Evaluation of m-valued Fixed Polarity Generalizations of Reed-Muller Canonical Form

Elena Dubrova
Electronic System Design Lab
Department of Electronics
Royal Institute of Technology
S-164 40 Kista, Sweden
elena@ele.kth.se

Abstract

This paper compares the complexity of three different fixed polarity generalizations of Reed-Muller canonical form to multiple-valued logic: the Galois Field-based expansion introduced by Green and Taylor, the Reed-Muller-Fourier form of Stanković and Moraga, and the expansion over addition modulo m, minimum and the set of all literal operators introduced by the author and Muzio. An algorithm for computing the minimal canonical forms for these generalizations is implemented and applied to a set of encoded 4-valued benchmark functions, 3- and 4-valued adders and multipliers. The experimental results show that, for the benchmark functions, the Reed-Muller-Fourier form and our expansion yield a comparable number of products on average. They have 40% less products on average than the expansion of Green and Taylor. The Reed-Muller-Fourier form gives a compact representation for adders, while our expansion seems to be suitable for multipliers.

1. Introduction

Two-level expressions of multiple-valued logic functions and their minimization has been a subject of active research for many years. This problem is important because it provides a means for optimizing the implementations of circuits that are direct translations of two-level expressions ([1]-[4]). Thus, two-level logic representations have direct impact on macro-cell design styles using programmable logic arrays (PLAs).

We can classify two-level expressions of multiple-valued logic functions into two major groups. The first one includes logic representations evolved from classical sum-of-products form of Boolean functions over a Boolean algebra. An example is the canonical form of multiple-valued functions in a Post algebra over the set of operations (MIN, MAX, literal) [5]. The second group includes logic representations based on generalization of modulo 2 polynomial over a Galois Field $GF(2)$, often referred to as *Reed-Muller canonical form* [6], [7]. In this paper we consider three different generalizations of the Reed-Muller canonical form to multiple-valued logic and compare their complexity.

The Reed-Muller canonical form can be extended to multiple-valued logic in several ways, depending on how its operations are generalized. Many extensions have been suggested, including [8]-[13]. In these extensions, AND and XOR operations (which are equivalent to multiplication and addition modulo 2, correspondently) are generalized to addition and multiplication in $GF(m)$. This is a natural extension, allowing to preserve the nice properties of fields. However, such generalizations are only applicable to algebras with m being a prime or a power of a prime number. This doesn't cover, for example, the interesting case of $m = 10$. In [14] we introduced another generalization of the fixed polarity Reed-Muller canonical form, where AND is extended to MIN operation, and XOR is extended to addition modulo m. Since MIN is not distributive over addition modulo m, such an algebraic structure is not a field any longer, but it is applicable to any positive integer m.

From an implementation point of view, the MIN operation is easier to implement than the multiplication in $GF(m)$. For example, in current-mode CMOS technology, MIN can be implemented using 5 transistors only [15]. Moreover, this implementation is independent of m, i.e. the number of transistors does not increase with increasing values of m. On the other hand, the implementation of multiplication in $GF(m)$ is larger and it always depends on m. For example, for $m = 3$, a current-mode CMOS circuit realizing multiplication modulo 3 consists of 16 transistors

[16].

However, this implementation advantage can only be utilized if the complexity of MIN-based expression of an m-valued function is not larger than the complexity of the $GF(m)$ multiplication-based expression. The purpose of this paper is to compare the complexity of our canonical form to the complexity of the canonical forms introduced in [11] and [13]. As a measure of complexity we use the number of products and the number of literals per term in the expression. This choice is motivated by the major optimization objectives in combinational logic design, which are *area* and *delay* reduction [17]. When considering a two-level implementation of a sum-of-products-type expression by a PLA, each row of a PLA is in one-to-one correspondence with a product-term of the sum-of-products expression. Each transistor in a row relates to the literals of a product-term. Therefore, the primary goal of logic optimization is the reduction of the number of products, and a secondary one is the reduction of the number of literals. Achieving a sum-of-products representation with the minimal number of products and with the minimal number of literals corresponds to optimizing both area and delay. So, these two parameters are good measures of the complexity of a two-level expression. For our purpose, a *minimal* expression is an expression with the smallest number of literals taken from the expressions with a smallest number of products.

Using the theory developed in [12], [13] and [14] we implemented an algorithm for computing the minimal canonical forms for the three different generalizations of Reed-Muller canonical form: the Galois Field-based expansion introduced by Green and Taylor [11], the Reed-Muller-Fourier form of Stanković and Moraga [13] and our generalization from [14]. We applied the algorithm to a set of encoded 4-valued benchmark functions, 3- and 4-valued adders and multipliers. The experimental results show that for the benchmark functions and multipliers Reed-Muller-Fourier form and our form have a comparable number of products on average. They have 40% less products on average than the expansion of Green and Taylor. Reed-Muller-Fourier form gives best results for adders, contrary to our form for which the adders seem to be the worst-case. However, our expansion seems to be suitable for multipliers.

The paper is organized as follows. In Section 2, the background on Reed-Muller canonical form and its generalizations is given. In Section 3, the algorithm for computing the minimal canonical forms from [11], [13] and [14] is described. In Section 4, the experimental results are shown. In the final section, some conclusions are drawn and directions for further research are proposed.

2. Reed-Muller canonical form and its generalizations

In this section we describe Reed-Muller canonical form of Boolean functions and its generalizations to multiple-valued logic.

Reed-Muller canonical form of Boolean functions was introduced in 1954 by Reed [6] and Muller [7]. It is an expression of type:

$$f(x_1,\ldots,x_n) = \sum_{i=0}^{2^n-1} c_i \cdot x_1^{i_1} \cdot x_2^{i_2} \cdot \ldots \cdot x_n^{i_n}, \qquad (1)$$

where $c_i \in \{0,1\}$ are constants, the sign $\sum$ stands for XOR and "$\cdot$" stands for AND. $(i_1 i_2 \ldots i_n)$ is the binary expansion of i with i_1 being the least significant digit, and $x_j^0 = 1$ and $x_j^1 = x_j$ for $j \in \{0,1,\ldots,n\}$. All variables in (1) are uncomplemented.

If the restriction on all the variables being uncomplemented is dropped, then the Reed-Muller canonical form extends to *fixed polarity* Reed-Muller canonical form, which is unique for a fixed polarity $k \in \{0,1,\ldots,2^n-1\}$ and is given by:

$$f(x_1,\ldots,x_n) = \sum_{i=0}^{2^n-1} c_i \cdot {}^{k_1}x_1^{i_1} \cdot {}^{k_2}x_2^{i_2} \cdot \ldots \cdot {}^{k_n}x_n^{i_n} \qquad (2)$$

where $(i_1 i_2 \ldots i_n)$ and $(k_1 k_2 \ldots k_n)$ are the binary expansions of i and k, respectively. The term ${}^{k_j}x_j^{i_j}, j \in \{0,1,\ldots,n\}$, is defined by: ${}^0x_j^1 = x_j$, ${}^1x_j^1 = x_j'$ and, ${}^{k_j}x_j^0 = 1$, for $k_j \in \{0,1\}$.

There are several ways to extend the Reed-Muller canonical form to m-valued logic. The first generalization was proposed by Cohn in 1960 [8]. It is an expression over a ring of integers modulo m, with m being a prime, namely:

$$f(x_1,\ldots,x_n) = \sum_{i=0}^{m^n-1} c_i \odot x_1^{i_1} \odot x_2^{i_2} \odot \ldots \odot x_n^{i_n}, \qquad (3)$$

where c_i are constants over a finite set of values $M \overset{df}{=} \{0,1,\ldots,m-1\}$, the sign $\sum$ stands for addition modulo m, and "$\odot$" stands for multiplication modulo m. $(i_1 i_2 \ldots i_n)$ is the m-ary expansion of i, and the term $x_j^{i_j}$ denotes the i_jth power of the variable x_j, $j \in \{0,1,\ldots,n\}$.

Cohn's generalization was extended by Pradhan [9] for the case when m is a power of a prime, and then further extended by Kodandapani and Setlur [10] to the fixed-polarity case, where the variables x_j, $j \in \{1,2,\ldots,n\}$, are represented by all literals except $\overset{k_j}{x}_j$, where $(k_n \ldots k_2 k_1)$ is the

m-ary expansion of fixed a polarity $k \in \{0, 1, \ldots, m^n - 1\}$. Recall, that the literal operator $\overset{i}{x}$, $i \in M$, is defined by

$$\overset{i}{x} \overset{df}{=} \begin{cases} m-1 & \text{if } x = i \\ 0 & \text{otherwise} \end{cases}$$

Green and Taylor [11] introduced a different extension of Cohn's form to the fixed-polarity case, where an additive transform $x_j + k_j$ is performed on each variable x_j, $j \in \{1, 2, \ldots, n\}$, according to a fixed polarity $k \in \{0, 1, \ldots, m^n - 1\}$. Harking and Moraga [12] presented an efficient algorithm for computing such an expansion for different polarities.

Stanković and Moraga [13] introduced yet another extension, called *Reed-Muller-Fourier form*, where the restriction on all the variables being uncomplemented is dropped, and the NOT operation is generalized to be any reordering of the elements of $GF(m)$, giving $(m!)^n$ different fixed polarities.

All of the above described generalizations are only applicable for the algebras with m being a prime or a power of a prime number. In [14] we introduced another generalization of the fixed polarity Reed-Muller canonical form, applicable to any positive integer m. We showed that any m-valued n-variable function can be expressed over the set of operations addition modulo m, MIN and the set of all literal operators $\overset{i}{x}$, $i \in M$, in the following fixed-polarity canonical form:

$$f(x_1, \ldots, x_n) = \sum_{i=0}^{m^n-1} c_j \cdot {}^{k_1}x_1^{i_1} \cdot {}^{k_2}x_2^{i_2} \cdot \ldots \cdot {}^{k_n}x_n^{i_n} \quad (4)$$

where the sign $\sum$ stands for addition modulo m, and "$\cdot$" stands for MIN, and the term ${}^{k_j}x_j^{i_j}$ is defined by

$${}^{k_j}x_j^{i_j} \overset{df}{=} \begin{cases} m-1 & \text{if } i_j = 0 \\ \overset{k_j \oplus i_j}{x}_j & \text{otherwise} \end{cases}$$

where x_j, $j \in \{1, 2, \ldots, n\}$, is a multiple-valued variable and $k_j, i_j \in M$ are constants.

In our generalization of the polarity, we assume that each variable x_j, $j \in \{1, 2, \ldots, n\}$, is represented by all literals except $\overset{k_j}{x}_j$, where $(k_n \ldots k_2 k_1)$ is the m-ary expansion of a polarity $k \in \{0, 1, \ldots, m^n - 1\}$. For example, if $m = 3$, polarity vector $k = (021)$ implies that in the canonical form x_1 is represented by literals $\overset{0}{x}_1$ and $\overset{2}{x}_1$, x_2 is represented by $\overset{0}{x}_2$ and $\overset{1}{x}_2$, and x_3 is represented by $\overset{1}{x}_3$ and $\overset{2}{x}_3$. A similar generalization of polarity is used in [10].

We select three fixed polarity generalizations of Reed-Muller canonical form: the form introduced by Green and Taylor [11], the Reed-Muller-Fourier form of Stanković and Moraga [13] and our generalization from [14], and compare their complexity. In the next section we describe the algorithm which we use for computing these forms.

3. Computation of the minimal canonical forms

In this section we present an algorithm for computing the minimal canonical forms for the generalizations introduced in [11], [13] and [14]. The algorithm is based on the theory developed in [12], [13], [14], and, apart from unifying the previous results, brings no novelty. The reason for reporting it here is merely to give the reader a clear picture of how the experiment was conducted, so that it can be repeated, if needed.

The pseudocode of the algorithm is shown in Figure 2. The same structure is used for computing all three canonical forms, with the difference in the transformation matrices and the operations used for the computations. For example, the transformation matrices for $n = 1$ and $m = 4$ are shown in Figure 1.

$$\begin{array}{ccc} \text{Green \& Taylor} & \text{Stanković \& Moraga} & \text{Dubrova \& Muzio} \\ \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 3 & 2 & 1 \\ 0 & 2 & 3 & 1 \end{bmatrix} & \begin{bmatrix} 3 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ 3 & 2 & 3 & 0 \\ 3 & 3 & 1 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ 3 & 0 & 1 & 0 \\ 3 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

Figure 1. The transformation matrices for $n = 1$ and $m = 4$.

The algorithm receives as an input a cube cover of the on-set of a completely specified multiple-valued multiple-output function f. It returns as an output the number of non-zero coefficients in a minimal canonical form of f. The number of non-zero coefficients corresponds to the number of products in the expression. Recall, that we define *minimal* expression as the expression with the smallest number of literals among the expressions with a smallest number of products. An *optimal* polarity is the polarity of a minimal expression.

First, the transformation matrix $TM(n)$ is generated. $TM(n)$ is a two-dimensional array of integers of size $m^n \times m^n$, where m is the radix of logic and n is the number of inputs.

The arrays P and R are two-dimensional arrays of integers of size $m^n \times k$, where k is the number of outputs of f. The polarity vector P is computed for a given polarity p by reordering the coefficients of the functions in a certain way, depending of the value of p (for more details the reader is referred to [12], [13] and [14]). R is the resulting vector of coefficients of the canonical form. It is obtained by multiplying the transformation matrix TM by the polarity vector P. For Reed-Muller-Fourier form [13] and our form [14], the computation is carried out modulo m. For the Green and Taylor form [11], the operations of multiplication and addition over $GF(m)$ are used. For a prime m these coincide with modulo m operations. For the extended fields, when m is a power of a prime, they differ.

Notice, that the algorithm processes multiple-output functions by first fixing the polarity, then computing the non-zero coefficients of the expressions for all outputs, and then evaluating the total number of products. We keep track of the products, common for several outputs, are count them just once. Another possible strategy would be to find an optimal polarity for each output function separately.

```
GeneralizedRM($F$)
input: on-set $F$ of $f$
output: number of non-zero coefficients in a minimal
canonical form of $f$

    $TM(n) \leftarrow$ Generate($n$);
    best_products = $\infty$;
    best_literals = $\infty$;
    best_polarity = 0;
    for (all polarities $p$) {
            make the polarity vector $P$ by reordering the
            coefficients of $f$;
            $R = TM(n) \cdot P$;
        products $\leftarrow$ CountProducts($R$);
        literals $\leftarrow$ CountLiterals($R$);
        if (products $\leq$ best_products) {
            if (literals $<$ best_literals) {
                best_products = products;
                best_literals = literals;
                best_polarity = $p$;
            }
        }
    }
    re-compute R and P for best_polarity;
    $TM^{-1}(n) \leftarrow$ GenerateReverse($n$);
    if ($TM^{-1}(n) \cdot R \neq P$)
        return("error");
    else
        return(best_products);
```

Figure 2. Pseudocode of the algorithm for computing minimal canonical forms.

After the optimal polarity is found, the solution is verified by performing the reverse transform $TM^{-1}(n) \cdot R$.

4. Experimental results

We implemented the algorithm shown in Figure 2 and applied it to a set of 4-valued benchmark functions, and 3- and 4-valued adders and multipliers. The m-valued adders and multipliers were obtained by a direct encoding of their functionality in m-valued logic. The 4-valued benchmark functions[1] were generated by pairing inputs *and* outputs of benchmarks and encoding them using the scheme shown in Figure 3. If a function has an odd number of inputs (or outputs), then we assume the existence of one additional input (or output) which has the don't care value for all combinations. Since our algorithm takes as an input completely specified functions only, we first run Espresso [18] to obtain covers of the original benchmarks and then apply the encoding to the resulting covers. However, a similar strategy can be used to obtain a set of incompletely specified 4-valued functions. The fragments of Espresso format of a Boolean function and the resulting 4-valued one are shown in Figure 4. Notice, that if a single value in a pair of binary values is don't care, then this pair is encoded in two different ways. A binary cube with r such pairs will be encoded in 2^r 4-valued cubes.

pair of values in the original function	resulting value(s) in the 4-valued function
00	0
01	1
10	2
11	3
--	-
-0	{0,2}
-1	{1,3}
0-	{0,1}
1-	{2,3}

Figure 3. Encoding scheme used to obtain 4-valued input 4-valued output benchmarks; "-" stands for the don't care value.

Table 1 shows the number of non-zero coefficients in the expressions computed by the algorithm in Figure 2 for the 4-valued benchmark functions. Columns 2 and 3 give the number of inputs n and the outputs k of the functions. The numbers of non-zero coefficients in the canonical forms [11], [13] and [14] are shown in the columns 4, 6 and 8, correspondently. To have a more representative comparison, we have also presented in columns 5, 7 and 9 the relative values in %, taking the Green & Taylor [11] form as ba-

[1]This set of 4-valued benchmarks is available at http://www.ele.kth.se/~elena/

Table 1. Benchmark results for 4-valued functions, showing the number of non-zero coefficients in the expressions computed by the algorithm in Figure 2.

Example function	n	k	Green & Taylor [11] abs.	Green & Taylor [11] %	Stanković & Moraga [13] abs.	Stanković & Moraga [13] %	Dubrova & Muzio [14] abs.	Dubrova & Muzio [14] %
5xp1	3	5	223	100%	154	69.06%	150	67.26%
9sym	5	1	261	100%	84	32.18%	135	51.72%
bw	3	14	118	100%	79	66.95%	96	81.36%
clip	5	3	1112	100%	673	60.52%	715	64.30%
con	4	1	57	100%	42	73.68%	23	40.35%
ex1010	5	5	2249	100%	2231	99.20%	2238	99.51%
inc	4	5	193	100%	123	63.73%	120	62.18%
misex1	4	4	99	100%	51	51.52%	28	28.28%
rd53	3	2	50	100%	25	50.00%	30	60.00%
rd73	4	2	189	100%	106	56.08%	120	63.49%
rd84	4	2	262	100%	125	47.71%	116	44.27%
sao2	5	2	657	100%	246	37.44%	89	13.55%
squar5	3	4	61	100%	38	62.30%	47	77.05%
average	4	4	425	100%	306	59.26%	301	57.95%

Table 2. Experimental results for 3- and 4-valued adders, showing the number of non-zero coefficients in the expressions computed by the algorithm in Figure 2.

m	Example function	n	k	Green & Taylor [11] abs.	Green & Taylor [11] %	Stanković & Moraga [13] abs.	Stanković & Moraga [13] %	Dubrova & Muzio [14] abs.	Dubrova & Muzio [14] %
3	adder 2-bit	4	3	21	100%	14	66.67%	32	152.38%
	adder 3-bit	6	4	75	100%	42	56.00%	165	220.00%
4	adder 2-bit	4	3	77	100%	34	44.16%	68	88.31%
	adder 3-bit	6	4	415	100%	174	41.93%	524	126.27%
	average	5	4	147	100%	66	52.19%	197	146.74%

Table 3. Experimental results for 3- and 4-valued multipliers, showing the number of non-zero coefficients in the expressions computed by the algorithm in Figure 2.

m	Example function	n	k	Green & Taylor [11] abs.	Green & Taylor [11] %	Stanković & Moraga [13] abs.	Stanković & Moraga [13] %	Dubrova & Muzio [14] abs.	Dubrova & Muzio [14] %
3	mult. 2-bit	4	4	52	100%	33	63.46%	31	59.62%
	mult. 3-bit	6	6	581	100%	369	63.51%	349	60.07%
4	mult. 2-bit	4	4	278	100%	113	40.36%	108	38.85%
	mult. 3-bit	6	6	5813	100%	2830	48.68%	2171	37.35%
	average	5	5	1681	100%	836	54.08%	665	48.97%

```
.i 9                              .i 5
.o 10                             .o 5
                                  .m 4
110000-10   1000100001            30010   20201
                                  30030   20201
                                  30011   20201
                                  30031   20201
...                               ..
10--10011   0000100100            2-212   00210
                                  2-213   00210
.e                                .e
```

Figure 4. Fragment of Espresso format of an original benchmark function and the encoded 4-valued one.

sis (100%). Tables 2 and 3 show the results for the 3- and 4-valued adders and multipliers, correspondently.

The experimental results show that for the benchmark functions Reed-Muller-Fourier form and our expansion are comparable in the number of products on average. The expansion of Green and Taylor has 40% more products on average. Reed-Muller-Fourier form gives best results for adders, contrary to our form for which the adders are hard. Our form seems particularity suitable for multipliers, with 5% reduction of the number of products over Reed-Muller-Fourier form and 51% reduction over Green and Taylor's form.

For the Green and Taylor's and our expansions, the optimal polarity is computed by an exhaustive search through all m^n possible polarities. This brute force approach makes the algorithm infeasible for the functions with $n > 6$ and $m > 4$. E.g., it takes only 4 min to compute the optimal polarity for the 3-valued 3-bit adder having 6 inputs. However, 2 days are needed to find the solution for the 3-valued 4-bit adder having 8 inputs. Using some heuristics for finding an optimal polarity would extend the capability of the algorithm, but so far no such heuristics have been found.

For Reed-Muller-Fourier form [13], the exhaustive search through all $(m!)^n$ possible polarities is out of question even for $n > 2$. Therefore, we compute m^n representatives of $(m!)^n$ possible polarities and take the best result. This, of course, means that we do not exploit the full power of Reed-Muller-Fourier form and for some functions the result we compute is not the best one.

When analyzing the results in the above tables, it should be taken into account that the products in the Reed-Muller-Fourier form and the form [11] use $GF(m)$ multiplication, while the products in our form [14] use MIN. With present technologies, MIN has a much simpler implementation than multiplication in $GF(m)$. So, the circuit realizing a MIN-based expression will occupy less area than the circuit realizing a $GF(m)$ multiplication-based expression of the same number of products and literals. This, of course, might change as technology evolves.

5. Conclusion

This paper compares the complexity of three fixed polarity generalizations of Reed-Muller canonical form: the Galois Field-based form introduced by Green and Taylor [11], the Reed-Muller-Fourier form of Stanković and Moraga [13] and our generalization over addition modulo m, minimum and the set of all literal operators [14]. Using the theory developed [12], [13] and [14] we implemented an algorithm for computing the minimal canonical forms for the corresponding generalizations and applied to a set of benchmark functions, adders and multipliers. The experimental results show that for the benchmark functions the Reed-Muller-Fourier form and our expansion give comparable results. On average, the number of products in these expansions is 40% smaller than the number of products in the expansion of Green and Taylor. Adders have most compact representations in Reed-Muller-Fourier form. Our expansion seems to be most suitable for multipliers.

In general, it is hard to make a comparison between different representations of functions. We based our comparison on the benchmarks results. An alternative would be to compare the upper bounds on the number of products in the canonical forms. Unfortunately, very few results on the upper bounds are reported for the case of $m > 2$ and none for the canonical forms introduced in [11], [13] and [14].

The main open problem remains finding a good heuristic for selecting an optimal polarity. We need to examine which functional properties can be used to guide the choice of polarity and also to investigate which search techniques are applicable to this problem.

Further work on the efficiency of computing the expressions might also incorporate representation of multiple-valued functions by Multiple-Valued Decision Diagrams (MDD) [19], and performing the basic operations of the algorithm directly on graphs. Since there is no direct correspondence between the size of the cover for the function and the size of the MDD that represents it, very large cube covers can be captured and efficiently manipulated using this representation. For the Reed-Muller-Fourier representation, some work in this direction for has been started in [20] and [21].

Acknowledgment

I am grateful to Prof. Claudio Moraga and Prof. Radomir Stanković for helpful discussions over the Reed-Muller-

Fourier form and to Dr. Dilian Gurov for carefully reading and commenting on the manuscript. I am also indebted to the anonymous referee who gave me the reference [11].

This work was supported in part by Research Grant No 240-98-101 from the Swedish Research Council for Engineering Sciences and by a fellowship from the Knut and Alice Wallenbergs foundation of Sweden.

References

[1] T. Sasao, On the optimal design of multiple-valued PLA's, *IEEE Trans. Comput.* **C-38** No. 4 (1989), 582-592.

[2] M. A-E-Barr, M. N. Hasen, New MVL-PLA structure based on current-mode CMOS technology, *Proc. 26th Int. Symp. Multiple-Valued Logic*, (1996), 98-103.

[3] X. Deng, T. Hanyu, M. Kameyama, Design and evaluation of a current-mode multiple-valued PLA based on a resonant tunneling transistor model, *IEE Proc. Circuits Devices Syst.* **141** No. 6 (1994), 445-450.

[4] T. Utsumi, N. Kamiura, Y. Hata, K. Yamato, Multiple-valued programmable logic array with universal literals, *Proc. 27th Int. Symp. Multiple-Valued Logic*, (1997), 163-169.

[5] J.C. Muzio, T.C. Wesselkamper, *Multiple-Valued Switching Theory*, Adam Hilger Ltd. Bristol and Boston, 1986.

[6] I. S. Reed, A class of multiple-error-correcting codes and their decoding scheme, *IRE Trans. on Inform. Theory* **4** (1954), 38-42.

[7] D. E. Muller, Application of Boolean algebra to switching circuit design and to error detection, *IRE Trans. Electron. Comput.* **EC-3** (1954), 6-12.

[8] M. Cohn, *Switching Function Canonical Form over Integer Fields*, Ph.D thesis, Harvard University, Cambridge, Mass., Dec. 1960.

[9] D. K. Pradhan, A multi-valued algebra based on finite fields, *Proc. 1974 International Symp. on MVL*, (1974) 95-112.

[10] K. L. Kodandapani, R. V. Setur, Multi-valued algebraic generalization of Reed-Muller canonical forms, *Proc. 1974 International Symp. on MVL* (1974), 505-544.

[11] D. H. Green, I. S. Taylor, Modular representation of multiple-valued logic systems *Proc. of the IEE*, **121**, (1974), 424-429.

[12] B. Harking, C. Moraga, Efficient derivation of Reed-Muller expansions in multiple-valued logic systems, *Proc. 22nd International Symp. on MVL* (1992), 436-441.

[13] R. S. Stanković, C. Moraga, Reed-Muller-Fourier versus Galois Field representations of four-valued logic functions, *Proc. 28th International Symp. on MVL* (1998), 186-191.

[14] E. V. Dubrova, J. C. Muzio, Generalized Reed-Muller canonical form for a multiple-valued algebra, *Multiple-Valued Logic, An International Journal* **1** (1996), 65-84.

[15] L. Zhijian, J. Hong, A CMOS current-mode high speed fuzzy logic microprocessor for a real-time expert system, *Proc. 20th International Symp. on MVL* (1990), 394-400.

[16] Z. Zilic, Z. Vranesic, Current-mode CMOS Galois field circuits, *Proc. 23rd International Symp. on MVL* (1993), 245-250.

[17] G. De Micheli, *Synthesis and optimization of digital circuits*, McGraw-Hill, 1994.

[18] R.K. Brayton, G. Hachtel, C. McMullen, A. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*, Kluwer Academic Publisher, 1984.

[19] D. M. Miller, Multiple-valued logic design tools, *Proc. 22rd International Symp. on MVL* (1993), 2-11.

[20] R. S. Stanković, C. Moraga, T. Sasao, Calculation of Reed-Muller-Fourier coefficient of multiple-valued functions through multiple-place decision diagrams, *Proc. 24th International Symp. on MVL* (1994), 82-88.

[21] R. S. Stanković, Functional decision diagrams for multiple-valued functions, *Proc. 25th International Symp. on MVL* (1995), 284-289.