An exact method to compute maximal implicants in a multivalued logic

ADRIAN ZAFIU[1] , ION STEFANESCU[2] , IONESCU LAURENTIU[3],

CONSTANTIN GHITA[4], EDUARD FRANTI[5]

[1]University of Piteşti, str. Târgul din Vale, Piteşti, România,
[2]University of Piteşti, str. Târgul din Vale, Piteşti, România,
[3]University of Piteşti, str. Târgul din Vale, Piteşti, România,
[4]University of Piteşti, str. Târgul din Vale, Piteşti, România,
[5] National Institute for Research and Development in Microtechnologies, Erou Iancu Nicolae St. 32B, Bucharest, Romania,

Summary: - The paper presents a method to determine maximal implicants in a multivalued logic. The method is used to minimize multivalued functions, complete or incomplete specified. In present, for bivalent functions are used Karnaugh diagrams or Quine-McCluskey method, and for multivalued functions are used software tools like Espresso, MVSIS2, BOOM II witch uses properties of binary decision diagrams (BDD) or SAT. The method presented here starts from a few remarks over the problem and transposes it into an equivalent problem having the computability proportional with the solution complexity.

Keyword- multivalued, minimization, implicants, algorithm, sets, truth table

1 Introduction

Multivalued logic synthesis is used in logical structures optimization of algorithms and digital circuit's synthesis [1].

Until now, there are well known software tools witch solve this kind of problems. Known methods, like:

1. diagram Veitch-Karnaugh;
2. method Quine McCluskey [2];
3. Espresso [3];
4. MVSIS [4];
5. discrimination method [5]

use total generative methods or heuristics to generate „almost optimal" solutions, without guarantee that thing.

Method 5 [5] uses a discrimination technique to generate all possible variants.

First two methods are exclusive binary. Method Espresso touches multivalued tables using a binary representation of multivalued values.

Method MVSIS is a multivalued method where every single variable could have its own domain and could handle even multiple outputs specifications. Variables are handled with binary coding.

Discrimination method is a multivalued method where variables are handled directly as multivalued entries and treats either simple, or multiple, as well as array outputs.

The method introduced in this paper uses a construction system of solutions, its complexity depending of formal complexity of minimal solutions for every output. System is a pure multivalued one and gives optimal solutions.

In principle, a constructive method has reduced computing complexity, in respect to generative methods.

Presented method is applied after the entrance specification (the value table) is analyzed from the point of view of its consistency.

This method results are used as input data for a next step witch completes the multivalued minimization, witch will be presented in a future paper.

2 Multivalued minimization

Presented method computes a set of maximal implicants group for a complete or an incomplete multivalued specification.

For example let's considerate specification given in Tab. 1:

	I_1	I_2	O
R1	1	1	0
R2	1	4	0
R3	3	3	0
R4	4	0	0
R5	4	5	0
R6	2	0	1
R7	4	2	1
R8	0	3	2

Tab. 1 Specification example

I have chosen especially this example because it has two inputs and could be graphically represented easily in two dimensions. Let consider Oy axis associated to I_1 and Ox axis associated to I_2. A two dimensional representation of this specification could be shown in Fig. 1.

	0	1	2	3	4	5
0	-	-	-	2	-	-
1	-	0	-	-	0	-
2	1	-	-	-	-	-
3	-	-	-	0	-	-
4	0	-	1	-	-	0

Fig. 1 2D representation for Tab. 1

In a 2D representation (Fig. 1) I have used the symbol „-" for unspecified zones. In every zone is placed the output value for that combination of inputs.

A minimization is composed of specified values and „-" (unspecified values). Unspecified values are also named DC - „don't care". In the measure a minimization result contains more DC and less rows, it is considered a better one. Every specified position involves an equivalence operator that could be implemented with a circuit component in hardware or with a new test ("if") in software.

Because our specification is simple one, a good minimization could be handled manually. Tab. 2 contains an appropriate set of coverage primes for 0 value of the function.

	I_1	I_2	C
1	1	-	R1, R2
2	3	-	R3
3	-	5	R5
4	4	0	R4

Tab. 2 Possible coverage set

This solution has been easily obtained, because the input table (initial specification) has a lower grade of complexity, and the dimension number of graphic representation is small (2 in that case), so the problem could be geometrically transposed. What happened, if dimension grows (higher than 3) and we could not give a geometrical interpretation? In that case we use one special tool like Espresso; MVSIS2; BOOM II witch uses properties of adjacent, binary decision diagrams (BDD) or SAT. Methods like this have a good time response for almost optimal solutions.

3 Problem statement

3.1 Theoretical aspects

Let's try to look from another angle over the problem. We must build biggest n-dimensional „volumes" witch could cover all inputs witch generate an output value, without including a set of coordinates witch involve another function value. In Fig. 2 it is shown an example of how could be eliminated the coordinate set (2,2) out of the entire space presented in Fig. 1. The coordinate set (-,-) represents all space and is equivalent with coordinate set ({0,1,2,3,4},{0,1,2,3,4,5}). DC has distinctive values for each dimension. DC represents all possible values for a given dimension, where it appears.

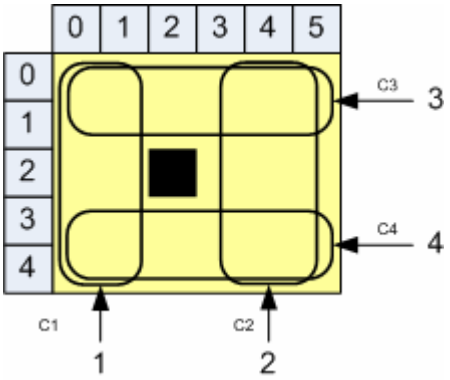

Fig. 2 Elimination

Elimination operation figured in Fig. 2 generates, by construction, 4 distinct zones named C1, C2, C3 and C4. They could be represented like coordinate spaces as:

1. C1=(-,{0,1})
2. C2=(-,{3,4,5})
3. C3=({0,1},-)
4. C4=({3,4},-)

Rel. 1 Coordinate sets for C1, C2, C3 and C4

The values {0,1,2,3,4} for Oy and {0,1,2,3,4,5} for Ox are not fixed in their positions. We have used digits symbols, ignoring the fact they have associated values that could be sorted. Thus, if we change the elements order inside the set, then Fig. 1 is converted into an equivalent representation given in Fig. 3. As we could see, Cx and Cy zones could be represented as coordinate spaces as follows:

1. Cx = (-,{0,1,3,4,5}) = C1+C2
2. Cy = ({0,1,3,4},-) = C3+C4

Rel. 2 Coordinate sets for Cx and Cy

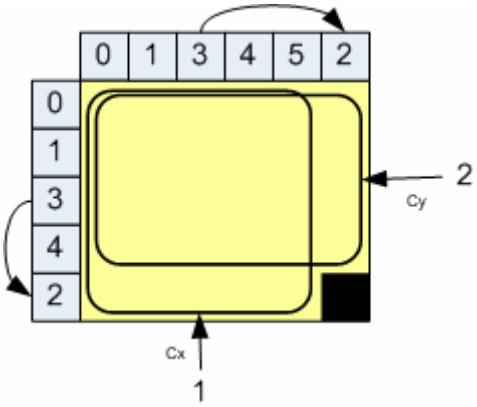

Fig. 3 Equivalent representation for Fig. 2

Practically Cx and Cy could be directly obtained by using an „elimination", implemented in a formal manner. If we eliminate (2,2) out of initial zone (-,-), that means we have to build zones witch doesn't intersect with (2,2).

Definition. Two zones doesn't intersect if each one of the effectuated intersection over the coordinates for every dimension of work space is a void set.

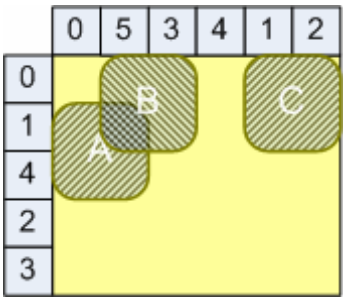

Fig. 4 Zones example

For example, in Fig. 4, zone A=({1,4},{0,5}) and zone B=({0,1},{3,5}) have zone (1,5) as an intersection, because it is common for both of them, and zone A doesn't intersect with C=({0,1},{1,2}), because over the dimension Cx the sets {0,5} and {1,2} doesn't intersect each other and generate disjoint zones.

Elimination operation that we need has to extract for every initial zone dimension, wherever it is possible, elements that are not part of the zone we want to discard. As there are two dimensions in this example, it means that we could obtain maximum two zones representing the result (Rel. 3).

(-,-) – (2,2) =
= ({0,1,2,3,4},{0,1,2,3,4,5}) - (2,2)=
= ({0,1,3,4},-)+ (-,{0,1,3,4,5})=Cy+Cx

Rel. 3 Elimination operation

3.2 Method description

An extract operation shown in **Error! Reference source not found.** could be used to extract such more zones. After first zone it was extracted in this example, we take the next zone to be extracted and apply the previous operation to each zone of the previous result. Out of the new obtained zones we keep the distinct elements only.

Applying the method over the example of Tab. 1, there is obtained the result presented in Tab. 3.

C0	**Z0**		**(-,-)**
	-R6		**(2,0)**
C1	**Z1**	**Z0-R6**	**({0,1,3,4},-)**
	Z2		**(-,{1,2,3,4,5})**
	-R7		**(4,2)**
C2	**Z3**	**Z1-R7**	**({0,1,3},-)**
	Z4		**({0,1,3,4},{0,1,3,4,5})**
	Z5	**Z2-R7**	**({0,1,2,3},{1,2,3,4,5})**
	Z6		**(-,{1,3,4,5})**
	-R8		**(0,3)**
C3	**Z7**	**Z3-R8**	**({1,3},-)**
	Z8		**({0,1,3},{0,1,2,4,5})**
	Z9	**Z4-R8**	**({1,3,4},{0,1,3,4,5})**
	Z10		**({0,1,3,4},{0,1,4,5})**
	Z11	**Z5-R8**	**({1,2,3},{1,2,3,4,5})**
	Z12		**({0,1,2,3},{1,2,4,5})**
	Z13	**Z6-R8**	**({1,2,3,4},{1,3,4,5})**
	Z14		**(-,{1,4,5})**

Tab. 3 Example with 3 eliminations

We could see that Z4, for example, could be reduced, because Z3 already contains a part of it. Z4 may be replaced, after C2 coverage was determined with (4,{0,1,3,4,5}). Also Z6 contains a part of Z4, witch allow replacement of Z4=(4,{0,1,3,4,5}) with (4,0). Zones Z3 and Z6 could be determined analyzing the DC positions number. It has a position with DC, confronted with Z4 and Z5 witch doesn't contain even one DC position.

The rule is: the intermediary result are sorted in a decreasing order of the DC number and it is tried to reduce them out of the zones having a les number of DC.

The next table contains an example treating the particularity of the problem.

C0	**Z0**		**(-,-)**
	-R6		**(2,0)**
C1	**Z1**	**Z0-R6**	**({0,1,3,4},-)**
	Z2		**(-,{1,2,3,4,5})**
	-R7		**(4,2)**
C2	**Z3**	**Z1-R7**	**({0,1,3},-)**
	Z4		**({0,1,3,4},{0,1,3,4,5})**
	Z5	**Z2-R7**	**({0,1,2,3},{1,2,3,4,5})**
	Z6		**(-,{1,3,4,5})**

C2'	Z3	Z1-R7	({0,1,3},-)
	Z4		(4,0)
	Z5	Z2-R7	(2,2)
	Z6		(-,{1,3,4,5})
	-R8		(0,3)
C3	Z7	Z3-R8	({1,3},-)
	Z8		({0,1,3},{0,1,2,4,5})
	Z9	Z4-R8	(4,0)
	Z10	Z5-R8	(2,2)
	Z11	Z6-R8	({1,2,3,4},{1,3,4,5})
	Z12		(-,{1,4,5})
C3'	Z7	Z3-R8	({1,3},-)
	Z8		(0,{0,2})
	Z9	Z4-R8	(4,0)
	Z10	Z5-R8	(2,2)
	Z11	Z6-R8	({2,4},3)
	Z12		(-,{1,4,5})

Tab. 4 Exemplu de extragere îmbunătățit

C3' zones are useless in this format because they have many redundancies. To eliminate this, every zone is divided in zones that contains on each dimension a value only, or a DC. Such zones are named primary zones. The results of the operation are presented in Tab.5.

C3'	Z7	({1,3},-)	→	Z13	(1,-)	R1, R2
				Z14	(3,-)	R3
	Z8	(0,{0,2})	→	Z15	(0,0)	-
				Z16	(0,2)	-
	Z9	(4,0)				R4
	Z10	(2,2)				-
	Z11	({2,4},3)	→	Z17	(2,3)	-
				Z18	(4,3)	-
	Z12	(-,{1,4,5})	→	Z19	(-,1)	R1
				Z20	(-,4)	R2
				Z21	(-,5)	R5

Tab. 5 Primary zones

In the folloing, primary zones are compared with zones which they had to in Tab. 1. The last column of Tab. 5 shows the covered zones for every primary zone.

Finally, we take zones Z9, Z10 and Z13, Z14, Z15 and so on to Z21, but from those are usable, according to the coverage presented by last column, Z9, Z13, Z14, Z19, Z20 and Z21 only. The minimization proposed in Tab. 2 referrs to Z13, Z14, Z21, Z9 zones.

Z13	(1,-)	R1, R2
Z14	(3,-)	R3
Z9	(4,0)	R4
Z19	(-,1)	R1
Z20	(-,4)	R2
Z21	(-,5)	R5

Tab. 6 Set of maximal implicouldts

3.3 Sufficiency

Suppose that a primary zone Z is part of maximal implicants set for which the function takes the value O. I will show the zone above will be found in the obtained result, using the presented method in 3.2 and it is not included in other primary zone having a larger coverage of the result.

The fact that is not included in a primary zone having a wider coverage is obviously, otherwise zone Z is not a maximal one.

The effectuated operation according ti the method has to eliminate the zones in which the function could not have O value. Let's consider U a zone that includes zone Z out of which we have to eliminate zone V (from input table). U is represented as a list of subsets of input value domains: $U(u_1u_2...u_n)$. Although Z is represented as subsets of input domains $Z(z_1z_2...z_n)$ with property that subsets are created by a single value or contain the entire sub domain. Like Z, V is represented as $V(v_1v_2...v_n)$.

If Z is a part of maximal implicants set, it means that Z doesn't intersect V, otherwise would be at least a sub zone from Z, even the entire zone, where the function has for V a value different of O. That means the Z is not an implicant for this value of the function.

We conclude that exist at least an integer i from [1...n] interval for which $z_i \cap v_i = \phi$. Knowing that Z is included in U zone, we could say that $z_j \cap u_j = z_j$, for any j from [1...n].

When we eliminate V out of U, at step i the obtained zone will is: $U_i(u_1, u_2, ...u_{i-1}, d_i, u_{i+1}, ..., u_n)$, where d_i contains z_i. u_i contains value z_i, and making difference between u_i and v_i, there have been eliminated the values of z_i, because $z_i \cap v_i = \phi$. As a conclusion, $d_i \cap z_i = z_i$. As the other zone elements are identical which those of zone U, that means $U_i \cap Z_i = Z_i$.

As we have at least a zone that contains Z, we could say that Z will be found in the result of the presented method.

From what we have presented in 3.3 we could say that the method is enough to generate a maximal implicants set .

4 Method complexity estimation

Suppose that we have an input table with n rows and k input columns. Operations to compute a set of maximal implicants involve maxim n operation of elimination. At each step the result could grow by maxim k times. So, in worst case, the complexity is equal with $O(k^n)$.

This complexity could be further reduced, if there is used the method presented in 3.2, which simplify the result at each step.

5 Conclusions

The method presented here gives an exact determination of the set of maximal implicants.

The computability complexity has a value les than $O(k^n)$.

The method uses a constructive system of solutions, which has a complexity depending only on the formal complexity of a minimal solution of each output value. The system is a pure multivalued one and leads to an optimal solution.

Each input or output variable can have its on value domain, there are allowed multiple outputs, and the input variables, as well the output ones, have either single or vectorial components.

A constructive method, as it is the one presented here, has the advantage of a reduced computability complexity, in respect to the generative methods.

The method is applicable after a consistency check is done over the input value table. Such a check is presented in (ref. 6).

In case of a great number of inputs, the method is less efficient, if the number of rows is very large. In such a case, a heuristic method represents a more convenient approach. Anyway, such a problem is inherent to the other known methods, too.

6 Future work

This method, combined with other techniques developed to manipulate this kind of matrix, is used to create a system for minimization of multivalued functions with multiple outputs. To compute one solution for each output could be inefficient, because this approach does not allow correlations between the variants of output solutions. This correlation is needed to compute an optimal global solution.

Next algorithms to compute appropriated solutions for each output in a compact form is designed such the number of common implicant vectors is maximum or as great as possible. Thus, an optimal global implementation of the specified system is attained.

Considering a multivalued function (representing a conversion of decimal digits from Z3 to Z5) and its solutions, there is obvious that its implementation is using 12 input configurations only, because some of them are common for both outputs.

Function	Solution for first output	Solution for second output
0 0 0/0 0	- 1 1 / 0	- 1 2 / 0
0 0 1/0 1	- 1 0 / 0	0 0 0 / 0
0 0 2/0 2	0 0 - / 0	
0 1 0/0 3		- 2 0 / 1
0 1 1/0 4	1 - - / 1	- 0 1 / 1
0 1 2/1 0	- 1 2 / 1	
0 2 0/1 1	- 2 - / 1	- 2 1 / 2
0 2 1/1 2		- 0 2 / 2
0 2 2/1 3		
1 0 0/1 4		- 2 2 / 3
		- 1 0 / 3
		1 - - / 4
		- 1 1 / 4

Tab. 7 A multivalued function and its solutions

An independent implementation of the function involves 6 input configurations for first input and 10 input configurations for second outputs (a total of 16 input configurations).

Moreover, the solutions of each output value could be helpful to compute a better global solution. For example, a classical bivalent function solution given in Tab. 7 (a BCD to 7 segment decoder) has 14 implicants, either for first, or second canonical form.

BCD Converter
0 0 0 0 / 1 1 1 1 1 1 0
0 0 0 1 / 0 1 1 0 0 0 0
0 0 1 0 / 1 1 0 1 1 0 1
0 0 1 1 / 1 1 1 1 0 0 1
0 1 0 0 / 0 1 1 0 0 1 1
0 1 0 1 / 1 0 1 1 0 1 1
0 1 1 0 / 0 0 1 1 1 1 1
0 1 1 1 / 1 1 1 0 0 0 0
1 0 0 0 / 1 1 1 1 1 1 1
1 0 0 1 / 1 1 1 0 0 1 1

Tab. 8 BCD 7 segments

In table Tab. 9 there are shown solutions for each segment. Segments 0 and 6 have two equivalent solutions for output 1. Implicant are counted from 1 to 22. For example, implicant 02 appear in five places. The remark could be used to select an optimal

global solution. In the table, marked vectors have more than one apparition.

Seg 0	Seg 1	Seg 2	Seg 3
01)- 0 - 0 / 1	08)- - 0 0 / 1	12)- - - 1 / 1	16)- - 1 0 / 1
02)1 - - - / 1	09)- 0 - - / 1	13)- - 0 - / 1	03)- 0 1 - / 1
03)- 0 1 - / 1	05)- - 1 1 / 1	14)- 1 - - / 1	01)- 0 - 0 / 1
04)- 1 - 1 / 1			10)- 1 0 1 / 1
	10)- 1 0 1 / 0	15)- 0 1 0 / 0	
01)- 0 - 0 / 1	11)- 1 1 0 / 0		17)- 1 0 0 / 0
02)1 - - - / 1			18)- 1 1 1 / 0
04)- 1 - 1 / 1			19)- 0 0 1 / 0
05)- - 1 1 / 1			
06)- 1 - 0 / 0			
07)0 0 0 1 / 0			

Seg 4	Seg 5	Seg 6
16)- - 1 0 / 1	08)- - 0 0 / 1	18_ - 1 1 1 / 0
01)- 0 - 0 / 1	02)1 - - - / 1	22)0 0 0 - / 0
	20)- 1 0 - / 1	
12)- - - 1 / 0	06)- 1 - 0 / 1	06)- 1 - 0 / 1
20)- 1 0 - / 0		02)1 - - - / 1
	05)- - 1 1 / 0	20)- 1 0 - / 1
	03)- 0 1 - / 0	03)- 0 1 - / 1
	21)0 0 - 1 / 0	
		02)1 - - - / 1
		16)- - 1 0 / 1
		20)- 1 0 - / 1
		03)- 0 1 - / 1

Tab. 9 Solutions for each segment

Considering a solution that combine the first, as well the second canonical form, the general solution is reduced to 11 implicants, which means 21% amelioration of previous solutions of the examined multiple output function.

Seg 0	Seg 1	Seg 2	Seg 3
01)- 0 - 0 / 1	10)- 1 0 1 / 0	15)- 0 1 0 / 0	16)- - 1 0 / 1
02)1 - - - / 1	11)- 1 1 0 / 0		03)- 0 1 - / 1
03)- 0 1 - / 1			01)- 0 - 0 / 1
04)- 1 - 1 / 1			10)- 1 0 1 / 1

Seg 4	Seg 5	Seg 6
16)- - 1 0 / 1	08)- - 0 0 / 1	06)- 1 - 0 / 1
01)- 0 - 0 / 1	02)1 - - - / 1	02)1 - - - / 1
	20)- 1 0 - / 1	20)- 1 0 - / 1
	06)- 1 - 0 / 1	03)- 0 1 - / 1

Tab. 10 Optimal solution for BCD 7 segments

Such an approach is allowed by the global presentation of each output solution, containing all equivalent solution versions for each value, let it be binary, or multivalued.

References

[1] S. Hassoun and T. Sasao, "*Logic Synthesis and Verification*",Boston, MA, Kluwer Academic Publishers, 2002, 454 pp.

[2] E.J. McCluskey, "*Minimization of Boolean functions*", The Bell System Technical Journal, 35, No. 5, Nov. 1956, pp. 1417-1444

[3] P. McGeer et al., "*ESPRESSO-SIGNATURE: A new exact minimizer for logic functions*", Proc. DAC'93

[4] Donald Chai, Jie-Hong Jiang, Yunjian Jiang, Yinghua Li, Alan Mishchenko, Robert Brayton , *„MVSIS 2.0 User's Manual"*, Department of Electrical Engineering and Computer Sciences University of California, Berkeley CA 94720 , 2003

[5] Ion Ştefănescu, "*DISCRIMINATION: A New Principle in the Efficient Minimization of the Binary and Multiple-Valued Functions"*, 1st International Conference on Electronics, Computers and Artificial Intelligence – ECAI 2005 – University of Piteşti, Romania, 1-2 July 2005.