Optimization of Arithmetic Expressions Using the Dual Polarity Property

Dragan Janković[1] , Radomir S. Stanković[1], and Claudio Moraga[2]

[1] Faculty of Electronic Engineering, University of Niš, 18000 Niš, Serbia and Montenegro
Email: {gaga,rstankovic}@elfak.ni.ac.yu
[2] University of Dortmund, Germany and
Department of Artificial Intelligence, Technical University of Madrid, Spain.
Email: claudio@moraga.de

Abstract. In this paper we propose a method for optimization of fixed polarity arithmetic expressions (FPAEs) based on dual polarity. The method exploits a simple relationship between two FPAEs for dual polarities. It starts from the zero polarity FPAE of the given function and calculates all FPAEs using dual polarity route. Conversion from one FPAE to another one is carried out using one-bit checking i.e. by using a very simple rule. Therefore, the proposed method is efficient. Experimental results show this.

1 Introduction

Arithmetic expressions are an alternative approach to description of logic circuits sharing useful properties of Reed-Muller expressions and permitting at the same time simplified representations of multi-output functions [8]. In many applications where Boolean functions need to be analyzed, arithmetic expressions provide a better insight into related problems and offer efficient solutions [9]. Examples are satisfiability, tautology, equivalence checking, etc. [1, 3-6, 10-13, 16-17]. Applications of arithmetic expressions in logic design dates back to early fifties [14], and a renewed interest in this subject is due to ever increasing challenges of probabilistic verification of logic circuits and requests for compact decision diagram and related representations, where other methods do not provide acceptable solutions or cannot be used for large space and time requirements [1, 15].

Arithmetic expressions are closely related to Reed-Muller expressions since are defined in terms of the same basis, however, with variables and function values interpreted as integers 0 and 1 instead of logic values. In this way, arithmetic expressions can be considered as integer counterparts of Reed-Muller expressions. Due to this interpretation, many efficient algorithms for Reed-Muller expressions as well as optimization methods [13] can be extended to arithmetic expressions. In particular, as for Reed-Muller expressions, the optimization of arithmetic expressions in the number of non-zero coefficients count can be performed by choosing literals of different polarity for integer counterparts of switching variables in terms of which arithmetic expressions are defined.

In this paper we propose a method for optimization of fixed polarity arithmetic expressions (FPAEs) based on a dual polarity. We derive relationships between two FPAEs for Boolean functions for dual polarities. Based on these relationships, a new method for FPAEs optimization is proposed. The algorithm starts from the zero polarity FPAE of the given functions and calculates all FPAEs using a route in which each two successive polarities are dual. This route is called the dual polarity route for the generation of which a corresponding procedure is defined.

The algorithm proposed is an exhaustive-search algorithm, but conversion from one FPAE to another is carried out by using one–bit checking. Due to that, and the simplicity of the related processing this algorithm appears to be efficient as confirmed by experimental results. It is important to notice that the algorithm proposed shows high possibilities for parallelization.

2 Basic Definition

Definition 1. Any n-variable switching function f given by the truth-vector $\mathbf{F} = \left[f_0, \ldots, f_{2^n-1}\right]^T$ can be represented by the positive polarity arithmetic expression (PPAE) defined as

$$f(x_1, \ldots, x_n) = \mathbf{X}(n)\mathbf{A}(n)\mathbf{F}$$

where

$$\mathbf{X}(n) = \bigotimes_{i=1}^{n} \begin{bmatrix} 1 & x_i \end{bmatrix},$$

$$\mathbf{A}(n) = \bigotimes_{i=1}^{n} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix},$$

where $\otimes$ denotes the Kronecker product, and addition and multiplication are arithmetic operations. $\mathbf{A}(n)$ represents the arithmetic transform matrix of order n.

If each variable can appear as complemented or uncomplemented, but not both, the related expressions are fixed-polarity arithmetic expressions (FPAEs) given as

$$f(x_1, \ldots, x_n) = \mathbf{X}_H(n)\mathbf{A}_H(n)\mathbf{F}$$

where

$$\mathbf{X}_H(n) = \bigotimes_{i=1}^{n} \begin{bmatrix} 1 & x_i^{h_i} \end{bmatrix}, \qquad x_i^{h_i} = \begin{cases} x_i, & h_i = 0, \\ \overline{x}_i, & h_i = 1, \end{cases}$$

and

$$\mathbf{A}_H(n) = \bigotimes_{i=1}^{n} \mathbf{A}^{h_i}(1)$$

$$\mathbf{A}^{h_i}(1) = \begin{cases} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}, & h_i = 0, \\ \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix}, & h_i = 1. \end{cases}$$

Therefore, FPAEs are uniquely characterized by the polarity vectors $\mathbf{H} = [h_1, \dots, h_n]^T$, $h_i \in \{0,1\}$, where $h_i = 1$ shows that the *i*-th variable is complemented and written as $\bar{x}_i$.

An FPAE can be given by the FPAE spectrum $\mathbf{A}_f^H$ calculated as

$$\mathbf{A}_f^H = \mathbf{A}_H(n)\mathbf{F}.$$

Example 1. The FPAE of a two-variable Boolean function *f*, given by the truth-vector $\mathbf{F} = [0,1,1,1]^T$, for a polarity vector $\mathbf{H} = (0,1)$ is given by

$$f(x_1, x_2) = 1 \cdot 1 - 1 \cdot \bar{x}_2 + 0 \cdot x_1 + 1 \cdot x_1\bar{x}_2 = 1 - \bar{x}_2 + x_1\bar{x}_2$$

The corresponding FPAE spectrum is given by

$$\mathbf{R}_f^H = [1,-1,0,1] \ .$$

Therefore, this function f can be represented by the set of terms $\{00 \sim (1), 01 \sim (-1), 11 \sim (1)\}$, where the first two digits express the binary encoded position of the spectral coefficient, and the digit in parenthesis, its value.

3 Dual Polarity

Each FPAE is characterized by its polarity vector. Two polarity vectors are dual if they differ in only one bit.

Definition 2: $\mathbf{H}' = (h'_1, \dots, h'_{i-1}, h'_i, h'_{i+1}, \dots, h'_n)$ is dual polarity of polarity $\mathbf{H} = (h_1, \dots, h_{i-1}, h_i, h_{i+1}, \dots, h_n)$ iff $h'_j = h_j$, $j \neq i$ and $h'_i \neq h_i$.

Example 2: Dual polarities for polarity $\mathbf{H} = (1,0)$ are the polarities (0,0), and (1,1).

The number of polarity vectors, which characterize all possible arithmetic expressions for an *n*-variable Boolean function is 2^n. It is possible to order all 2^n polarities in such a way that each two successive polarities are dual polarities. This order is denoted as the “dual polarity route”. Traversing the two-valued *n*-dimensional hypercube can generate one of several possible dual polarity routes. It becomes apparent that a polarity route generates a Gray code.

Example 3: A dual polarity route generated by traversing a two-valued 3-dimensional hypercube is given by
(000)—(001)—(011)—(010)—(110)—(111)—(101)—(100).

A dual polarity route can be constructed by using the recursive procedure **route**(*level*, *direction*) given in Fig. 1, called as *route*(0,0).

```
void route (int level, int direction)
{
if (level == no_variable)
{
-- out new polarity vector h;
return;
}
          if (direction == 0)
          {
           h[level] = 0;
           route (level + 1, 0);
           h[level] = 1;
           route (level + 1, 1);
          }
          else
          {
           h[level] = 1;
           route (level + 1, 0);
           h[level] = 0;
           route (level + 1, 1);
          }
}
```

Fig. 1. Procedure *route*

4 Method for Calculation of Arithmetic Expression

Let $m = m_1^{i-1} m_i m_{i+1}^n$ be the compact representation of a term in the arithmetic expression for a given function *f* for the polarity $h = (h_1 \cdots h_{i-1} h_i h_{i+1} \cdots h_n)$. Term *m* produces new terms in the arithmetic expression of the function *f* for the dual polarity $h' = (h'_1 \cdots h'_{i-1}\, h'_i\, h'_{i+1} \cdots h'_n)$ depending on the value m_i. Term *m* generates a new term *m*' if $m_i = 1$. A new term at the *i*-th position has the value 0 and contribution for this term is the same as for the term *m*. Also, the contribution of the term *m* changes the sign.

Table 1 shows the used processing rule, whose simplicity ensures efficiency of the method. After processing all terms by using this rule, a procedure for deleting terms whose contribution is equal to 0 starts.

h_i	h'_i	New terms
0	1	if m_i =1 then a) generate $m' = m_1^{i-1} 0 m_{i+1}^n$ and $v(m') = v(m)$
1	0	b) set $v(m)$ to $-v(m)$

Table 1. Processing rule

Example 4: For a three-variable Boolean function f given by the truth vector $\mathbf{F} = [0,1,1,0,0,1,1,1]^T$, the (010)-polarity arithmetic expression is given by

$$f = 1 - x_3 - \bar{x}_2 + 2\bar{x}_2 x_3 + x_1 x_3 - x_1 \bar{x}_2 x_3$$

i.e., the arithmetic spectrum for f is $[1,-1,-1,2,0,1,0,-1]^T$. The dual polarities and the corresponding dual polarity arithmetic expressions are given in Table 2.

Polarity	Spectrum
011	[0,1,1,-2,1,-1,-1,1]
000	[0,1,1,-2,0,0,0,1]
110	[1,0,-1,1,0,-1,0,1]

Table 2. Spectrum of f for dual polarities of (010)

Procedures for calculation of these dual polarity arithmetic expressions are shown in Tables 3, 4, and 5. Note that canceled terms are underlined while exchanged terms are double underlined.

5 Optimization Algorithm

Example 4 shows that it is possible to calculate all possible arithmetic expressions using the proposed method for transforming fixed polarity arithmetic expressions into dual polarity arithmetic expression along the route without repetitive calculations. Therefore, we can perform the optimization of arithmetic expressions by using the exhaustive-search algorithm shown in Fig. 2.

6 Experimental Results

In this section, we present some experimental results estimating features and efficiency of the proposed algorithm for minimization of arithmetic expressions. We developed a program in C for determination of optimal arithmetic expression for arbitrary Boolean functions represented by minterms. The experiments were carried out on a 600 MHz Celeron PC with 128 Mbytes of main memory and all runtimes are given in CPU seconds. Table 4 compares the runtimes for optimization of arithmetic expressions by the Tabular technique in [7] (columns ATT) with the algorithm proposed in this paper (columns Dual). We consider the simple functions taking the value

1 at the first three minterms (0,1,2), randomly generated functions with 25% of all possible minterms, and randomly generated functions with 75% of all possible minterms, where the number of variables n ranged from 7 to 12. Columns %d show the ratio (Dual – ATT)/ATT where ATT and Dual refer to the method in [7] and the proposed algorithm, respectively.

It can be concluded that the number of minterms strongly influences the runtime of the proposed algorithm, but the algorithm is faster than ATT.

1. For an n-variable Boolean function f, calculate the positive polarity arithmetic expression (for example by using the algorithms in [2,7]).
2. List all the terms for a positive polarity arithmetic expression. Set $C_{\min}$ = *the number of non-zero coefficients in positive polarity arithmetic expression.*
3. Determine the next polarity h', of the arithmetic expansion according to the recursive route.
4. Obtain the arithmetic expansion of polarity h' based on the proposed rule. Calculate the total number of non-zero coefficients $C'_{\min}$.

   If $C'_{\min} < C_{\min}$ then $C_{\min} = C'_{\min}$.
5. Stop if all polarities have been treated. Otherwise go to the step 3.

Fig. 2. Optimization algorithm

7 Concluding Remarks

We have introduced the notion of dual polarities for arithmetic expressions for Boolean functions and present a method for conversion of arithmetic expressions from one polarity to another. Based on this method, we determine an algorithm for calculation of all arithmetic expressions for a given function f. Calculation is performed by starting from the positive polarity arithmetic expression which is calculated by using tabular method for calculation of fixed polarity arithmetic transform [1] of functions represented by the truth-vector. All arithmetic expressions are calculated along the route that provides calculation of each fixed polarity arithmetic expressions exactly once.

The proposed method for transformation of FPAE from one to another dual polarity is simple. Therefore, our exhaustive-search arithmetic expression optimization method is efficient. Experimental results confirm this.

Future work will be on extension of the proposed method and the algorithm to various other polynomial expressions for multiple-valued functions.

It is important to notice that the method proposed has high possibilities for parallelization. Each of processors performs the method along a piece of dual polarity route. In Table 7 are given these pieces of route for the optimization of arithmetic expression of a three-variable Boolean function if the number of processors is 1,2 and 3.

polarity (010)	new terms	polarity (011)
$\underline{000 \sim (1)}$		001 ~ (1)
$\underline{\underline{001 \sim (-1)}}$	$\underline{000 \sim (-1)}$	010 ~ (1)
	001 ~ (1)	011 ~ (-2)
010 ~ (-1)		100 ~ (1)
$\underline{\underline{011 \sim (2)}}$	010 ~ (2)	101 ~ (-1)
	011 ~ (-2)	110 ~ (-1)
$\underline{\underline{101 \sim (1)}}$	100 ~ (1)	111 ~ (1)
	101 ~ (-1)	
$\underline{\underline{111 \sim (-1)}}$	110 ~ (-1)	
	111 ~ (1)	

Table 3. Polarity (010) to (011)

polarity (010)	new terms	polarity (000)
$\underline{000 \sim (1)}$		001 ~ (1)
001 ~ (-1)		010 ~ (1)
$\underline{010 \sim (-1)}$	$\underline{000 \sim (-1)}$	011 ~ (-2)
	010 ~ (1)	111 ~ (1)
$\underline{\underline{011 \sim (2)}}$	001 ~ (2)	
	011 ~ (-2)	
$\underline{101 \sim (1)}$		
$\underline{\underline{111 \sim (-1)}}$	$\underline{101 \sim (-1)}$	
	111 ~ (1)	

Table 4. Polarity (010) to (000)

polarity (010)	new terms	polarity (110)
000 ~ (1)		000 ~ (1)
$\underline{001 \sim (-1)}$		010 ~ (-1)
010 ~ (-1)		011 ~ (1)
011 ~ (2)		101 ~ (-1)
$\underline{\underline{101 \sim (1)}}$	$\underline{001 \sim (1)}$	111 ~ (1)
	101 ~ (-1)	
$\underline{\underline{111 \sim (-1)}}$	011 ~ (-1)	
	111 ~ (1)	

Table 5. Polarity (010) to (110)

n	(012)			25%			75%		
	ATT	Dual	%d	ATT	Dual	%d	ATT	Dual	%d
7	<0,01	<0,01	-	0,03	<0,01	-66,67	0,09	<0,01	-88,89
8	0,02	<0,01	-50	0,16	<0,01	-93,75	0,52	0,01	-98,08
9	0,08	<0,01	-87,5	1,04	<0,01	-99,04	3,01	0,01	-99,67
10	0,3	0,01	-96,67	6,12	0,03	-99,51	17,91	0,03	-99,83
11	1,14	0,04	-96,49	36,66	0,07	-99,81	108,29	0,11	-99,90
12	4,62	0,16	-96,53	222,22	0,30	-99,86	222,41	0,35	-99,84

Table 6. Experimental results

Number of processors	Polarities
1	P1: {(000)-(001)-(011)-(010)-(110)-(111)-(101)-(100)}

2	P1: {(000)-(001)-(011)-(010)-(110)}
	P2: {(000)-(100)-(101)-(111)}

3	P1: {(000)-(001)-(011)}
	P2: {(000)-(010)-(110)}
	P3: {(000)-(100)-(101)-(111)}

Table 7. Parallel execution

References

1. Dubrova E., Krenz R., Kuehlmann A., "Circuit-based Evaluation of the Arithmetic Transform of Boolean Functions", *Notes of International Workshop on Logic Synthesis,* New Orleans, LO, (2002), 321-327.
2. Falkowski B.J. Schaefer I., Perkowski M.A., "Generation of Adding and Arithmetic Multi-polarity Transforms for Incompletely Specified Boolean Functions", *International Journal of Electronics*, 73(2):321-331 (1992)
3. Falkowski B.J., Shmerko, V.P., Yanushkevich, S.N., "Arithmetical Logic - its Status and Achievements", *Proceedings International Conference on Applications of Computer Systems*, Szczecin, Poland, (1997), 208-223.
4. Falkowski B.J., "A Note on the Polynomial Form of Boolean Functions and Related Topics", *IEEE Transactions on Computers*, 48(8):860-864, (1999)
5. Heidtmann K.D., "Arithmetic Spectrum Applied to Fault Detection for Combinatorial Networks", *IEEE Transactions on Computers*, 40(3):320-324 (1991)
6. Heidtmann K.D., "Arithmetic Spectra Applied to Stuck-at-fault Detection for Combinatorial Networks", *Proceedings 2nd Technical Workshop New Directions in IC Testing*, Winnipeg, Canada, (1987) 4.1-4.13
7. Janković D., "Tabular Technique for the Fixed Polarity Arithmetic Transform Calculation" (in Serbian), submitted at *XLVII Conference ETRAN*, 2003, Herceg-Novi, Serbia and Montenegro (2003)
8. Malyugin V.D., *Paralleled Calculations by Means of Arithmetic Polynomials,* Physical and Mathematical Publishing Company, Russian Academy of Sciences, Moscow, (1997)
9. Malyugin V.D., Kukharev G.A., Shmerko V.P., ``Transforms of Polynomial Forms of Boolean Functions", Institute of Control Sciences, Moscow, (1986), 1-48.
10. Muzio J.C., ``Stuck Fault Sensitivity of Reed-Muller and Arithmetic Coefficients", C. Moraga, (Ed.), *Theory and Applications of Spectral Techniques*, Dortmund, (1989), 36-45.
11. Rahardja S., Falkowski B.J., "Application of Linearly Independent Arithmetic Transform in Testing of Digital Circuits", *Electronic Letters*, 35(5):363-364, (1999)
12. Rahardja S., Falkowski B.J., "Fast Linearly Independent Arithmetic Expansion", *IEEE Transactions on Computers*, 48(9):991-999, (1999)
13. Sasao T., *Switching Theory for Logic Synthesis*, Kluwer, (1999)
14. Stanković R.S., Sasao T., "A Discussion on the History of Research in Arithmetic and Reed-Muller Expressions", *IEEE Transactions on CAD*, 20(9):1177-1179, (2001)

15. Stanković R.S., "Word-level Ternary Decision Diagrams and Arithmetic Expressions", *Proceedings 5th International Workshop on Applications of Reed-Muller Expansion in Circuit Design*, Starkville, Mississippi, (2001), 34-50.
16. Yanushkevich S.N., *Logic Differential Calculus in Multi-Valued Logic Design*, Tech University of Szczecin Academic Publishers, Poland, 1998.
17. Yanushkevich S.N., Falkowski B.J., Shmerko V.P., "Spectral Linear Arithmetic Approach for Multiple-valued Functions - Overview of Applications in CAD of Circuit Design", *Proceedings 3rd International Conerence. on Information, Communications and Signal Processing (ICICS)*, CD publication, Singapore, (2001)