2. Represent the fuzzy logic function in a canonical form.

This is done using the identities and transforms of fuzzy logic.

Example 4. When the function is not in sum-of-products form it is transformed to sum-of-products form: $F = X_1 (X_1 X_2 X'_2)' (X_1 X_2 X_3)' = X_1 (X'_1 + X'_2 + X_2) (X'_1 + X'_2 + X'_3) = X_1 X'_1 + X_1 X'_1 X'_2 + X_1 X'_1 X'_3 + X_1 X'_1 X'_2 + X_1 X'_2 + X_1 X'_2 X'_3 + X_1 X'_1 X'_2 + X_1 X_2 X'_2 + X_1 X_2 X'_3 = X_1 X'_1 + X_1 X'_1 X'_2 + X_1 X'_1 X'_3 + X_1 X'_2 + X_1 X'_2 X'_3 + X_1 X_2 X'_2 + X_1 X_2 X'_3$.

Eliminate the terms which can be subsumed by other terms. After changing the fuzzy logic function into the sum-of-products form, the function is not reduced to its simplest form. The function now needs to be reduced to its simplest form to be canonical. This is done using the subsume Rule, as shown in Fig. 7.

Example 5: The function $F(x_1, x_2) = x_2 x'_2 + x'_1 x_2 + x_1 x'_2 + x_1 x'_1 x'_2$ is equivalent to $F(x_1, x_2) = x'_1 x_2 + x_1 x'_2$ as shown in Fig. 9.

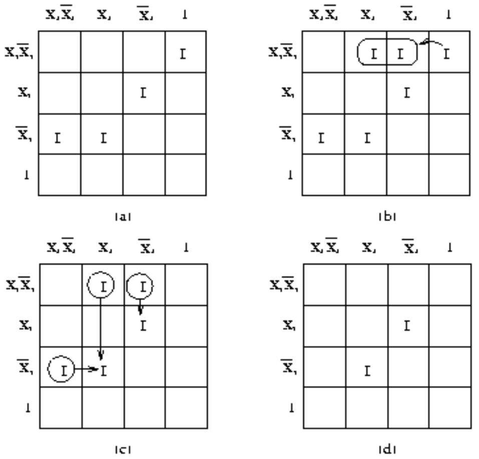


Figure 9. Using fuzzy maps to find the canonical form of f(x₁, x₂)

Problems with using Kandel's and Francioni's method of Fuzzy logic decomposition.

The decomposition method, proposed by Kandel and Francioni [3], uses fuzzy maps to determine if a fuzzy decomposition exists and then uses S-maps for the fuzzy decomposition. Theorems, definitions, and a table are used to tell if the function in a fuzzy map or a S-map is decomposable and how to perform the decomposition. Special patterns must be recognized in the maps. Their method was not implemented as a computer program, nor its correctness was verified, and it is difficult to use both fuzzy maps and fuzzy S-maps concurrently. Conceptually and didactically the many theorems from [3] and definitions are complex and hard to understand because they are not linked in any way to the well-known concepts of Ashenhurst/Curtis decomposition or other general decompositions. Besides, the method is difficult to extend to an arbitrary number of variables and to generalized Curtis-like decompositions. In the sequel we will explain the steps of converting any fuzzy function to a functional form that can be decomposed using multiple-valued decomposition approaches developed recently [12,13,14]. The next sections will show how a fuzzy function in a sum-of-products form can be converted to a multiple-valued function, decomposed, and next converted back into a multiple-valued fuzzy function.

4. FUZZY FUNCTION TO MULTIPLE-VALUED FUNCTION CONVERSION

The procedure to convert a fuzzy function to the multiple-valued (MV) function is the following:

1. A fuzzy logic function needs to be in a sum-of-products form.
2. The new map for the MV function needs to be of dimension equal to the number of variables in the fuzzy function. Every variable X_i in the map will have 3 values. The value $X_i = 0$ is used in the map where variable's complement is present in the term. The value of 1 is used when the variable and its complement are present in the term, and the value of 2 is used in the map where the variable is present in the term.
3. For every product term of the fuzzy function, convert all variables to ternary form and perform the MIN operations on them.

4. After multiple-valued map for each product term is created, the cells which are covered by these products are MAX-ed together to create the function's multiple-valued map.

Fig. 10 explains the mapping between the fuzzy terms and terms in the MV map. A whole row or column of cells corresponds to a single variable. For instance, all cells in column 1 are for $x_2 x'_2$. The next example shows how converting a fuzzy function to MV function reduces the function to a canonical form, as example 5 showed for fuzzy maps.

Example 6. We use the same function as in example 5 to show how to convert a fuzzy function into an MV function. This example demonstrates that converting a fuzzy function into a MV function reduces to a canonical form because as shown in Fig. 12 and in Fig. 13 the results are the same.

The next section shows an example of taking a fuzzy function and converting it into a MV function to be decomposed and then converted back into two fuzzy functions.

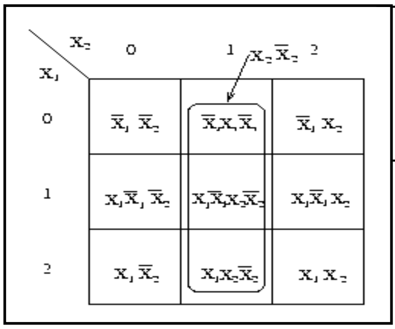
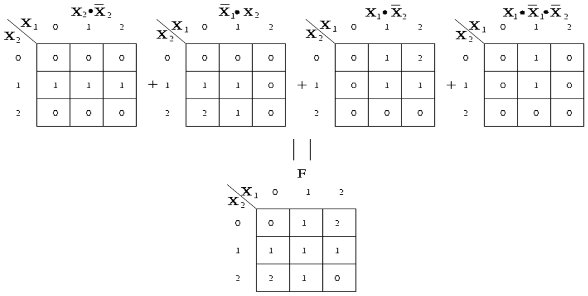


Figure 10. Conversion of fuzzy terms to Multiple-valued terms on variables x_1 and x_2 . For instance, fuzzy term = x_1 x'_1 + x_2 x'_2 is converted to $X_1{}^1$ $X_2{}^1$ and fuzzy term x_1 x'_1 x'_2 is converted to $X_1{}^1$ $X_2{}^2$



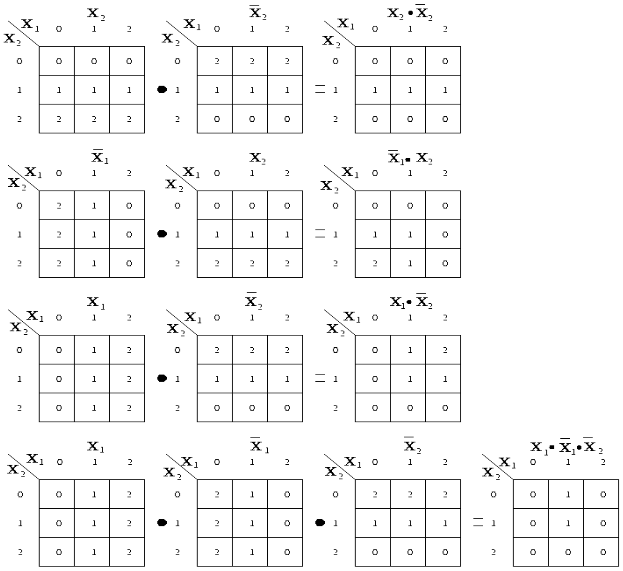


Figure 11. Conversion of a Fuzzy Function from Example 6 in non-canonical form to a Multiple-Valued Function.

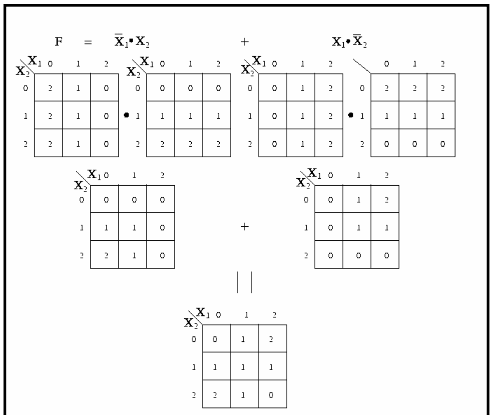


Figure 12. Conversion of a Fuzzy Function from Example 6 in a canonical form to a Multiple-Valued Function. Observe the same result as in Fig. 11.

5. DECOMPOSITION OF MULTIPLE-VALUED FUNCTIONS.

Example 7. *F(x, y, z) = yz + x' y' zz' + xz* . Fig. 13 shows the conversion of this three-variable fuzzy function to a ternary MV function of three ternary variables. The first map corresponds to *xz* , the second map corresponds to *x' y' z z'* and the third map corresponds to *yz* . The map on bottom is the maximum of the three maps above and it represents the ternary function *F(X, Y, Z)* to be decomposed. This function is converted to a format of the Input Function from Table 1. Next it is decomposed by our software. Simplified explanation of ternary decomposition follows. Let us observe in the bottom map from Figure 13 that for all possible cofactors *X*=i, *Y*=j , i, j = 0,1,2, the characteristic patterns 010, 011 and 012 exist. For instance, pattern 011 exists for *X*=0 and *Y*=1; *X*=1 and *Y*=1; *X*=1 and *Y*=0 . Pattern 010 exists only for X=0 and Y=0. Pattern 012 exists for all other combinations of *X* and *Y* values. Thus from [13,14] the function has three patterns for the bound set {*X, Y*} and is Ashenhurst decomposable, which means that only one intermediate (ternary) signal *G* is needed. The tables of functions *G* and *H* are the result of the MV decomposition. Such decomposed functions can be obtained using any of the two developed by us decomposers [13,14], or any other general-purpose or ternary decomposer.

6. CONVERTING THE MULTIPLE-VALUED FUNCTION TO THE FUZZY FUNCTION

The initial fuzzy function is converted to the multiple valued function and then decomposed to several interconnected multiple-valued functions called blocks. After completing the iterative multi level decomposition process of multi valued functions to non-decomposable blocks, [13,14], the block functions need to be converted back to fuzzy functions. The procedure to convert a multiple valued block function to a fuzzy function is:

1. Use multiple valued minimization to minimize the function option.
2. Convert the multiple valued product terms back into fuzzy product terms where each variable value $X_i = 1$ is converted into a variable and its complement $x_i x'_i$. Each variable value $X_i = 2$ is converted into the variable x_i and each variable value $X_i = 0$ is converted to the variable's complement x'_i .

The results of the decomposition process, functions G and H are shown in Fig. 15, a, b, respectively, as MV maps. Fuzzy terms Gz G'zz' and zz' of H are shown. Two solutions are obtained, G(x, y) = x + y, H(x, y) = Gz + zz' (Fig. 14c) and G(x, y) = x + y, H(x, y) = Gz + G' zz' (Fig. 14d). The correctness of these two decompositions can be verified by a reader by drawing all intermediate MV maps (as in Example 6) that create functions G and H , composing functions G and H back to F , and converting from ternary to fuzzy. In this case functions G and H are not decomposed further but in general these functions can be decomposed thus creating a tree or a Directed Acyclic graph of decomposed fuzzy blocks.

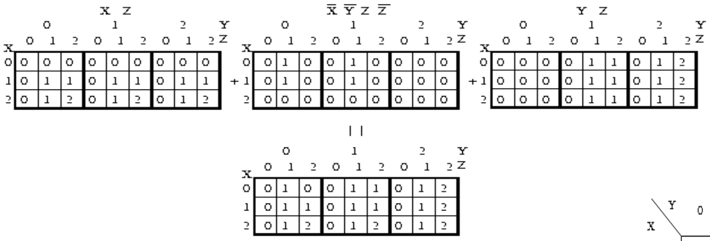


Figure 13. Conversion of three-variable fuzzy function F(x,y,z)=xz+x'y'zz'+yz from Example 7 to MV function.

Input Function

X	Y	Z	F
0	0	0	0
0	0	1	1
0	0	2	0
1	0	0	0
1	0	1	1
1	0	2	1
2	0	0	0
2	0	1	1
2	0	2	2
0	1	0	0
0	1	1	1
0	1	2	1
1	1	0	0
1	1	1	1
1	1	2	1
2	1	0	0
2	1	1	1
2	1	2	2
0	2	0	0
0	2	1	1
0	2	2	2
1	2	0	0
1	2	1	1
1	2	2	2
2	2	0	0
2	2	1	1
2	2	2	2

(a)

G Function

X	Y	G
0	1	1
1	0	1
2	0	2
0	0	0
1	2	2
2	2	2
0	2	2
1	1	1
2	1	2

(b)

H Function

Z	G	F
2	2	2
1	2	1
0	2	0
2	1	1
1	1	1
0	1	0
2	0	0
1	0	1
0	0	0

(c)

Table 1

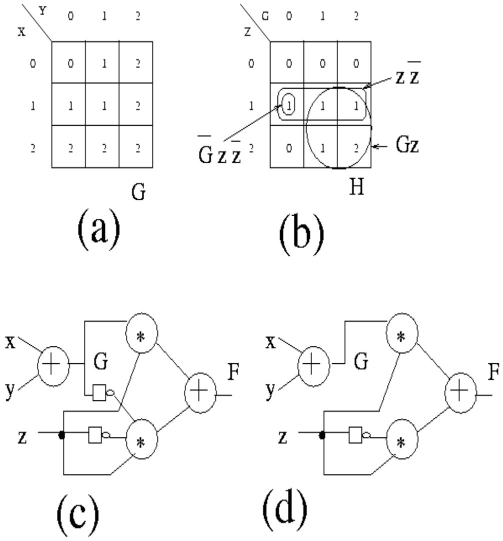


Figure 14. (a) Resulting MV Map of function G, (b) Resulting MV Map of function H, (c) First variant of decomposition, (d) Second variant of decomposition