Direct Reed-Muller Transform of Digital Logic Netlists

Mitchell A. Thornton and Jennifer Dworak
Southern Methodist University
Dallas, Texas, USA

Abstract

A method for computing the Reed-Muller spectrum of a digital logic circuit based on the circuit topology is developed. The technique is based on the definition and use of a transfer matrix to characterize the circuit of interest. The transfer matrix is computed through the use of the structure of the circuit and the transfer matrices of the individual logic gates. The resulting transfer matrix describes the Boolean domain response of the circuit. We next formulate the transfer matrix in the Reed-Muller domain that is also capable of being similarly formulated directly from the structure of the circuit. The result is that the Reed-Muller spectrum of a circuit, or subcircuit partition, can be computed directly thus alleviating the need to first determine the switching function followed by subsequent application of the RM transform.

1.0 INTRODUCTION

Many applications for spectral analysis of digital logic circuits have been developed including the use of the Reed-Muller family of spectra. Central to these applications is the computation of the spectra in an efficient manner since resulting applications are often hampered by the initial large computational expense of the spectrum. Past methods rely upon transforming the switching functions representing the circuit behavior. Techniques to address the spectral computation problem have been devised that utilize cube list and decision diagram representations, however, all of these approaches are exponentially complex in the worst case since the entire switching function must be represented in some form prior to transformation.

The concept of a transfer function describing the input-output relation of a linear system is a mature idea and is heavily used in electrical circuit analysis. Such transfer functions have the property of allowing for their construction based on the transfer functions of individual system components. This hierarchical construction of transfer functions is useful for dealing with large systems allowing the overall system characterization to be achieved in a bottom-up manner. Furthermore, such transfer functions can be transformed to frequency domain representations allowing for design and analysis techniques to be employed in alternative basis domains.

We describe a technique that enables frequency domain analysis of digital logic circuits through the use of the Reed-Muller transform applied directly upon a logic circuit netlist. The technique is based on formulating a linear algebraic model of the switching function allowing for the overall circuit switching function to be represented by a characteristic linear transfer matrix in the Boolean domain. The transfer matrix is analogous to a transfer function or matrix as is commonly used in classical linear system analysis. Linear system theory is a mature approach and many excellent references are available, an example of which is [1]. Preliminary ideas used to formulate these results are briefly outlined in [2] but they were only discussed in the context of a single logic gate in the Boolean domain and they utilized a different form of the transfer matrix than that described here.

Because the circuit transfer matrix can be derived from a hierarchical construction of the individual logic gate transfer matrices, the technique is applicable to the use of the structure of a netlist or interconnection of individual logic gates. Furthermore, the RM transform may be applied at the single gate or larger subcircuit level avoiding the memory explosion that can occur when the entire circuit model switching function is transformed. A similar formulation for the Walsh transform is described in [3] although the transfer matrices were formulated as transposes of those used here. An alternative method for the computation of single Walsh spectral coefficients was described in [4], however, this method required structural modification of the circuit netlist before it was traversed for the computation of each coefficient.

The Reed-Muller transfer function response of the entire circuit can also be derived in a hierarchical manner. These results lead to a method for computing the Reed-Muller transform of a circuit or subcircuit directly from a netlist representation. An application of this technique is that the Reed-Muller transform of partitioned subcircuits can be computed directly from the netlist without resorting to first computing the representation of the overall switching function.

The organization of the paper is as follows. A review of the mathematical properties of the Reed-Muller transform and its definition are first reviewed. Next, we will describe a formulation of classical digital logic circuits utilizing linear algebra rather than Boolean algebra analogous to time domain analysis of linear electrical circuits. Following these introductory concepts, the Reed-Muller transform of digital circuit netlists is formulated using the concepts of linear algebraic analysis. Finally, conclusions and future work are outlined.

2.0 REED-MULLER TRANSFORM

The Reed-Muller transform has been previously described and written about extensively [5, 6]. In this paper, we will provide a brief overview of the transform for the sake of completeness and to emphasize its relationship with other transforms.

One way to conceptualize discrete spectral transforms is to consider the set of function range values to be transformed as a set of coefficients over which to evaluate a particular polynomial. The particular polynomial that is selected defines both the transform and the discrete function undergoing transformation. For example, if the polynomial is based on various powers of the roots of unity with coefficients representing the discrete range space of a function, *f*(*x*), the discrete Fourier transform results. The form of the polynomial is shown in Equation (1).

$$p(x) = a_0x^0 + a_1x^1 + a_2x^2 + \ldots + a_{n-1}x^{n-1} \quad (1)$$

Discrete values of the polynomial can be computed by expressing the polynomial in the form of a linear transformation as shown in Equation (2). The coefficient matrix of Equation (2) is a Vandermonde matrix since it contains row vector components that are in the form of a geometric series.

$$\begin{bmatrix} x_0^0 & x_0^1 & x_0^2 & \cdots & x_0^k & \cdots & x_0^{n-1} \\ x_1^0 & x_1^1 & x_1^2 & \cdots & x_1^k & \cdots & x_1^{n-1} \\ x_2^0 & x_2^1 & x_2^2 & \cdots & x_2^k & \cdots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \iddots & x_0^{n-1} \\ x_m^0 & x_m^1 & x_m^2 & \cdots & x_m^k & \cdots & x_m^{n-1} \\ \vdots & \vdots & \vdots & \iddots & \vdots & \ddots & \vdots \\ x_{n-1}^0 & x_{n-1}^1 & x_{n-1}^2 & \cdots & x_{n-1}^k & \cdots & x_{n-1}^{n-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_m \\ \vdots \\ a_{n-1} \end{bmatrix} = \begin{bmatrix} p(x_0) \\ p(x_1) \\ p(x_2) \\ \vdots \\ p(x_m) \\ \vdots \\ p(x_{n-1}) \end{bmatrix} \quad (2)$$

When the roots of unity are used for the x_i values in Equation (2) and they are restricted to the square roots of unity, the discrete Fourier transform matrix over *GF*(2) results. Depending on the representation of the square roots of unity, various different transformations result and are commonly known as the Walsh [7], Hadamard [8, 9], or Reed-Muller transform [5, 6]. When a particular discrete function, *f*(*x*) is used for the a_i coefficient vector, Equation (2) can be evaluated and the $p(x_i)$ components are the discrete Fourier transform coefficients of *f*(*x*). Equation (3) contains an expression for the *m* roots of unity, that are used as the x_i in Equation (2).

$$x_m^k = e^{\frac{2\pi ik}{m}} \quad (3)$$

Figure 1 contains a diagram of the square roots of unity plotted on the unit circle in the complex plane.

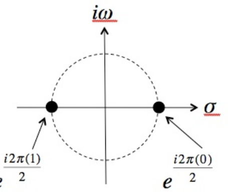

Figure 1: Square Roots of Unity

When the values of x_i are used in the transformation matrix of Equation (2) for *m*=2, the Walsh or Hadamard transform results. Likewise, when the values of *k* are used for *m*=2, as shown in Figure 1, the Reed-Muller transform results. As an example, the polarity-0 Reed-Muller transformation matrix, $\mathbf{R}_3$, for a function of *n*=3 variables has the form as shown in Equation (4).

$$\mathbf{R}_3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad (4)$$

Due to the mathematical properties of Vandermonde matrices and those of the roots of unity that are used within the matrix, efficient algorithms such as the so-called “fast” transforms have been developed and are usually credited to the work in [10, 11] although the method was actually used by the famous mathematician, Carl Gauss in the early 1800’s. These methods were later applied to efficient discrete function representations in the form of decision diagrams [12, 13]. These methods are applicable for the computation of the RM transform if the discrete function to be transformed is represented as the vector of coefficients, $[a_i]$ in Equation (2). The difficulty in applying these "fast" formulations is that the vector $[a_i]$ is exponential in size, $O(2^n)$ for the case of the RM transform.

The approach described here allows for frequency responses of the discrete function to be computed directly from the digital logic netlist representation of the discrete function. Furthermore, the spectrum may be computed in a hierarchical fashion by computing the frequency responses of partitioned subcircuits from the netlist.

3.0 NETLIST ANALYSIS

Traditionally, digital logic circuits are modeled using the axioms and postulates of binary-valued Boolean algebra and discrete scalar-valued switching functions. Here, we reformulate these mathematical models in terms of linear transforms over vector spaces. Through the representation of switching functions in this manner, various spectral transformations are easily applied including the Reed-Muller transform.

Linear algebraic notation is expressed using the 'bra-ket' notation devised by physicist Paul Dirac for the purposes of quantum mechanical calculations [14]. The column vector **a** is represented as 'ket $|a\rangle$'. Likewise, the row vector $\mathbf{a}^T$ is represented as a 'bra' $\langle a|$. This notation is particularly convenient when the norm of a vector or the inner product of two vectors is expressed. The norm of a vector **a** written as $||\mathbf{a}||$ can be expressed in bra-ket notation as $\langle a|a\rangle$ and the inner product of two vectors **a** and **b** can be written as $\langle a|b\rangle$. The outer product, or tensor product of two 1st-order tensors **a** and **b** can be expressed as $|a\rangle\langle b|$. Using this notation, many naturally arising 2nd-order tensors, or, matrices can be expressed as a sum of outer product expressions.

The binary constants {0,1} are represented as the row vectors $\langle 0|=[1\ 0]$ and $\langle 1|=[0\ 1]$. A transfer matrix for a logic gate can be expressed in terms of the outer product operation analogous to the transfer function of a linear system. The outer product is computed as the tensor, or Kronecker, product of two vectors or matrices. Row (or column) vectors representing multiple bits are likewise computed using the outer product. As an example, $\langle 01|=\langle 0|\otimes\langle 1|=[1\ 0]\otimes[0\ 1]=[0\ 1\ 0\ 0]$.

3.1 Logic Gate Transfer Matrix

Using the definitions of the switching circuit constants and the linear algebraic notation of the previous section, a transfer matrix for a particular logic gate can be formulated. The transfer matrix can be used to derive the logic gate output for a given input stimulus.

Definition 1:

The *transfer matrix* for a logic gate, $\mathbf{T}_{\mathbf{GATE}}$, can be computed as the sum of outer products of vectors formed from the truth table input and output values. □

Example 1:

As an example, the transfer matrix for a two-input XOR gate is expressed as:

$$\mathbf{T}_{XOR} = |00\rangle\langle 0| + |01\rangle\langle 1| + |10\rangle\langle 1| + |11\rangle\langle 0|$$

$$\mathbf{T}_{XOR} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}\begin{bmatrix}1 & 0\end{bmatrix} + \begin{bmatrix}0\\1\\0\\0\end{bmatrix}\begin{bmatrix}0 & 1\end{bmatrix} + \begin{bmatrix}0\\0\\1\\0\end{bmatrix}\begin{bmatrix}0 & 1\end{bmatrix} + \begin{bmatrix}0\\0\\0\\1\end{bmatrix}\begin{bmatrix}1 & 0\end{bmatrix}$$

$$\mathbf{T}_{XOR} = \begin{bmatrix}1&0\\0&0\\0&0\\0&0\end{bmatrix} + \begin{bmatrix}0&0\\0&1\\0&0\\0&0\end{bmatrix} + \begin{bmatrix}0&0\\0&0\\0&1\\0&0\end{bmatrix} + \begin{bmatrix}0&0\\0&0\\0&0\\1&0\end{bmatrix}$$

$$\mathbf{T}_{XOR} = \begin{bmatrix}1&0\\0&1\\0&1\\1&0\end{bmatrix}$$

□

Examination of the transfer matrix of $\mathbf{T}_{\mathbf{XOR}}$ reveals that each row of the matrix is simply the truth table output encoded as vector constants. The second column is the truth table output vector and the first column is the inverse function truth vector. These observations allow for efficient methods of function representation to be used for transfer matrices such as Binary Decision Diagrams (BDDs) or cube lists.

The term 'transfer matrix' is used since it is analogous to the 'transfer function' of linear systems as commonly used in classical linear system analyses [1]. As is the case with transfer functions, the output response is easily obtained by multiplying the transfer matrix of a gate with a particular input stimulus.

Example 2:

The output response of the XOR gate when the input is two logic-1 values, ⟨11|, is calculated as the direct matrix product.

$$\langle 11|\mathbf{T}_{XOR} = \left(\begin{bmatrix} 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 \end{bmatrix}\right)\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\langle 11|\mathbf{T}_{XOR} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\langle 11|\mathbf{T}_{XOR} = \begin{bmatrix} 1 & 0 \end{bmatrix} = \langle 0|$$

□

3.2 Transfer Matrices of Logic Circuits

The overall transfer matrix of a logic circuit may be calculated using the individual component transfer matrices. In general, a logic netlist can contain fanin and fanout points, $\mathbf{F_{IN}}$ and $\mathbf{F_{OUT}}$ as depicted in Figure 2. Transfer matrices must also be used to model these circuit nodes and are represented in Equations (5) and (6). It is noted that the case for a fanin input of ⟨01| or ⟨10| is not included. For the purposes of modeling classical logic, such cases should be included and are technology-dependent. For example, if static CMOS circuitry is being modeled, the 'wired-AND' relationship would result. In general, three-valued logic could also be used (as is done in hardware description languages) and a third unknown logic value '**X**' could be used.

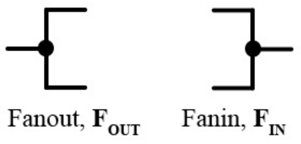

Figure 2: fanout and fanin circuit nodes

$$\mathbf{F}_{OUT} = |0\rangle\langle 00| + |1\rangle\langle 11| = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

$$\mathbf{F}_{IN} = |00\rangle\langle 0| + |11\rangle\langle 1| = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \quad (6)$$

Transfer matrices can be computed for networks of logic gates by partitioning them into serial cascades. In each stage of the cascade, individual gate matrices are combined from top to bottom using the tensor or Kronecker product operation. The overall circuit transfer matrix is calculated as the direct product of the individual cascade stages. Example 3 is the application of this principle and contains a fanout point.

Example 3:

The transfer matrix for the two-input, two-output logic circuit depicted in Figure 3 is computed using the tensor and direct matrix products of the circuit components. The vertical lines indicate the circuit partitions in three distinct stages.

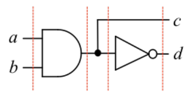

Figure 3: Example Logic Circuit

$$\mathbf{T} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\right)$$

$$\mathbf{T} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

□

As is the case for single-gate transfer matrices, the matrix of the circuit of example 2 can be considered as being comprised of all circuit output row vectors in numerical order. The output responses in this case are in the form of the tensor product of the individual output bits from top to bottom. As an example, the fourth row of **T** is [0 0 1 0]= [0 1] ⊗ [1 0]=⟨1|⊗⟨0|.

Example 4:

The output response for the logic circuit in Figure 4 for the input value ⟨11| is calculated as the direct matrix product with the circuit transfer matrix.

$$\langle 11|\mathbf{T} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} = \langle 10|$$

□

The use of transfer matrices for circuit output response calculations also allows for multiple output responses to be calculated with a single matrix-vector multiply operation. It is necessary to consider the complete set of single bit constants that result from all possible two-dimensional vectors with components 0 and 1. The complete set is defined in terms of a Hasse diagram as shown in Figure 4 and are denoted as

$\langle t|$=[1 1], $\langle 1|$=[0 1], $\langle 0|$=[1 0], and $\langle \varnothing|$=[0 0]. $\langle t|$ denotes a value that is both $\langle 0|$ and $\langle 1|$ while $\langle \varnothing|$ is the case where neither $\langle 0|$ or $\langle 1|$ are present.

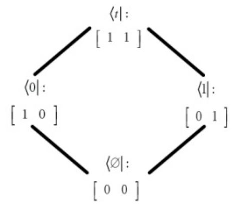

Figure 4: Hasse Diagram of Logic Values

Example 5:

The output response for the logic circuit in Figure 2 for all possible input values $\langle tt|$ is calculated as a direct matrix product.

$$\langle tt|\mathbf{T} = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 3 & 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 3 & 1 & 0 \end{bmatrix} = 3\begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} + 1\begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 3 & 1 & 0 \end{bmatrix} = 3\langle 01| + 1\langle 10| = \langle 01| + \langle 01| + \langle 01| + \langle 10|$$

This result indicates that three of the input cases yield an output of $\langle 01|$ while a single input case results in $\langle 10|$. This is easily verified by examination of the circuit diagram showing that the inputs $\langle 00|$, $\langle 01|$, and $\langle 10|$ result in an output of$\langle 01|$ while the input case $\langle 11|$ results in $\langle 10|$.

□

3.3 Logic Circuit Implication

The determination of a set of input values that cause a particular output response of a circuit is known as *circuit implication*. Logic circuit implication is used in many synthesis, verification, and test algorithms. To compute the implication, the transfer matrix must be invertible. In general the transfer matrices describing a specific logic netlist are non-square and are not of full rank resulting in matrices with no inverses. However, the concept of the pseudo-inverse can be applied to the circuit implication problem.

In particular, the Moore-Penrose pseudo-inverse of the circuit transfer matrix can be used denoted by $\mathbf{T}^{+}$. Depending on the dimensions of the non-square transfer matrix, either the left-hand or right-hand pseudoinverse is used. In the case of multi-input, single-output logic gates, the transfer matrices have dimension $n \times m$ with $n > m$, thus the left-hand pseudoinverse is used as shown in Equation (7).

$$\mathbf{T}^{+} = (\mathbf{T}^{*}\mathbf{T})^{-1}\mathbf{T}^{*} \tag{7}$$

Where **T*** denotes the conjugate transpose of matrix **T**. In the case of classical logic as considered here, the $\mathbf{T}^{*} = \mathbf{T}^{T}$ since all matrix components are real and Equation (7) is rewritten as shown in Equation (8).

$$\mathbf{T}^{+} = \left(\mathbf{T}^{T}\mathbf{T}\right)^{-1}\mathbf{T}^{T} \tag{8}$$

As an example, consider the case where a logic-0 is observed at the output of the AND gate. The implication calculation becomes:

$$\langle 0|\mathbf{T}^{-1}_{\text{AND}} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \end{bmatrix}$$

$$\langle 0|\mathbf{T}^{-1}_{\text{AND}} = \frac{1}{3}\left(\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}\right)$$

$$\langle 0|\mathbf{T}^{-1}_{\text{AND}} = \frac{1}{3}((\langle 00| + \langle 01| + \langle 10|)$$

Thus, the result of the implication is that the inputs $\{\langle 00|, \langle 01|, \langle 10|\}$ result in an output response of $\langle 0|$.

4.0 RM TRANSFORMS OF CIRCUITS

The polarity-zero Reed-Muller transform of a discrete switching function, *f*, can be calculated through the application of the transformation matrix, $\mathbf{R}_0$, to the column vector, $\mathbf{f}_1$, representing the discrete range values. The $\mathbf{R}_0$ transformation matrix for a function of one variable is defined in Equation (9) and can used as a kernel to generate higher-dimensioned transformations through the application of the tensor product as shown in Equation (10).

$$\mathbf{R}_1 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \tag{9}$$

$$\mathbf{R}_n = \bigotimes_{i=1}^{n} \mathbf{R}_i \tag{10}$$

From the previous section, the column vector $\mathbf{f}_1$ can be obtained as the result of the direct product of the circuit transformation matrix, **T**, with a particular input row vector, **x**. Thus the Reed-Muller transform, $\mathbf{f}_{RM}$, can be obtained as shown in Equation (11).

$$\mathbf{f}_{\text{RM}} = \mathbf{R}_n \langle x|\mathbf{T} \tag{11}$$

The RM transform of all single-bit constants are computed as the linear transform of $\mathbf{R}_0$ with the tensor representations. Table 1 summarizes all constant values in both the natural and RM domain. It is noted that the RM transformed values of the constants is a permutation of those as depicted in the Hasse diagram in Figure 4. Figure 5 contains the Hasse diagram of constant values in the RM domain. Another interesting characteristic is that the RM transform matrix, $\mathbf{R}_1$, can be interpreted as consisting of a top row vector that represents RM domain value of $\langle 0|$ and the bottom row vector represents the RM domain value of $\langle 1|$.

Table 1: Tensor Constant Values

	Logic-∅	Logic-0	Logic-1	Logic-*t*
Boolean	[0 0]	[1 0]	[0 1]	[1 1]
Reed-Muller	[0 0]	[1 0]	[1 1]	[0 1]

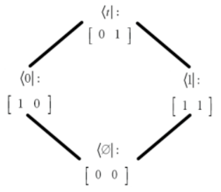

Figure 5: Hasse Diagram of RM Logic Values

An alternative definition of the Reed-Muller transform can now be formulated based on the representation of constants in the Boolean and Reed-Muller domains.

<u>*Definition*</u> 2:

The *transfer matrix* describing the mapping from the constants in the Boolean Domain to the Reed-Muller domain is the Reed-Muller transform, $\mathbf{R}_1$.

$$\mathbf{T} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}\begin{bmatrix} 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}\begin{bmatrix} 1 & 1 \end{bmatrix}$$

$$\mathbf{T} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = \mathbf{R}_1$$

□

4.1 RM Transfer Matrices of Logic Gates

The RM response of a logic gate may be computed by using the RM transform of the gate transfer matrix. As an example, consider the two-input OR gate whose Boolean transfer matrix is given as $\mathbf{T}_{OR}$. Notice that these calculations utilize modulo-2 addition operator, ⊕, during the matrix multiply calculations.

$$\mathbf{T}_{OR} = |00\rangle\langle 0| + |01\rangle\langle 1| + |10\rangle\langle 1| + |11\rangle\langle 1|$$

$$\mathbf{T}_{OR} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \quad (12)$$

The RM transfer matrix of the OR gate is computed as $\mathbf{R}_2\mathbf{T}_{OR}$.

$$\mathbf{R}_2\mathbf{T}_{OR} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \quad (13)$$

To obtain the RM frequency response of the OR gate for a particular input value, its RM transformed value is multiplied by the RM transform gate matrix. As an example, the response of an OR gate when the inputs are both logic-0, $\langle 00|$, is computed as shown. To obtain the RM frequency response, the input row vector must be transformed to the RM domain as [1 0 0 0]. The resultant output is the RM frequency domain response since the RM transfer matrix of the OR gate is multiplied by the Boolean domain input value.

$$\mathbf{R}_2\langle 00|\mathbf{R}_2\mathbf{T}_{OR} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$\mathbf{R}_2\langle 00|\mathbf{R}_2\mathbf{T}_{OR} = \begin{bmatrix} 1 & 0 \end{bmatrix} = \mathbf{R}_2\langle 0|$$

4.2 RM Transfer Matrices of Netlists

The RM frequency response can also be calculated from general netlists in two ways. The explicit approach is to first compute the transfer matrix of the entire circuit in the Boolean domain and then to apply the appropriately dimensioned RM transform matrix, $\mathbf{R}_n$, where n is the number of dependent inputs of the circuit. An example of this form of computation was shown in the previous section for the two-input OR gate. Note that the second column of the RM transfer matrix of the OR gate is in fact the zero-polarity RM spectrum of the OR function.

The second, and more interesting, method allows for hierarchical calculation of the circuit transform by traversing the netlist. To illustrate the method, consider the circuit shown in Figure 6.

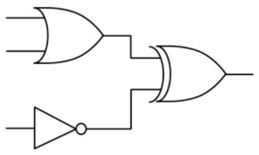

Figure 6: Circuit for Example 6

<u>*Example*</u> 6:

The Boolean domain transfer matrix for the circuit in Figure 6 is computed as follows:

$$\mathbf{T} = \left(\mathbf{T}_{\mathrm{OR}} \otimes \mathbf{T}_{\mathrm{NOT}}\right)\mathbf{T}_{\mathrm{XOR}}$$

Application of the RM transform to the entire circuit would require the use of $\mathbf{R}_3$ since the circuit of Figure 6 has three inputs:

$$\mathbf{R}_3\mathbf{T} = \mathbf{R}_3\left(\mathbf{T}_{\mathrm{OR}} \otimes \mathbf{T}_{\mathrm{NOT}}\right)\mathbf{T}_{\mathrm{XOR}}$$

Making use of the tensor product identity:

$$(a \otimes b)(c \otimes d) = ac \otimes bd$$

$$\mathbf{R}_3\mathbf{T} = \left(\mathbf{R}_2 \otimes \mathbf{R}_1\right)\left(\mathbf{T}_{\mathrm{OR}} \otimes \mathbf{T}_{\mathrm{NOT}}\right)\mathbf{T}_{\mathrm{XOR}}$$

$$\mathbf{R}_3\mathbf{T} = \left(\mathbf{R}_2\mathbf{T}_{\mathrm{OR}} \otimes \mathbf{R}_1\mathbf{T}_{\mathrm{NOT}}\right)\mathbf{T}_{\mathrm{XOR}}$$

The computation becomes:

$$\mathbf{R}_3\mathbf{T} = \left(\begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}\right)\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\mathbf{R}_3\mathbf{T} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 0 & 0 \end{bmatrix}$$

□

The second column of the resultant matrix is the RM transform of the overall switching function represented by the circuit in Figure 6. The first column represents the RM spectrum of the inverse of the function. The first column contains a "1" in the topmost position which translates into $1 \oplus \mathbf{f}_1 = \mathbf{f}_0$.

The RM frequency response can also be computed using the RM transfer matrix of a circuit. As an example, consider the multi-output circuit in Figure 3. Example 7 shows how the RM frequency response can be used to compute all possible output values for the circuit.

<u>Example 7</u>:

The RM frequency response for the logic circuit in Figure 3 is calculated as:

$$\mathbf{R}_2\langle tt|\mathbf{R}_2\mathbf{T} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\mathbf{R}_2\langle tt|\mathbf{R}_2\mathbf{T} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$$\mathbf{R}_2\langle tt|\mathbf{R}_2\mathbf{T} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix} \oplus \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}$$

□

The frequency response vector of Example 7 must be decomposed in the RM domain using the modulo-2 addition operator. To verify the final equation of Example 7 is indeed the correct decomposition, the following equations transform the indicated circuit outputs $\langle 01|$ and $\langle 10|$ directly, then they show that the modulo-2 sum results in the result of Example 7.

$$\mathbf{R}_2\langle 01| = \begin{bmatrix} 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix}$$

$$\mathbf{R}_2\langle 10| = \begin{bmatrix} 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}$$

$$\mathbf{R}_2\langle 01| \oplus \mathbf{R}_2\langle 10| = \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix}$$

4.3 Non-zero Polarity RM Transform

The hierarchical technique can also be applied to circuits for other polarities of the RM transform. So far, all the examples in this paper have utilized the 0-polarity RM transform meaning that all literals (or circuit inputs) are assumed to be non-inverted. There exist 2^n different RM transforms for a circuit of n inputs. The polarity number, in binary form, indicates the polarity of a particular circuit input. Example 6 illustrated the computation of the transform of the circuit when all inputs are assumed to be present in positive-polarity or non-inverted form.

If the bottom-most input of the OR gate is assumed to present in inverted form, the corresponding transform would be a polarity-2 transform. To account for the inversion of this circuit input, the $\mathbf{R}_2$ matrix used in Example 6 would be modified as shown in the following equation.

$$\hat{\mathbf{R}}_3\mathbf{T} = \left(\left(\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}\right)\mathbf{T}_{\mathrm{OR}} \otimes \mathbf{R}_1\mathbf{T}_{\mathrm{NOT}}\right)\mathbf{T}_{\mathrm{XOR}}$$

$$\hat{\mathbf{R}}_3\mathbf{T} = \left(\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\right)\mathbf{T}_{\mathrm{XOR}}$$

$$\hat{\mathbf{R}}_3\mathbf{T} = \left(\begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}\right)\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\hat{\mathbf{R}}_3\mathbf{T} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 0 & 0 \end{bmatrix}$$

5.0 APPLICATION OF APPROACH

One of the common uses of the RM transformation is to replace circuits with their RM-equivalent form consisting of product terms combined by a multi-input XOR gate. To compute the RM transform of a function, the traditional approach is to multiply the range vector of the representative switching function with an appropriately dimensioned **R** matrix. The number of product terms in the RM transformed circuit is then equivalent to the non-zero terms in the RM spectral vector.

Using the methodology above, the RM transform may be computed through direct traversal of a netlist. Furthermore, the netlist can be partitioned and the RM transform can be computed on internal subcircuits present in the netlist.

There are multiple reasons why the RM form of a subcircuit may be preferable to the original. In some cases, the RM version may require fewer gates and thus allow the area to be reduced. In addition, using the RM form of a subcircuit can make a subcircuit more testable [15]. One application is to perform localized Reed-Muller transforms on internal subcircuits and to replace those subcircuits with corresponding RM forms if it is deemed that the RM form of the subcircuit has more desirable properties than that of the original subcircuit.

As an example of replacing an internal subcircuit with its RM equivalent, consider the logic circuit depicted in Figure 7. The outlined portion of the circuit is the partition of interest and the methods described previously are used to compute the RM transform.

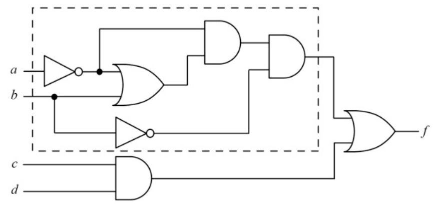

Figure 7: Circuit with Partition

The polarity-0 RM transfer matrix of the subcircuit indicated by the dotted box in Figure 7 has the following form.

$$\mathbf{R}_2\mathbf{T}_{\text{part}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}$$

$$\mathbf{R}_2\mathbf{T}_{\text{part}} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \tag{14}$$

Depending on the goals of the local substitution methodology in the synthesis tool, it may be determined that it is advantageous to replace the subcircuit with its equivalent RM form. Performing this replacement yields the circuit in Figure 8. The advantage of this approach is that that the RM transform was computed based on a logic circuit partition and the spectrum of the overall switching function using the 16×16 matrix, $\mathbf{R}_4$, was never required.

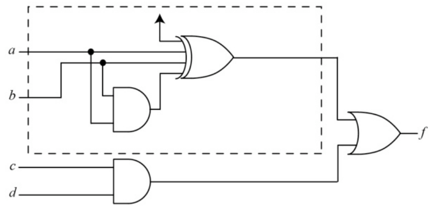

Figure 8: Updated Circuit of Figure 7

As previously mentioned, the second column of the RM transfer matrix in Equation (14) is the 0-polarity RM spectrum of the subcircuit while the first column represents the spectrum of the inverse of the subcircuit. In this case, the first column has one fewer "1" components indicating that the inverse circuit would not require an XOR gate input tied to a constant "1" value. To take advantage of this circumstance, the XOR gate in the circuit partition could be replaced with an exclusive-NOR gate that does not require an input to V_{dd}. This further optimization is shown in the circuit depicted in Figure 9.

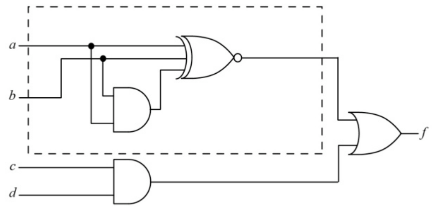

Figure 9: Circuit with XNOR Gate

As an example in the area of testability, consider Figure 10. This figure shows a portion of a larger circuit. If fault collapse is used to remove structurally equivalent faults, the internal faults in the circuit that must be detected during test include: *P* sa1, *Q* sa0, *Q* sa1, *R* sa0, *S* sa0, *T* sa0, *U* sa1, *V* sa1, *W* sa1, and *X* sa1. We are not considering faults on the primary inputs or outputs of the subcircuit as these locations remain functionally unchanged by the RM transformation, thus, their testability will thus remain unchanged as well.

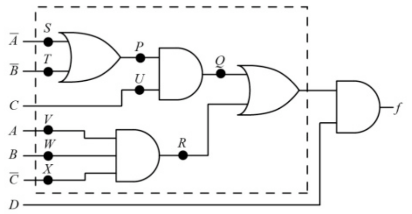

Figure 10: Circuit with Internal Fault Sites

For the sake of simplicity, assume that the inputs to the subcircuit as well as site *D* have disjoint input cones and thus are functionally independent. Furthermore, assume that inputs *A* and *B* each have a 50% chance of being equal to a logic-1 and a 50% chance of being equal to a logic-0. However, assume that input *C* has only a 10% chance of being equal to a logic-1 due to the characteristics of the input cone that drives it. Similarly, site *D* has a 20% chance of being equal to logic-1. Under these conditions, four of the faults have only a 0.5% chance of being detected with a random pattern. Furthermore, even if targeted ATPG is used to generate the patterns, at least 6 patterns must be applied because the test sets of many of the faults are disjoint. Due to these testability concerns, the dashed box in Figure 9 indicates the subcircuit partition of interest. Using the methods of Sections 3 and 4, the Boolean and RM transfer matrices of the partition in Figure 10 result:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad \mathbf{R}_3\mathbf{T} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \\ 0 & 0 \end{bmatrix}$$

Now consider the same subcircuit in Figure 11 where the partition of interest is replaced with the polarity-0 RM equivalent subcircuit. This circuit is much smaller, and the area savings are significant. It is also much more testable. Assuming that *A*, *B*, and *C* do not fanout elsewhere, there is only one internal circuit site *K* we must consider, and thus only two internal faults that must be detected. The probability of detecting the harder-to-detect fault, *K* sa0, is now 5%, an increase in detectability of an order of magnitude as compared to the 0.5% value in Figure 9. At the same time, *K* sa1 has a 15% chance of being detected. Only two test patterns are necessary in a targeted test set to detect these two faults.

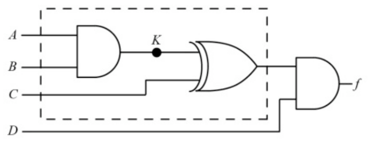

Figure 11: New Circuit with Fault Sites

In general, increases in testability are likely to occur for many potential subcircuits. Hard-to-detect faults are often hard to detect due to the difficulty of fault observation. However, each input of an XOR gate is automatically observable. Thus, replacing a subcircuit with its equivalent RM form results in the presence of internal sites that are likely to be more observable than those of the original circuit.

6.0 IMPLEMENTATION USING BDDS

The application of the result of the previous section is that a subcircuit partition may be analyzed to determine if the RM circuit form is more desirable. A multi-level netlist is partitioned into subcircuits and the RM transfer matrix of each subcircuit is then computed in hierarchical manner by traversing the netlist.

One concern is the large matrices and corresponding arithmetic that would be required in these calculations. However, using Binary Decision diagram representations to perform the matrix arithmetic computations reduces the average complexity of the calculations.

Observation of the form of the Boolean transform matrices for a basic logic gate (or subcircuit for that matter) reveals that, for a single output circuit, the rightmost column vector of the matrix is simply the underlying switching function output vector and the leftmost column vector represents the inverse of the switching function. Thus, a BDD representation of the switching function is interpreted as a representation of the transfer matrix in the Boolean domain. Using this observation, the procedure can be implemented using BDDs as follows:

1) Partition netlist into subcircuits
2) Represent each partition as a BDD
3) Transform the BDD to its' RM form using the efficient methods described in [12, 13]
4) Examine the RM transfer matrix represented by each BDD to determine if it is desirable to replace that portion of the subcircuit with the equivalent RM form.
5) If circuit replacement is deemed desirable, perform the replacement.

7.0 CONCLUSIONS

In the past, RM transformation techniques required first formulating the switching function of the entire circuit followed by application of a subsequent transformation. This paper has presented a method whereby this computationally prohibitive step is avoided and the transformations can be applied locally on smaller circuit partitions. Section 2.0 showed that the RM transform is a special case of the Fourier transform, thus these results are applicable to a wide variety of transforms.

Modeling of classical logic circuits has traditionally been accomplished through the use of switching theory based on Boolean algebras for binary circuits while spectral analysis has been traditionally expressed in terms of linear algebra. This work reformulates classical circuit analysis in terms of linear algebra hence the mathematics of classical and RM spectral netlist analysis is now unified with a common mathematical basis.

Although the transfer matrices are exponentially large, it is noted that their structure is very regular and that efficient representations in terms of BDDs and cube lists can in fact be interpreted as transfer matrix representations. This fact allows the representation complexity to be equivalent to that of traditional switching theory based methods.

The result presented here is an algorithm for performing local transformations of partitioned subcircuits in a netlist and replacing the subcircuits with their Reed-Muller equivalents if a merit function indicates such a substitution is desirable. This technique can be implemented using BDDs for all matrix computations to reduce the average exponential complexity encountered when matrix arithmetic is used explicitly.

Future research will investigate the effective extraction of appropriate subcircuits to which the proposed method may be applied to reduce area and/or enhance testability. We will also consider the insertion of test points at the outputs of subcircuit partitions. Implementation will be accomplished through the use of BDDs for the linear algebraic computations to reduce average complexity.

REFERENCES

[1] Chen, C.-T., **Linear System Theory and Design**, Holt, Rinehart and Winston, ISBN 0-03-060289-0, 1984.

[2] Yanofsky, N.S. and Mannucci, M.A., **Quantum Computing for Computer Scientists**, Cambridge University Press, 2008, ISBN 978-0-521-879965.

[3] Thornton, M.A., “Spectral analysis of digital logic using circuit netlists,” *Proc. of the International Conference on Computer Aided Systems Theory* (EUROCAST), ISBN 978-84-693-9560-8 Modeling and Design of Complex Digital Systems Workshop, pp. 444-445, February 2011.

[4] Drechsler, R. and Thornton, M.A., “Computation of spectral information from logic netlists,” *Proc. Int. Symp. Multiple-Valued Logic*, pp. 53-58, 2000.

[5] Reed, I.S., “A class of multiple-error-correcting codes and their decoding scheme,” *IRE Transactions on Information Theory*, 4:38-49, 1954.

[6] Muller, D.E., “Application of Boolean algebra to switching circuit design and error detection,” *IRE Transactions on Electronic Computers*, 3:6-12, 1954.

[7] Walsh, J.L., “A closed set of normal orthogonal functions,” *American Journal of Mathematics*, 55:5-24, 1923.

[8] Sylvester, J.J., “Thoughts on inverse orthogonal matrices, simultaneous sign successions, and tessellated pavements in two or more colours, with applications to Newton’s rule, ornamental tilework, and the theory of numbers,” *Philosophical Magazine*, 34:461-475, 1867.

[9] Hadamard, J., “Résolution d’une question relative aux déterminants,” *Bulletin des Sciences Mathématiques*, 17:240-246, 1893, (in French).

[10] Good, I.J., “The interaction algorithm and practical Fourier analysis,” *Journal of the Royal Statistical Society B*, 20:2, pp. 361-372, 1958; addendum, ibid. 22(2), pp. 373-375, 1960.

[11] Cooley, J.T. and Tukey, J.W., “An algorithm for the machine calculation of complex Fourier series,” *Math. Computation*, 19:297-301, 1965.

[12] Thornton, M.A., Drechsler, R., and Miller, D.M., **Spectral Techniques in VLSI CAD**, Kluwer Academic Publishers, July 2001, ISBN 0-7923-7433-9.

[13] Stankovic´, R. and Astola, J., **Spectral Interpretation of Decision Diagrams**, Springer-Verlag, ISBN 0-387-95545-3, 2003.

[14] Dirac, P.A.M., “A new notation for quantum mechanics,” *Proceedings of the Cambridge Philosophical Society*, vol. 35, p. 416, 1939.

[15] Reddy, S.M., “Easily testable realizations for logic functions,” *IEEE Transactions on Computers*, vol. 21, no. 11, pp. 1183-1188, 1972.