ON THE CALCULATION OF GENERALIZED REED-MULLER CANONICAL EXPANSIONS FROM DISJOINT CUBE REPRESENTATION OF BOOLEAN FUNCTIONS

Bogdan J. Falkowski,

Marek A. Perkowski.

Department of Electrical Engineering,
Portland State University,
P.O. Box 751,
Portland, OR 97207.
phone (503) 725-3806

ABSTRACT

A new algorithm is shown that converts disjoint cube representation of Boolean functions into fixed-polarity Generalized Reed-Muller Expansions (GRME). Since the known fast algorithm that generates the GRME based on the factorization of the Reed-Muller transform matrix always starts from the truth table (minterms) of Boolean function, then the described method has the advantages due to smaller required computer memory. Moreover, for the Boolean functions described by only few disjoint cubes the method is much more efficient than the fast algorithm.

1. INTRODUCTION

The classical approach to analysis, synthesis and testing of digital circuits is based on the description by the operators of Boolean algebra. However, for many years, an alternative description based on the operations of modulo-2 arithmetic has been developed [1, 8]. The algebra corresponding to this second approach, being an example of a finite field, supports such familiar for digital signal processing operations as matrices, and fast transforms. It is obvious, that this modulo-2 algebra is the simplest case of an algebra known in the literature [1] as a finite field or Galois field algebra and it is why it is frequently denoted by the symbol $GF(2)$.

Any Boolean function can be represented in the modulo-2 algebra. The modulo-2 sum-of-products expression is known in the literature [1, 8, 14] as the *complement-free ring-sum* or *Reed-Muller* expansion. In such an expression there is 2^n possible product terms selected from the n variables of a Boolean function, each of the terms having only the not complemented variables of a Boolean function. By allowing the complementation of the input variables one can derive the *Generalized Reed-Muller* expansions (GRMEs) where each input variable x_i can appear either true throughout the expansion or complemented throughout it. It is apparent that there exist 2^n GRMEs for a Boolean function with n variables, each with different polarity, including the positive polarity form (i.e., Reed-Muller expansion). Since the polarity of an input is constant in a GRME these expansions are termed fixed-polarity forms. It should be noted that for a given Boolean function each GRME is unique and forms a *canonical form.*

The application of exclusive OR gates has some advantages over the gates used commonly in logic design. One of the reasons is that many useful functions have a high content of the so called linear part (EXOR part of the function). Some of the examples of such functions are: adders and parity checkers. What is more, the circuit build around the EXOR gates is easily testable. Reddy showed, that when the circuit is represented as a Reed-Muller expansion then, if only permanent stuck-at faults occur in either a single AND or only a single EXOR gate is faulty, only $(n+4)$ tests are required for fault-free primary inputs for an arbitrary n input Boolean function [12].

2. BASIC DEFINITIONS AND PROPERTIES

The properties of disjoint cube representation of Boolean functions used in the description of the algorithm calculating the GRME are stated. An algorithm that generates such a representation has been shown in [4] and its implementation has been described in [5]. In what follows, the definitions of the RME [8] and GRME are also presented.

Definition 1: The *n-variable Reed-Muller expression* takes the form:

$$F\ (x_n, x_{n-1}, \ldots, x_1) = \sum_{i=1}^{2^n-1} c_i\ x_n^{e_{i,n}}\ x_{n-1}^{e_{i,n-1}}, \ldots, x_1^{e_{i,1}}$$

In the above definition $\sum$ means summation over $GF(2)$, and the $e_{i,j}$ are either 0 or 1 so that literal $x_k^0 = 1$ and $x_k^1 = x_k$.

Property 1: The n-variable Reed-Muller expression has 2^n *possible product terms (piterms represented by symbol* π*)* selected from the n variables.

Property 2: The subscript i in the piterm π_i represents the decimal equivalent of the straight binary code (SBC) formed from the join of symbols $e_{i,n}, e_{i,n-1}, \ldots, e_{i,1}$, where the most significant bit of the SBC is $e_{i,n}$ and the least significant bit of the SBC is $e_{i,1}$.

Definition 2: The n-variable Reed-Muller expression can be represented in short by:

$$F\ (x_n, x_{n-1}, \ldots, x_1) = \sum_{i=1}^{2^n-1} c_i\ \pi_i,$$

where the meaning of the summation is exactly the same as in Definition 1.

The next property is the consequence of Definition 1 and Definition 2.

Property 3: The n-variable Reed-Muller expression is fully described by the set of all c_i Reed-Muller spectral coefficients.

Definition 3: The *n-variable Generalized Reed-Muller expression* takes the form:

$$F\ (x_n, x_{n-1}, \ldots, x_1) = \sum_{i=1}^{2^n-1} c_i\ x_n^{e_{i,n}^{\omega}}\ x_{n-1}^{e_{i,n-1}^{\omega}}, \ldots, x_1^{e_{i,1}^{\omega}}$$

In the above definition $\sum$ means summation over $GF(2)$, and the $e_{i,j}^{\omega}$ are either 0, 1, or -1 so that literal $x_k^0 = 1$, $x_k^1 = x_k$, and $x_k^{-1} = \overline{x_k}$. The symbol ω denotes the polarity of the GRME.

Property 4: The n-variable Generalized Reed-Muller expression has 2^n *possible product terms (generalized piterms represented by symbol* π^{ω}*)* selected from the n variables.

Definition 4: The *polarity number of the GRME expression denoted by* ω is a binary string computed by taking the n bit straight binary code (SBC) formed by writing a 0 or a 1 for each variable according to whether it is used in affirmative or negative form respectively.

Property 5: For a given polarity ω each variable in all generalized piterms can be either in affirmative, negative form or be absent from the piterm.

Definition 5: The n-variable Generalized Reed-Muller expression can be represented in short by:

$$F\ (x_n, x_{n-1}, \ldots, x_1) = \sum_{i=1}^{2^n-1} c_i\ \pi_i^{\omega},$$

where the meaning of the summation is exactly the same as in Definition 3.

The next property is the consequence of Definition 3 and Definition 5.

Property 6: The n-variable Generalized Reed-Muller expression is fully described by the set of all c_i Reed-Muller spectral coefficients.

Definition 6: The *cube of degree m* is a cube that has m defined literals that can be either affirmative or negative (i.e., m is equal to the sum of the number of zeros and ones in the description of a cube).

Let symbol p denote the number of X's in the cube (for explanation of the symbol X see Definition 9) and n denote the number of variables of a given Boolean function F. Then, $n = m + p$.

Definition 7: The cube and the piterm are of the *same degree (order)* when the number of defined literals is the same for both of them.

Property 7: The number of piterms of z-*th* order is equal to $C_n^z = \binom{n}{z}$, where n is the number of variables of a Boolean function.

Definition 8: The *partial Reed-Muller spectral coefficients* of an ON- cube cu_i of degree m from disjoint cube representation of a Boolean function F are those parts of the final Reed-Muller spectral coefficients c_i that correspond to the contribution of the cube cu_i of degree m to full $n-$ space Reed-Muller spectrum of the Boolean function F described by the array of all the disjoint cubes.

Property 8: Each final Reed-Muller spectral coefficient is obtained by adding modulo 2 all partial coefficients of all the disjoint cubes describing the function.

Property 9: The number of partial Reed-Muller spectral coefficients *npsc* describing the Boolean function F is equal to the number of ON- cubes describing this function times 2^n.

Property 10: When the Boolean function is described by its truth table (a set of true and false minterms) then the number of partial spectral coefficients of the Boolean function in this representation is equal to the sum of the number of ON- minterms times 2^n. Let symbol f denote the number of false minterms. Then, $npsc = 2^n\ (2^n - f\ )$.

In Table 3 and 4 each row of symbols "0" and "1" next to the cube represents values of partial spectral coefficients of a given cube. The order of each of these coefficients is the same as the order of the piterm placed in the top of the column above the particular partial spectral coefficient.

Definition 9: A cube is represented by a *positional notation* where "0" corresponds to a negated literal, "1" to the affirmative literal, and "X" to the literal that is absent in a given cube.

For example, the cube $x_1\,\overline{x_3}\,x_4$ is represented by $1X01$.

Definition 10: A piterm π_i^ω is represented by a *positional notation* that is the same as in Definition 9 for cubes.

For example, the piterm $\pi_{134}^{0101} = x_1\,x_3\,\overline{x_4}$ is represented by $1X10$.

Definition 11: A cube *adjusted to a polarity* ω is such a cube that is build of these literals of the original cube that are different than the bits of the polarity ω in the SBC. The same literals are marked by the symbol "-".

The adjustment operator is defined in Table 1. When the cube $cu_i = 1\ X\ 0\ 1$ is adjusted to the polarity $\omega = 0000$ then it is equal to $cu_i^{0000} = 1X-1$, and when the same cube is adjusted to the polarity $\omega = 0101$ then it is equal to $cu_i^{0101} = 1X--$.

Definition 12: A cube and a piterm of the same degrees *match exactly* when both of them have the same literals in the same polarity ω or the cube has some literals that have been created by the adjustment operation and are marked by "-".

The exact matching operator is defined for bits of a piterm and a cube in positional notation in Table 2. When a cube is previously adjusted to some polarity then the value of the exact matching operator for the bits of the adjusted cube marked by symbols "-" is always 1 independently of the value of the piterm. It is shown in the fourth row of Table 2. From this point on it will be understood that the exact matching operation is defined by the whole Table 2.

Definition 13: A cube and a piterm of the same degrees *match in all but one literal* when there exists one literal in the piterm that is negated in the cube and the rest of the literals in both the cube and piterm match exactly.

The above definition can be extended to a more general case.

Definition 14: A cube and a piterm of the same degrees *match in all but k literals* when there exist k literals in the piterm that are negated in the cube and the rest of the literals in both cube and piterm match exactly.

Definition 15: A cube and a piterm of the same degree *match opposing* when all the literals in the piterm are in negation to the literals of the cube, i.e., literals in the cube are in the polarity $\overline{\omega}$ ($\overline{\omega}$ means bit by bit negation).

Definition 16: A *set of expanded piterms* of a given cube and a piterm that match in all but k literals is composed of the highest order piterm that matches exactly and all the higher order piterms that are composed of the literals of the piterm that matches exactly and are expanded by the remaining literals of the cube in such a way that the obtained piterms match the given cube in all but one, all but two, ..., all but k literals.

Property 11: A set of expanded piterms from Definition 16 has 2^k members.

Property 12: All members of the set of expanded piterms match exactly the adjusted cube for which they are generated.

Property 13: The order of the highest order member of the set of expanded piterm of a given cube is not higher than the order of the adjusted cube itself.

The next property is a consequence of Property 13.

Property 14: The number of the piterms that has to be checked for exact matching in order to generate the whole set of expanded piterms for an adjusted cube of order z is equal to $C_n^1 + C_n^2 + C_n^3 + ,\ ,\ , + C_n^z$, where n is the number of variables of a Boolean function. It is obvious that $n > z$.

3. ALGORITHM TO GENERATE GRME FROM DISJOINT CUBES

The algorithm refers to properties and definitions from the previous section. It is assumed that the Boolean function is described by an array of disjoint ON- cubes that can be generated by the algorithms presented in [4, 5]. During execution of the algorithm the values of the partial spectral coefficients corresponding to adjusted cubes are stored in a temporary array. The dimension of the array is $m \times 2^n$, where m is a number of disjoint cubes that represent an n variable Boolean function.

Algorithm 1: **Generation of GRME from disjoint cubes.**

```
step 1.
    Set a number of cubes and all piterms to a given polarity ω.
step 2.
    Adjust all cubes to a given polarity ω.
step 3.
    For each adjusted cube generate a set of all piterms that match exactly.
    Operation of exact matching starts from the piterms having the same
    order as the cube and continues for piterms of all smaller orders.
    If all cubes are matched go to step 4 else repeat step 3.
step 4.
    Add modulo 2 each column of the temporary array. The result
    describes Reed-Muller spectral coefficients of the Boolean function.
```

The detailed example of the execution of this algorithm is shown next.

Example 1: An example of the calculation of Reed-Muller canonical expansion for a four variable completely specified Boolean function described by a set of disjoint cubes in positional notation is shown in Table 6 and Table 7. In order to obtain the values of all Reed-Muller spectral coefficients the columns of the partial spectral coefficients from the temporary array corresponding to all cubes describing a given Boolean function are added modulo 2.

Let us show the execution of all steps of the algorithm simultaneously for both tables. First, the number of cubes is determined to be 6. The piterms for Table 6 are in the polarity $\omega = 0000$, and for Table 7 in the polarity $\omega = 0101$. For short, the piterms from both tables have the same symbols. It should be noticed, however, that piterm π_{1234} from Table 6 corresponds to the group $x_1\,x_2\,x_3\,x_4$ while piterm π_{1234} from Table 7 corresponds to the group $x_1\,\overline{x_2}\,x_3\,\overline{x_4}$, etc.

Secondly, all the cubes are adjusted to the given polarities. The results of this operation for both tables are shown in Table 3.

In the third step, the exact matching operation generates the expanded set of piterms. The result of this operation for the adjusted cubes from Table 6 is shown in Table 4 and Table 7 in Table 5.

For the sake of explanation the Table 4 is rewritten to Table 6, and Table 5 to Table 6. Let us notice that both pairs of tables content the same information in a different notation. First the columns cube in Table 6 and Table 7 correspond to the cubes before adjustment (see Table 3 and Table 4). Secondly, the members of the set of expanded piterms from Table 5 and Table 6 are marked in Table 6 and Table 7 by 1-es accordingly. In both tables all the partial coefficients are given for each cube - not only those that have values 1 (that mark the piterms from the set of expanded piterms) but also the partial spectral coefficients that have values 0.

The last step of the algorithm is the operation of addition modulo 2 of all the partial spectral coefficients. The result of this operation is shown in the bottom rows of Table 6 and Table 7.

The resulting 4-variable Reed-Muller expression for Table 1 takes the form (the symbol "+" stands for the sum modulo 2):

$$F\ (x_1, x_2, x_3, x_4) = x_1 + x_2 + x_1\,x_2 + x_1\,x_4 + x_3\,x_4 + x_1\,x_2\,x_4 + x_2\,x_3\,x_4.$$

The resulting 4-variable Generalized Reed-Muller expression for Table 2 is as follows (the symbol "+" has the same meaning as above):

$$F\ (x_1, \overline{x_2}, x_3, \overline{x_4}) = 1 + \overline{x_2} + \overline{x_2}\,x_3 + x_1\,\overline{x_2}\,\overline{x_4} + \overline{x_2}\,x_3\,\overline{x_4}$$

One can notice, that the original set of disjoint cubes can be taken directly as one possible mixed polarity Generalized Reed-Muller form [9-11] of the considered Boolean function.

4. CONCLUSION

In this paper, a new algorithm that generates any fixed-polarity GRM expressions from the disjoint cube representation of Boolean functions is shown. The Boolean functions are represented in the form of arrays of disjoint cubes [4, 5]. For each cube the appropriate partial GRME is obtained. By adding modulo-2 all the entries corresponding to the full set of disjoint cubes describing the given Boolean function its fixed-polarity GRME is found. This algorithm can be executed in parallel.

The algorithm presented in this paper is similar to the algorithms for the calculation of Hadamard-Walsh spectra of Boolean functions developed previously by the authors [2, 3]. It is possible also to develop similar algorithms for Arithmetic and Adding transforms [7] and for Walsh transforms of multiple-valued input binary functions [6]. All these transforms have been recently introduced by the authors [6, 7]. When the Boolean function has a big number of literals and is described by only few disjoint cubes then the presented algorithm is very efficient in comparison to the fast algorithm based on the transform matrix factorization [13]. The presented algorithm can be expanded for the case of mixed-polarity GRME [9, 10] and it is the topic of the ongoing research of the authors.

5. ACKNOWLEDGMENT

This research was financially supported in part by the Faculty Development Grant 90-050-1009-2000 from Portland State University. The attendance at this conference was supported by a travel grant from the Academically-Controlled Auxiliary Activities Committee at Portland State University.

6. REFERENCES

[1] P. Davio, J.P. Deschamps, and A. Thayse, *Discrete and Switching Functions*. New York: George and McGraw-Hill, 1978.

[2] B. J. Falkowski, and M. A. Perkowski, "One more method for calculating the Hadamard-Walsh spectrum for completely and incompletely specified Boolean functions," Accepted for publication in *Int. J. of Electronics*, 1990.

[3] B. J. Falkowski, and M. A. Perkowski, "Algorithms for the calculation of Hadamard-Walsh spectrum for completely and incompletely specified Boolean functions," *Proc. of IEEE Int. Phoenix Conf. on Computers & Communications (9th IPCCC)*, Scottsdale, AR, pp. 868-869, March 1990.

[4] B. J. Falkowski, and M. A. Perkowski, "An algorithm for the calculation of disjoint cube representation of completely and incompletely specified Boolean functions," Accepted for publication in *Int. J. of Electronics*, 1990.

[5] B. J. Falkowski, I. Schäfer, and M. A. Perkowski, "A fast computer algorithm for the generation of disjoint cubes for completely and incompletely specified Boolean functions," *Proc. of IEEE Midwest Symp. on Circuits & Systems (33rd MIDSCAS)*, Calgary, Alberta, August 1990.

[6] B. J. Falkowski, and M. A. Perkowski, "Walsh type transforms for completely and incompletely specified multiple-valued input binary functions," *Proc. of IEEE Int. Symp. on Multiple-Valued Logic (20th ISMVL)*, Charlotte, NC, pp. 75-82, May 1990.

[7] B. J. Falkowski, and M. A. Perkowski, "A family of all essential radix-2 addition/subtraction multi-polarity transforms: algorithms and interpretations in Boolean domain," *Proc. of IEEE Int. Symp. on Circuits & Systems (23rd ISCAS)*, New Orleans, LA, pp. 2913-2916, May 1990.

[8] D. Green, *Modern Logic Design*. Wokingham: Addison-Wesley, 1986.

[9] M. Helliwell, and M. A. Perkowski, "A fast algorithm to minimize multi-output mixed-polarity generalized Reed-Muller forms," *Proc. of ACM/IEEE Design Automation Conf. (25th DAC)*, Anaheim, CA, pp. 427-432, June 1988.

[10] M. Perkowski, M. Helliwell, and P. Wu, "Minimization of multiple-valued input multi-output mixed-radix exclusive sums of products for incompletely specified Boolean functions," *Proc. of IEEE Int. Symp. on Multiple-Valued Logic (19th ISMVL)*, pp. 256-263, Guangzhou, China, 1989.

[11] M. A. Perkowski, P. Dysko, B. J. Falkowski, "Two learning methods for a tree-search combinatorial optimizer", *Proc. of IEEE Int. Phoenix Conf. on Computers & Communications (9th IPCCC)*, Scottsdale, AR, pp. 606-613, March 1990.

[12] S. M. Reddy, "Easily testable realizations for logic functions," *IEEE Trans. Comput.*, vol. C-21, pp. 1183-1188, Nov. 1972.

[13] B.R.K. Reddy, and A.L. Pai, A.L, "Reed-Muller transform image coding," *Computer Vision, Graphics, and Image Processing*, vol. 42, pp. 48-61, 1988.

[14] A. Tran, "Graphical method for the conversion of minterms to Reed-Muller coefficients and the minimization of exclusive-OR switching functions," *Proc. IEE*, vol. 134, Pt. E, No. 2, 1987

	Polarity	
Cube	0	1
0	-	0
1	1	-
X	X	X

Table 1. Adjustment operator between bits of a cube in positional notation and polarity in the SBC.

	Piterm		
Cube	0	1	X
0	1	0	0
1	0	1	0
X	0	0	1
-	1	1	1

Table 2. Exact matching operator between bits of a cube and a piterm in positional notation.

Cube	Table 6	Table 7
	0000	0101
1X00 ON	1X--	1X-0
1X1X ON	1X1X	1X1X
0X11 ON	-X11	-X1-
010X ON	-1-X	---X
1101 ON	11-1	1---
0110 ON	-11-	--10

Table 3. Cube adjustment to polarities 0000 and 0101.

Cube	Piterm			
1X--	1XXX	1X1X	1XX1	1X11
1X1X	1X1X			
-X11	XX11	1X11		
-1-X	X1XX	11XX	X11X	111X
11-1	11X1	1111		
-11-	X11X	111X	X111	1111

Table 4. Generation of expanded set of piterms by exact matching operation.

Cube	Piterm							
1X-0	1XX0	1X10						
1X1X	1X1X							
-X1-	XX1X	1X1X	XX10	1X10				
---X	XXXX	1XXX	X0XX	XX1X	10XX	1X1X	X01X	101X
1---	1XXX	10XX	1X1X	1XX0	101X	10X0	1X10	1010
--10	XX10	1X10	X010	1010				

Table 5. Generation of expanded set of piterms by exact matching operation.

	π_0	π_1	π_2	π_3	π_4	π_{12}	π_{13}	π_{14}	π_{23}	π_{24}	π_{34}	π_{123}	π_{124}	π_{134}	π_{234}	π_{1234}
1X00 ON	0	1	0	0	0	0	1	1	0	0	0	0	0	1	0	0
1X1X ON	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0X11 ON	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0
010X ON	0	0	1	0	0	1	0	0	1	0	0	1	0	0	0	0
1101 ON	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
0110 ON	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	1
	0	1	1	0	0	1	0	1	0	0	1	0	1	0	1	0

Table 6. Reed-Muller spectral coefficients for polarity 0000.

	π_0	π_1	π_2	π_3	π_4	π_{12}	π_{13}	π_{14}	π_{23}	π_{24}	π_{34}	π_{123}	π_{124}	π_{134}	π_{234}	π_{1234}
1X00 ON	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0
1X1X ON	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0X11 ON	0	0	0	1	0	0	1	0	0	0	1	0	0	1	0	0
010X ON	1	1	1	1	0	1	1	0	1	0	0	1	0	0	0	0
1101 ON	0	1	0	0	0	1	1	1	0	0	0	1	1	1	0	1
0110 ON	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1
	1	0	1	0	0	0	0	0	1	0	0	0	1	0	1	0

Table 7. Reed-Muller spectral coefficients for polarity 0101.