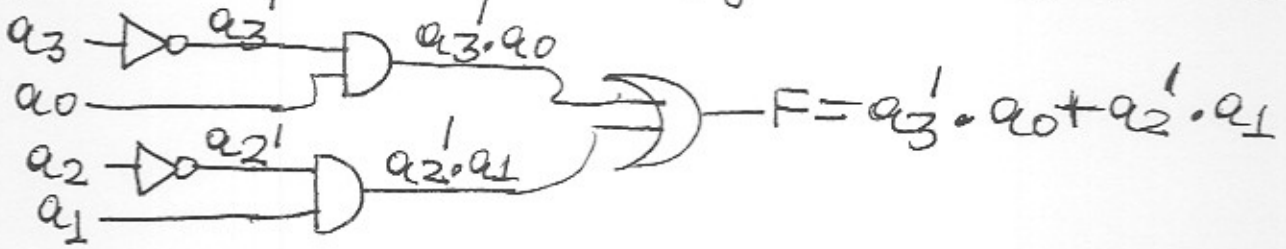


Figure 10: Minimized AND-OR logic circuit for the BCD-digit prime detector.

Note: We can easily transform the circuit of fig. 10 into one that relies only on NAND gates by using the graphical approach (bubbles etc). Do it if you want. We have done such things many times.

Problem: Using a Karnaugh map, provide a simplified product-of-sums expression for the prime BCD-digit detector of the previous example.

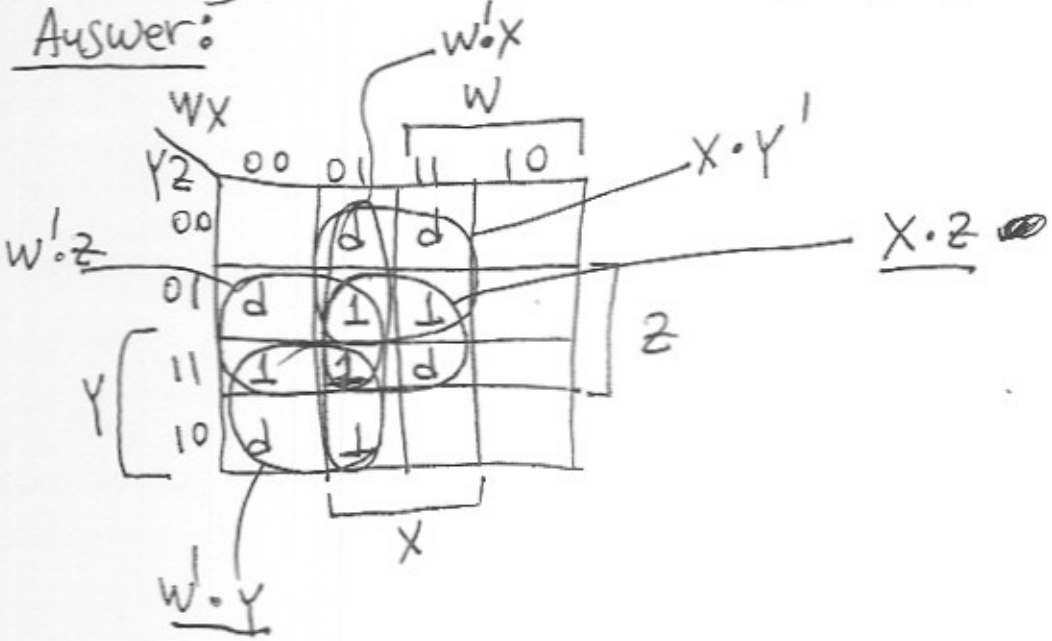
Answer: Do it as a homework problem if you want. I might assign it as a homework problem later. You know how to do this. I already told you. Look on page 11 of this handout etc.

Two examples follow. I will not do them in class. You can read them at home. You know how to do them.

Example 7: Using a Karnaugh map, provide a simplified sum-of-products expression for the logic function

$$F = \Sigma_{w,x,y,z}(3,5,6,7,13) + d(1,2,4,12,15)$$

Answer:



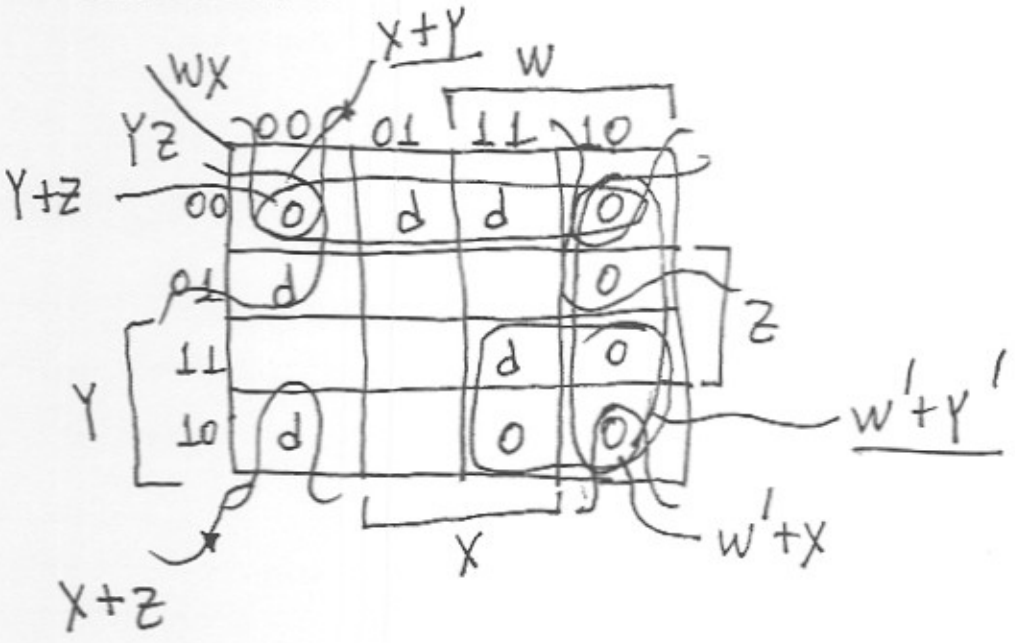
From the above Karnaugh map we get:

$$F = w' \cdot z + w' \cdot y + x \cdot z + x \cdot y' + w \cdot x$$

Note: The above expression of F is not the minimal sum. Can you find the minimal sum? Look at the essential prime implicants on pages 3 and 4 of this handout. What you should get is $F = w' \cdot y + x \cdot z$.

Example 8: Using a Karnaugh map, provide a simplified product-of-sums expression for the logic function of example 7 of the previous page.

Answer:



From the above Karnaugh map we get:

$$F=(Y+Z)\cdot(X+Z)\cdot(W'+X)\cdot(W'+Y')\cdot(X+Y)$$

Note: The above expression for F that we got is not the minimal product - So how do we get the minimal product? Some definitions must follow about implicants, prime implicants and essential prime implicants when combining 0's on a Karnaugh map; (maxterms this time; not minterms).

- Definition: When combining 0's on a Karnaugh map, any single 0 or any group of 0's which can be combined together on a Karnaugh map of a function represents a sum term which is called an implicant of F.
- Definition: A sum term implicant is called a prime implicant if it cannot be combined with another

term to eliminate a variable. All of the prime implicants of a function F can be easily obtained from a karnaugh map as follows: A single 0 on a karnaugh map represents a prime implicant if it is not adjacent to any other 0's. Two adjacent 0s on a karnaugh map combined form a prime implicant if they are not contained in a group of four 0's. four adjacent 0's combined form a prime implicant if they are not contained in a group of eight 0's etc.

- Definition: A sum term essential prime implicant of a logic function is a prime implicant where all its maxterms (contained in this prime implicant) are covered by only one prime implicant.
- Important note: Every essential prime implicant must be included in the minimal product for the logic function. So, just include all the essential prime implicants in the minimal product; (only the essential ones).

Note: Now you can provide the minimal product for the function F of example 8 on page 14. Do it if you want. What you should get is $F=(X+Y)\cdot(w'+y')$. It is easy by now!

We now present the last topic on karnaugh maps. This is the topic on multiple-output minimization presented below. I will try to make the topic simple although it might not always be that simple. Consider for example the example presented on the next page.

An introduction follows before the example:

Most combinational logic circuits require more than one output. We can handle a circuit with n outputs as n independent single-output design circuit-problems. However, in doing so, we might miss some opportunities for optimization; (circuit minimization I mean).
The example below clarifies the above issue:

Example 9: Find simplified AND-OR logic circuits for the functions F and G where F and G are:

$$F = \Sigma_{x,y,z}(3,6,7)$$

$$G = \Sigma_{x,y,z}(0,1,3)$$

Answer: I will first provide an answer treating the design as two independent single-output designs. The Karnaugh maps for the functions F and G are shown in figures 11 and 12 below:

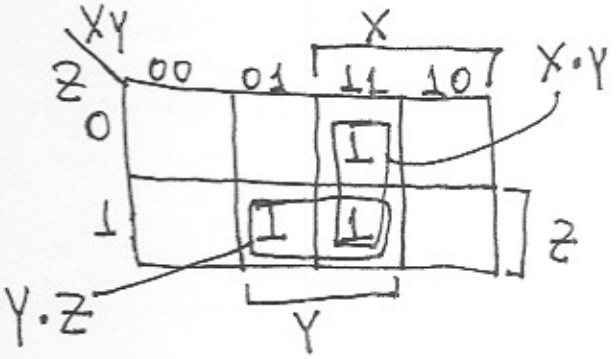


Figure 11: karnaugh map for F of example 9.

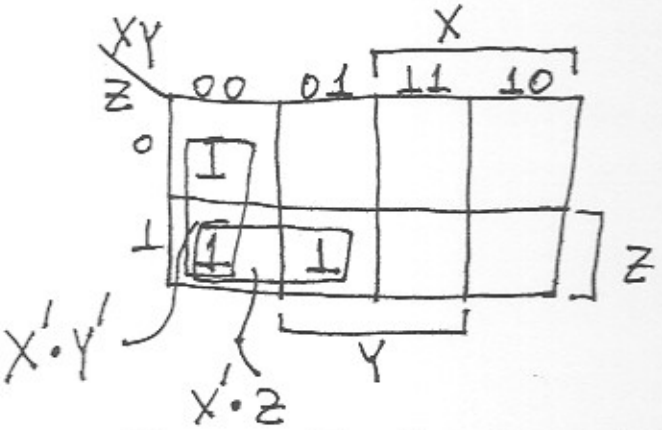


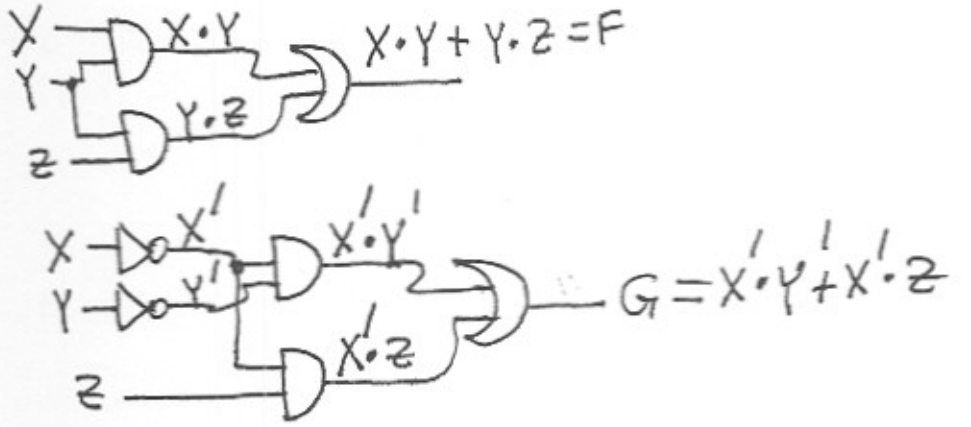
Figure 12: Karnaugh map for G of example 9.

The above Karnaugh maps give the following simplified expressions for F and G

$$F = X \cdot Y + Y \cdot Z \quad (10).$$

$$G = X' \cdot Y' + X' \cdot Z \quad (11).$$

From equations (10) and (11) of the previous page we can easily get the following AND-OR logic circuit realizations for F and G:



However, we can also find a pair of sum-of-products expressions that share a product term, such as the resulting logic circuit has one fewer gate than the original design shown in the figure above. The Karnaugh maps of figures 13 and 14 and the circuit diagram of figure 15 show the situation.

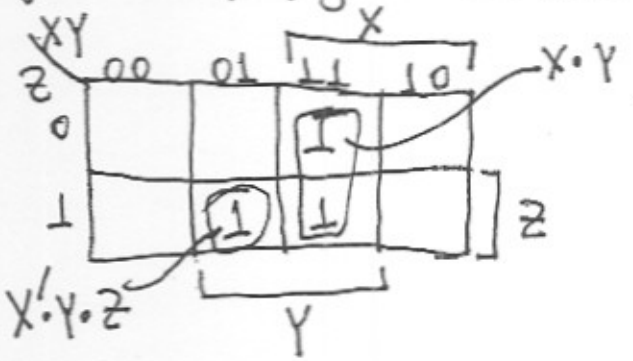


Figure 13: Karnaugh map for F of example 9. Here $F = X \cdot Y + X' \cdot Y \cdot Z$

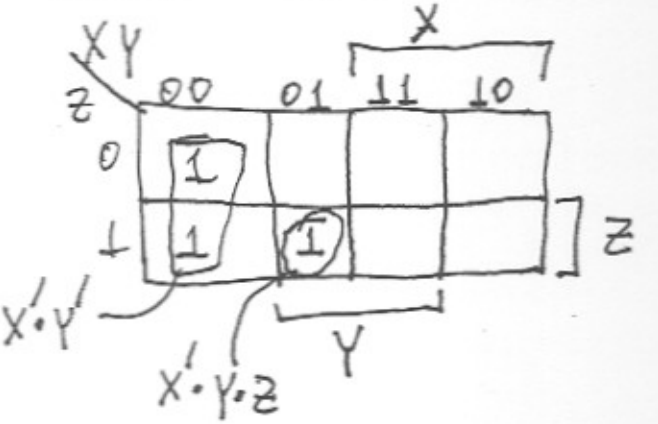


Figure 14: Karnaugh map for G of example 9. Here $G = X' \cdot Y' + X' \cdot Y' \cdot Z$.

Figure 15 (the circuit diagram for F and G) is shown on the next page.

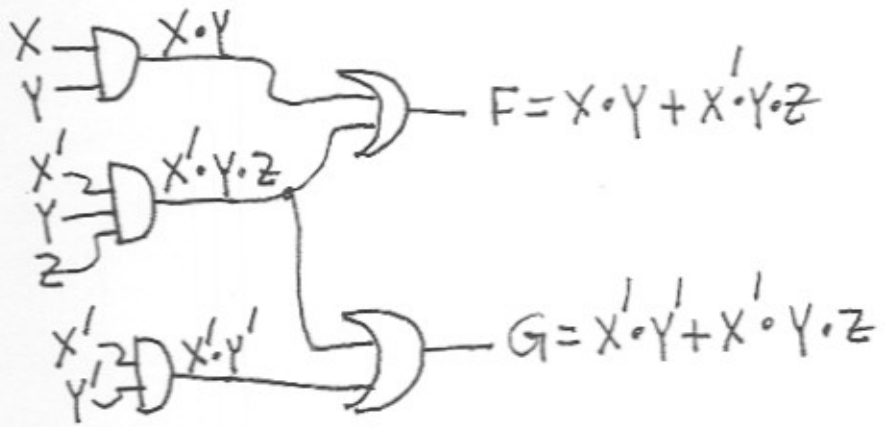


Figure 15: Logic circuit for F and G. Here we are treating F and G together.

Comparing the original logic circuit (top of page 17) with this of figure 15, we see that in figure 15 we save one AND gate by treating F and G together.

Note: This is the end of the subject of Karnaugh maps. We dedicated 3 handouts and almost 49 pages but the subject is very important. Probably the most important in the entire course. I hope you enjoy the subject.