BDD-based spectral approach for Reed–Muller circuit realisation

M.A.Thornton
V.S.S.Nair

Indexing terms: Spectral coefficients, Reed–Muller circuits, Binary decision diagram

Abstract: In the past, the use of spectral coefficients for the realisation of Reed–Muller circuits led to computational difficulties for functions of even moderate size, due to large memory requirements. A new approach is presented that overcomes these difficulties by allowing the function to be represented using a binary decision diagram, thus reducing the storage requirements. In addition, the computational requirements are also reduced since an efficient method for computing the spectral coefficients is employed. Furthermore, the approach is illustrated with examples.

1 Introduction

The well-known properties of the generalised Reed–Muller (GRM) canonical form make it an attractive candidate for logic circuit implementations. It has been shown that exclusive-OR sum of product (ESOP) forms generally require fewer logic gates than the more traditional sum-of-products forms [1]. For symmetric functions it has been proven that the ESOP circuit form will never require more product terms than the number needed for the corresponding SOP realisation [2]. Many FPGAs are now being manufactured that utilise the XOR gate as a basic cell component. This requires ESOP implementations to be considered when an FPGA implementation is desired. Also, the ESOP form allows all single stuck-at faults to be tested with a minimal number of test vectors [3]. Finally, recent developments in layout technology have reduced the layout area required for XOR gates so that it is comparable with that of the other basic logic gates [4].

The Reed–Muller (RM) transformation matrix can be used to realise a circuit given its output vector. Although this deterministic technique is very attractive for computer-aided design (CAD) applications, it is not a practical method for reasonably sized circuits due to the large storage requirements that spectral transformation methods demand. In this paper we describe a means for the realisation of a RM circuit by using its binary decision diagram (BDD) representation instead of a truth table description. This methodology uses the principles of RM transformations coupled with a novel method for obtaining the coefficients.

Recently we developed an efficient methodology for the computation of the Walsh coefficients using output probability expressions (OPEs) [5]. This methodology was not directly applicable to the RM spectrum since it does not use the required modulo-2 arithmetic. The results presented here show that all calculations may be performed by using real-valued arithmetic and then applying an isomorphic transformation afterwards. This result allows the recently discovered efficient method for the computation of the Walsh spectrum to be applied to the RM spectra as well.

2 Mathematical formulation

This Section provides the development of two important points. First, we provide the mathematical properties that allow the RM spectra to be computed using real arithmetic instead of the modulo-2 arithmetic that is used when the definition is applied. Second, the relationship between the coefficients as computed using real arithmetic and the output probability of a logic function are developed. This relationship will lead to the formation of a method for computing a RM coefficient using efficient BDD-based algorithms.

2.1 Isomorphic relation of GF(2) and the real field

This Section will prove that an isomorphic mapping function exists between the Galois field containing two numbers, GF(2), and the real field. Furthermore, it is proven that the isomorphic function maps into equivalence classes containing single elements and hence may be used to determine a unique value. Since the RM transform is defined over GF(2), the existence of an isomorphism that provides a unique mapping is crucial because it will allow the transform calculations to be performed in the real field. To develop the isomorphic relation, the concept of an algebraic ring and, hence, a ring morphism is utilised.

A Boolean ring satisfies all of the properties of a general ring along with the idempotence property [6]. The familiar Boolean algebra can be formulated with respect to the Boolean ring, $\Re$, defined as follows:

$$\Re : \{0, 1, \oplus, \cdot\} \quad (1)$$

where 0 and 1 denote the logic values of zero and one, the addition operator, $\oplus$, denotes the binary XOR operation, and the multiplicative operator, $\cdot$, denotes

IEE Proceedings online no. 19960067

Paper first received 6th January 1995 and in final revised form 4th August 1995

M.A. Thornton is with the Department of Computer Systems Engineering, University of Arkansas, Fayetville, AR 72701, USA

V.S.S. Nair is with the Department of Computer Science & Engineering, Southern Methodist University, Dallas, TX 75275, USA

the binary AND function. Clearly, the set, $\Re$, forms a Boolean ring since all of the properties of a general ring are satisfied [7] along with the idempotence property. Next, consider an alternative set, $\Re'$. This set contains the following elements:

$$\Re' : \{\emptyset, I, +, \times\} \tag{2}$$

where

$$\emptyset \equiv \{0, \pm 2, \pm 4, \pm 6, \ldots, \pm 2j, \ldots\} \tag{3}$$

and

$$I \equiv \{\pm 1, \pm 3, \pm 5, \ldots, \pm(2i+1), \ldots\} \tag{4}$$

Thus, $\emptyset$ and I are the sets of all even integers (including 0) and all odd integers (excluding 0), respectively. The two operators in the set, $\Re'$, are the additive operator, +, and the multiplicative operator, $\times$. These operators perform real addition and real multiplication, respectively. $\Re'$ also satisfies the properties of a Boolean ring [6].

The following lemma establishes the relationship between the two rings $\Re$ and $\Re'$:

Lemma 1: The function, $f(x) = x(\text{mod}2)$, forms a ring morphism from $\Re'$ to R.

Proof: To prove this lemma, we demonstrate that the following relationships exist:

$$f(a+b) = f(a) \oplus f(b) \tag{5}$$

$$f(a \times b) = f(a) \cdot f(b) \tag{6}$$

for the following three exhaustive cases:

Case 1: $(a, b) \in \emptyset$

Case 2: $(a, b) \in I$

Case 3: $a \in \emptyset, b \in I$

Each case is easily seen to be satisfied by observing that the following elementary properties of number theory hold:

Case 1 is a statement that the sum and product of two even integers always yield values that are also even.

Case 2 is a statement that the sum of two odd integers is always even and the product of two odd numbers is always odd.

Case 3 is a statement that the sum of an even integer and an odd integer always results in an odd value while their product is always even. □

Next, the RM form is presented with a 3-variable example as motivation for the following mathematical results. The generalised RM formulation for a three-variable function may be expressed as

$$F(x) = r_0 \oplus r_1\dot{x}_1 \oplus r_2\dot{x}_2 \oplus r_3\dot{x}_3 \oplus r_4\dot{x}_1\dot{x}_2 \oplus r_5\dot{x}_1\dot{x}_3 \oplus r_6\dot{x}_2\dot{x}_3 \oplus r_7\dot{x}_1\dot{x}_2\dot{x}_3 \tag{7}$$

where the dotted variables represent function inputs that are either complemented or not complemented in all occurrences. The r_i terms are Boolean constants that have value '1' or '0' and are members of the ring, $\Re$. In fact, the r_i terms are the RM spectral coefficients of the GRM formulation of $F(x)$. Eqn. 7 contains operations and elements from the Boolean ring, $\Re$ (the omission of a binary operator between two consecutive variables in eqn. 7 implies that the operation denoted by · occurs). Eqn. 7 is rewritten in the following form:

$$F(x) = r_0g_0 \oplus r_1g_1 \oplus r_2g_2 \oplus r_3g_3 \oplus r_4g_4 \oplus r_5g_5 \oplus r_6g_6 \oplus r_7g_7 \tag{8}$$

Where the r_i are Boolean constants as described above, and each g_i is the AND (modulo-2 product) of a subset of the function's literals. Each particular realisation of the GRM form is then a specified set of r_i values that will be referred to as the vector, **R**. The set of functions, g_i, will be referred to as the 'constituent functions' of $f(x)$. It is convenient to express the relationship in eqn. 8 in an alternative vector matrix form as:

$$\mathbf{GR} = \mathbf{F} \tag{9}$$

where the elements of the vector, **F**, and the constituent function matrix, **G**, are known (or are easily computed), and the elements of the **R** vector specify the spectrum of a GRM canonical form. The column vectors formed from all possible outputs of the constituent functions, g_i, are concatenated to form the coefficient matrix, **G**, and the corresponding function outputs are concatenated to form the function vector, **F**.

There are 2^n possible versions of eqn. 7 since all possible combinations of complemented and uncomplemented variables may be used. A convenient way to classify the various forms of GRM expressions is to use the concept of polarity [8]. Typically, the polarity number is obtained by assigning a '1' to all uncomplemented primary inputs and a '0' to all complemented primary inputs. These bits are then ordered according to descending subscripts of the corresponding primary inputs. The resulting binary number is then the polarity number.

For illustrative purposes, two forms of the GRM expressions will be considered for eqn. 7 and the two transformation matrices will be derived. It should be noted that any arbitrary polarity matrix could be derived and the details for such derivations are given in [9]. We will consider the polarity-0 and polarity-2^n matrices denoted by **G** and **G'**, respectively. The matrix, **G**, is used when all $\dot{x}_i = x_i$ and **G'** is used when all $\dot{x}_i = x_i'$ (where ' indicates the application of the complement function to a Boolean variable). In the following, we define the coefficient matrix, **G**.

Definition 1: The matrix, **G**, is a concatenation of the output column vectors of the constituent functions, g_i, with no inverted inputs.

Consider the smallest possible **G** matrix, denoted as $\mathbf{G}_2$, where the subscript indicates the number of function inputs.

$$\mathbf{G}_2 = \begin{array}{c} \begin{matrix} g_0 & g_1 & g_2 & g_3 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \end{array} \tag{10}$$

The constituent functions for $\mathbf{G}_2$ are defined as:

$$g_0 = 1$$
$$g_1 = x_0$$
$$g_2 = x_1$$
$$g_3 = x_0 \cdot x_1$$

Likewise, the matrix $\mathbf{G}_2'$ is defined as:

$$\mathbf{G}_2' = \begin{array}{c} \begin{matrix} g_3' & g_2' & g_1' & g_0 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \tag{11}$$

The constituent functions for $\mathbf{G}_2'$ are defined as

$$g_0 = 1$$
$$g_1' = x_0'$$
$$g_2' = x_1'$$
$$g_3' = x_0' \cdot x_1'$$

The following theorem shows the relationship between the operations in the ring $\Re'$ and those in $\Re$. This is accomplished by establishing that the solution vector, **R**, always contains components that are members from the field of positive and negative integers, Z. Where Z is defined as

$$Z = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\} \tag{12}$$

In essence, it is shown that the solution of eqn. 9 is performed over the Boolean ring, $\Re'$.

Theorem 1: Given any Boolean function, F, and a binary vector, $\mathbf{B}^T = [b_1, b_2, b_3 \ldots, b_n]$ (where T denotes the transpose) such that $F = b_1 g_1 \oplus b_2 g_2, \oplus \ldots \oplus b_n g_n$, then there exists a vector, **R** with each $r_i \in Z$, such that **GR** = **F** and **R** (mod2) = **B**, where **F** is the binary vector formed from the output column of the truth table of the function, F.

Proof: It is easily seen that **R** has the property that each $r_i \in Z$ since all the transformation matrices $\mathbf{G}_n$ and $\mathbf{G}'_n$ are triangular with diagonals consisting of all 1s. Hence, the following equations hold:

$$\det(\mathbf{G}_n) = \det(\mathbf{G}'_n) = 1 \tag{13}$$

It is desired to solve the equation

$$\mathbf{G}^*\mathbf{R} = \mathbf{F} \tag{14}$$

where $\mathbf{G}^*$ represents either $\mathbf{G}_n$, or $\mathbf{G}'_n$. The solution to this equation is well known and may be expressed in terms of the cofactors of $\mathbf{G}^*$ as

$$\begin{aligned}\mathbf{R} &= \mathbf{G}^{*-1}\mathbf{F} \\ &= [\det(\mathbf{G}^*)]^{-1}[\text{cofactor}\,(\mathbf{G}^*)]\mathbf{F} \\ &= [1][\text{cofactor}\,(\mathbf{G}^*)]\mathbf{F}\end{aligned}$$

Since the cofactor matrices are formed with no division operations, the matrix, [cofactor($\mathbf{G}^*$)], contains only integers. Since the matrix, $\mathbf{G}^*$, always has a determinant of 1, the solution vector, **R**, is formed as a vector-matrix multiplication of the integer matrix, [cofactor($\mathbf{G}^*$)], and the integer vector **F**. Therefore the solution vector, **R**, always contains integer components.

From lemma 1, the function, $f(x) = x$(mod2) forms a ring morphism between $\Re'$ and $\Re$. Hence, the application of $f(x)$ to each integer component in the vector, **R**, isomorphically maps **R** to **B**. □

Although it is well known that the GRM forms are unique, the result of theorem 1 allows this fact to be proven succinctly and trivially in the following corollary.

Corollary 1: Every Boolean function has a unique form of the GRM expansion.

Proof: From the definition given above, it is apparent that all $\mathbf{G}_n$ and $\mathbf{G}'_n$ matrices are triangular. From definition 1 it was noted that all $g_{ii} = g'_{ii} = 1$. Hence the determinant of the coefficient matrices is 1, guaranteeing a unique matrix inverse exists. Since **R** and **B** from theorem 1 are isomorphic and **R** is unique due to the existence of a unique inverse of **G** and **G'**, then **B** also is unique. □

This section has presented the mathematical justification for the synthesis technique for the complement-free RM forms. These results may easily be extended to any of the generalised Reed–Muller (GRM) expansions with any arbitrary polarity number. The following section will provide a numerical example and a discussion of implementation issues.

2.2 Relationship of output probabilities and the RM spectra

The output probability of a circuit is the probability that the output of the circuit is a logic 1 given the probabilities that each of the inputs are at a logic 1 value. In the work of [10] OPEs were used to evaluate the effectiveness of random testing. Two methods were developed for computing the OPE in algebraic form in [10]. One method used the Boolean logic equation and the other used logic diagrams. In [5] we noted that for all possible input combinations, each input is equally likely to be 0 or 1, and thus if 0.5 is used as the probability value for all inputs, the resulting evaluation of the OPE will yield the overall percentage of times that the function output is at logic 1.

Following this observation, we presented a methodology for computing this quantity directly from a BDD representation of a logic function without resorting to using symbolic algebraic manipulations to form the actual OPE. This technique is called the probability assignment algorithm (PAA), and is summarised as follows: *Probability assignment algorithm*

1: Assign probability = 1 for the input node.

2: If the probability of node $j = P_j$, assign a probability of $(1/2)P_j$ to each of the outgoing arcs from j.

3: The probability, P_k, of node k is the sum of the probabilities of the incoming arcs.

In the development given in the preceding subsection, we have proven that a RM spectral coefficient may be obtained by first computing a value in $\Re'$ (denoted as r' henceforth) and then mapping it to $\Re$. Unfortunately, the formation and solution of the matrix equation to compute the values in $\Re'$ has a complexity similar to computing the coefficient directly using GF(2) arithmetic. However, the r' value can be computed without resorting to solving a matrix equation by exploiting its relation to the output probability of a circuit. Before this result is proven, the following definition is helpful:

Definition 2: A Boolean function, $f \cdot g_c$, which is composed as the AND of a function to be transformed, f, and a constituent function, g_c, is termed the composition function and is denoted by f_{comp}.

Definition 2 allows lemma 2 to be easily stated and proven.

Lemma 2: The real value, r', is directly proportional to the output probability of a Boolean function.

Proof: As noted in the previous subsection, r', can be viewed as the inner product of the output vector of the function to be transformed, f, and some constituent function, g_c. Since each element in these two output vectors is either 1 or 0, each partial product of the inner product is also 1 or 0. In fact, the partial product value of 1 only occurs when both f and g_c produce an output of 1 for a common given set of input values. Thus, r' is the total number of times the composition function, $f_{comp} = f \cdot g_c$, produces an output of 1. If the output probability of f_{comp} is known (denoted by

$\wp\{f_{comp}\}$) then r' is easily computed as shown in eqn. 15.

$$r' = 2^n \times \wp\{f_{comp}\} \tag{15}$$

Therefore r' is directly proportional to the output probability of the composition function with a constant of proportionality of 2^n where n is the number of primary inputs. □

It is now clear that the value of r' can be obtained without computing an inner product if the output probability is known. This leads to the main result of this paper and is given in theorem 2.

Theorem 2: A RM spectral coefficient is a function of the output probability of the function being transformed.

Proof: From lemma 1 it was proven that a particular RM coefficient can be computed from a real value as

$$r = r'(\mathrm{mod}\,2) \tag{16}$$

From lemma 2, the value r' is related to the output probability, $\wp\{f_{comp}\}$. Substituting eqn. 15 into eqn. 16 yields the final result as given in eqn. 17

$$r = (2^n \wp\{f \cdot g_c\})(\mathrm{mod}\,2) \tag{17}$$

□

Theorem 2 provides the necessary theory for the implementation of a method for computing the RM spectra without requiring storage resources proportional to the size of a functions' truth table.

3 Implementation

The mathematical basis for the synthesis of GRM combinational logic circuits discussed in the previous Section is used as a basis for the implementation of a behavioural to structural translation tool. This tool allows the designer to specify the logic functions of a system in a compact behavioural form without having to translate the BDDs into a structural form.

This tool is implemented in two main modules. The first module receives the BDD description and the polarity number of the desired GRM form as input. Upon receiving these inputs, a BDD representation of each composition function is formed sequentially. After the formation of the g_c BDD, the *APPLY* algorithm [11] is invoked thus forming the BDD representing the composition function, f_{comp}. Next, the PAA is invoked resulting in the output probability which is converted to the RM spectral coefficient according to eqn. 17. Each time a coefficient is computed, it is passed to the second module which generates the output netlist.

Many logic functions are represented in a very compact form when they are expressed as BDDs [12] in contrast to to the exponentially large (in terms of number of inputs) size required by a truth table, and hence an output vector. This fact allows the methodology implemented here to compute a RM coefficient to be on the order of $O(\|E_{comp}\|)$ where $\|E_{comp}\|$ is the number of edges in the BDD representing f_{comp}. However, there are some functions that have a number of BDD nodes that are comparable to the number of truth table values [13,14]. For these cases, our methodology degenerates to having a complexity equivalent to performing the matrix calculations. Fortunately, many functions of practical importance are described with far fewer BDD nodes than truth table entries.

The overall time complexity of our approach is $O(\|E_f\| \times \|E_{gc}\|)$ for each RM coefficient since each application of the apply algorithm visits each node in the BDD representing f and fc. However, it should be noted that the constituent functions for the RM transform are extremely simple and result in very small BDDs. Most of the time $\|E_{comp}\| < \|E_f\| \times \|E_{gc}\|$ so the PAA algorithm requires relatively little computation time. The storage requirement is also of the order $O(\|E_{comp}\|)$ since all that is required is the storage of the composition function BDD which is generally much smaller than that required for storing the RM transformation matrix.

The second portion of this tool is also written using the *C* programming language and its function is to produce the resulting netlist. We chose to use the *Verilog* hardware description language (HDL) as the output medium for this tool, but any netlist format would have been possible. An overall block diagram of this tool is given in Fig. 1.

The translator is divided into these two components for various reasons. Chiefly, by separating the synthesis calculations, we could easily interchange other codes for this purpose. This will allow us to utilise the existing front-end for synthesis of functions that produce circuit types other than GRM. The two modules were implemented in the *C* programming language since the computations in the tool are of a numerical and string manipulation nature. The *C* library that contains 'string' functions was especially handy for the back-end of the tool that produces the *Verilog* netlist output.

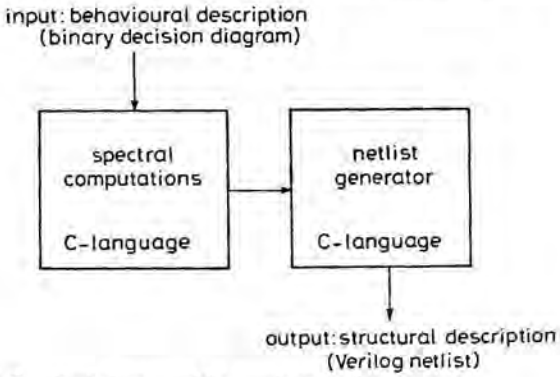

Fig. 1 *Block diagram of the translation tool*

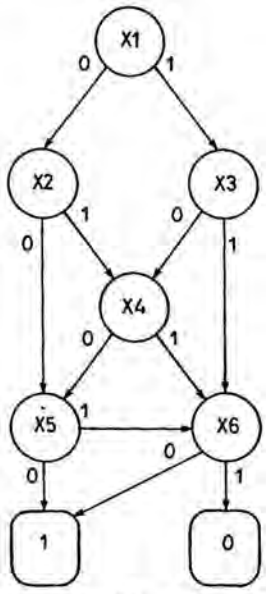

Fig. 2 *Binary decision diagram of function in example 1*

4 Experimental results

There are different ESOP forms of realisation for a given logic function. However, for a specific set of of product terms (i.e. a set that can be used to formulate a coefficient matrix, **G**, of full rank), there is a single unique GRM form. For this reason, there is no concept of minimisation for a particular GRM form although some GRM forms are smaller than others. Finding the minimal GRM form efficiently is currently an open problem and is beyond the scope of this paper.

For the sake of simplicity, the first example is a small six-input function. This allows a diagram of the BDD and the full netlist to be given. To illustrate that the program is capable of computing results for larger circuits, the second example consists of a table containing a subset of RM spectral coefficients for the *ISCAS*85 benchmark circuit, *c*432.

Example 1: The example circuit is a six-input function described by the BDD shown in Fig. 2.

The behavioral description supplied by the BDD was used as input to the translator along with a polarity value of 0. For the sake of clarity, the minimised AND/OR form of this function is given in equation form in eqn. 18 and the corresponding canonical SOP form is given in eqn. 19.

$$f(x) = \bar{x}_1\bar{x}_2\bar{x}_5 + \bar{x}_1\bar{x}_2x_5\bar{x}_6 + \bar{x}_1x_2\bar{x}_4\bar{x}_5 + \bar{x}_1x_2\bar{x}_4x_5\bar{x}_6 + \bar{x}_1x_2x_4\bar{x}_6 + x_1\bar{x}_3\bar{x}_4\bar{x}_5 + x_1\bar{x}_3\bar{x}_4x_5\bar{x}_6 + x_1\bar{x}_3x_4\bar{x}_6 + x_1x_3\bar{x}_6 \quad (18)$$

$$f(x) = \Sigma(0,1,2,4,5,6,8,9,10,12,13,14,16,17,18, 20,22,24,25,26,28,30,32,33,34,36,38,40, 42,44,46,48,49,50,52,54,56,58,60,62) \quad (19)$$

The netlist generator portion of the tool created the following output. Since the circuit was relatively small it was easily verified exhaustively and found to be correct.

```
module rmcircuit (f, inp1 , inp2 , inp3 , inp4 , inp5 , inp6 );
output f;
input inp1,inp2,inp3,inp4,inp5,inp6;
and g_0 ( wir0 , inp1, inp2);
and g_1 ( wir1 , inp2, inp5);
and g_2 ( wir2 , inp1, inp5);
and g_3 ( wir3 , inp1, inp4, inp5);
and g_4 ( wir4 , inp2, inp4, inp5);
and g_5 ( wir5 , inp1, inp3, inp5);
and g_6 ( wir6 , inp1, inp2, inp5);
and g_7 ( wir7 , inp2, inp4, inp6);
and g_8 ( wir8 , inp1, inp2, inp4, inp5);
and g_9 ( wir9 , inp1, inp3, inp4, inp5);
and g_10 ( wir10 , inp1, inp3, inp5, inp6);
and g_11 ( wir11 , inp1, inp4, inp5, inp6);
and g_12 ( wir12 , inp1, inp2, inp3, inp6);
and g_13 ( wir13 , inp1, inp2, inp3, inp5, inp6);
and g_14 ( wir14 , inp1, inp3, inp4, inp5, inp6);
and g_15 ( wir15 , inp1, inp2, inp4, inp5, inp6);
and g_16 ( wir16 , inp1, inp2, inp3, inp4, inp6);
and g_17 ( wir17 , inp1, inp2, inp3, inp4, inp5, inp6);
xor g_18 ( f, inp1, inp2, wir1, wir2, wir3, wir4, wir5, wir6,
wir7, wir8, wir9, wir10, wir11, wir12, wir13, wir14, wir15,
wir16, wir17);
endmodule
```

For the purposes of illustration, the equivalent polarity-0 RM equation is given in eqn. 20.

$$f(x) = x_1 \oplus x_2 \oplus x_1x_2 \oplus x_2x_5 \oplus x_1x_5 \oplus x_1x_4x_5 \oplus x_2x_4x_5 \oplus x_1x_3x_5 \oplus x_2x_4x_6 \oplus x_1x_2x_4x_5 \oplus x_1x_3x_4x_5 \oplus x_1x_3x_5x_6 \oplus x_1x_4x_5x_6 \oplus x_1x_2x_3x_6 \oplus x_1x_2x_3x_5x_6 \oplus x_1x_3x_4x_5x_6 \oplus x_1x_2x_4x_5x_6 \oplus x_1x_2x_3x_4x_6 \oplus x_1x_2x_3x_4x_5x_6 \quad (20)$$

Example 2: The example circuit is the standard benchmark circuit named *c*432 which is part of the *ISCAS*85 set. Table 1 contains a subset of the polarity-0 RM coefficients corresponding to the output labeled 329*gat*. The 329*gat* output of circuit *c*432 depends upon 14 inputs. Table 1 contains only the first-order RM coefficients, that is those coefficients that correspond to the products consisting only of the primary inputs. The first column of the table contains the product term, the second contains the size of the composition BDD in terms of vertices, and the third column contains the RM spectral coefficient. The product terms are given using the corresponding labels in the *ISCAS*85 netlist for designation.

Table 1: First order RM coefficients for output 329gat of Circuit c432

Product	$\|\|f \cdot g_c\|\|$	Coefficient
1*gat*	1387	0
4*gat*	2156	1
11*gat*	1389	0
17*gat*	2160	1
24*gat*	1393	0
30*gat*	2168	1
37*gat*	1401	0
43*gat*	2184	1
50*gat*	1417	0
60*gat*	1488	1
73*gat*	1558	1
86*gat*	1712	1
99*gat*	2028	1
112*gat*	2027	1

The netlist for this example has a similar structure to that in the first example. For those coefficients that are 0-valued, no output is generated in the netlist, for those that are 1-valued, a product term is output to the netlist file.

These results show the input and output of our translation tool. The output in these examples is in GRM form with all inputs uncomplemented. This is the polarity-0 GRM form. Other unique GRM forms can also be computed by providing alternate polarity number inputs.

5 Conclusions

We have presented a behavioural to structural translator that utilises the *Verilog* HDL for output. The translator produces a netlist in GRM form offering the many advantages present in this class of circuits such as ease of testability and a reduced logic gate count. The tool is constructed in a modular manner so that other synthesis functions may be used as they become available.

The mathematical methodology used as a basis for the implementation of the synthesis function in this tool has been presented. Further, an example circuit has been translated and the results provided.

Further work is planned to incorporate heuristics for estimating an optimal polarity number in the front end of this tool based upon the structure of the input BDD. The goal will be to estimate polarity numbers that minimise the number of AND gates resulting in the output netlist. We also plan to extend this technique to use shared BDDs such as those proposed in [15] so that multi-output netlists may be generated.

6 Acknowledgments

This project was supported in part by NSF under grant MIP-9410822.

7 References

1 SASAO, T., and BESSLICH, P.: 'On the complexity of mod-2 sum PLAs', *IEEE Trans.*, 1990, **C-39**, (2), pp. 262–266
2 ROLLWAGE, U.R.: 'The complexity of mod-2 sum PLA's for symmetric functions'. Proceedings IFIP WG 10.5 Workshop on *Applications of the Reed-Muller Expansion in Circuit Design*, September 1993, pp. 6–12
3 REDDY, S.M.: 'Easily testable realizations for logic functions', *IEEE Trans.*, 1972, **C-21**, (11), pp. 1183–1188
4 SARABI, A., and PERKOWSKI, M.: 'Fast exact quasi-minimal minimization of highly testable fixed-polarity ANDXOR canonical networks'. Proceedings of the IEEE *Design Automation* Conference, 1992, pp. 30–35
5 THORNTON, M.A., and NAIR, V.S.S.: 'Efficient spectral coefficient calculation using circuit output probabilities', *Digital Sig. Proc.: Rev. J.*, 1994, pp. 245–254
6 BARTEE, T.C., LEBOW, I.L., and REED, I.S.: 'Theory and design of digital machines' (McGraw-Hill, New York, 1962)
7 LARNEY, V.H.: 'Abstract algebra a first course' (Prindle, Weber, and Schmigt, Boston, MA, 1975)
8 GREEN, D.: 'Modern logic design' (Addison-Wesley, Reading, MA, 1986)
9 DAVIO, M., DESCHAMPS, J.-P., and THAYSE, A.: 'Discrete and switching functions' (McGraw-Hill, New York, 1978)
10 PARKER, K.P., and MCCLUSKEY, E.J.: 'Probabilistic treatment of general combinational networks', *IEEE Trans.*, 1975, **C-24**, pp. 668–670
11 BRYANT, R.E.: 'Graph-based algorithms for Boolean function manipulation', *IEEE Trans.*, 1986, **C-35**, (8), pp. 677–691
12 AKERS, S.B.: 'Binary decision diagrams', *IEEE Trans.*, 1978, **C-27**, (6), pp. 509–516
13 DEVADAS, S.: 'Comparing two-level and ordered binary decision diagram representations of logic functions', *IEEE Trans. CAD Integrated Circuits and Systems*, 1993, **12**, (5), pp. 722–723
14 BRYANT, R.: 'On the complexity of VLSI implementations and graph representations of Boolean functions with applications to integer multiplication', *IEEE Trans. Comput.*, 1991, **40**, (2), pp. 205–213
15 MINATO, S., ISHIURA, N., and YAJIMA, S.: 'Shared binary design diagram with attributed edges for efficient Boolean function manipulation'. Proceedings of the ACM/IEEE *Design Automation* Conference, 1990, pp. 52–57