A Transfer Function Model for Ternary Switching Logic Circuits

Mitchell A. Thornton
Southern Methodist University
Dallas, Texas USA 75275–0122
Email: mitch@lyle.smu.edu

Abstract

Ternary switching functions are formulated as transformations over vector spaces resulting in a characterization in the form of a transfer function. Ternary logic constants are modeled as vectors, thus the transfer functions are of the form of matrices that map vectors representing logic network input values to corresponding output vectors. Techniques for determination of the transfer matrix from a logic switching model or directly from a netlist are provided. The use of transfer matrices for logic network simulation are then developed that allow for multiple output responses to be obtained through a single vector-matrix product calculation.

1. Introduction

Design and analysis tasks require some means to specify logic network functionality that is compact and useful for purposes such as simulation, implication, synthesis, and others. We introduce a linear algebraic model analogous to the transfer function model used in other areas of engineering. The transfer function captures the behavior of the network and can also be used to determine the network output response for a given set of input stimuli [1]. The transfer function model is formulated as a mapping of elements over vector spaces representing the function domain and range sets. Transfer functions are in the form of a matrix or linear transformation, thus, we use the terms "transfer function" and "transfer matrix" interchangeably.

Ternary switching functions are traditionally modeled with an algebraic structure consisting of a discrete set of logic constants $\{0, 1, 2\}$ and an appropriate set of operators. Logic networks represent implementations using electronic or some other technology with a corresponding input/output characteristic modeled by ternary switching functions. A ternary logic network is an interconnection of symbols or gates whose functionality is modeled by the operators within the algebraic structure. A common set of ternary logic network elements are the *MIN*, *MAX*, and *Literal Selection* gates whose corresponding functionalities provide for the specification of a functionally complete algebra with constants. For conciseness we refer to the *Literal Selection* gate as a J_i gate where i denotes the polarity of the selected literal, $i \in \{0, 1, 2\}$ as defined in [2].

Here we use three-dimensional vectors to represent ternary logic values instead of the integers $\{0, 1, 2\}$ and we model a ternary logic function or a ternary logic network by a characterizing transfer matrix. The transfer matrix transforms the logic network input vectors to corresponding output vectors. This linear algebraic model of a ternary logic network or function is an alternative to the more commonly employed switching model. While we do not propose that the rich set of results based on switching functions be abandoned in favor of the linear algebraic model, we do propose that this alternative model may be advantageous for certain ternary logic synthesis and analysis tasks such as those described in [3], [4]. As an example, we examine the use of the linear algebra approach when used for ternary logic simulation. We utilize a netlist as input where the term "netlist" is used in the commonly accepted definition of representing a structural interconnections of logic gates. Modern EDA tools parse netlists into intermediate graphical representations representing a structural logic circuit and the term netlist does not refer to any specific format.

The remainder of the paper is organized as follows. In Section 2, background and notation is introduced to support the derivations of the transfer matrices for a logic network. Section 3 contains the definitions and derivations of matrices that characterize a network and also includes descriptions of how the matrices can be obtained from a high-level switching function specification or through direct traversal of a logic

Table 1. Ternary Logic Constants

Integral	Bra-Vector	Row-Vector
0	$\langle 0\|$	$\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$
1	$\langle 1\|$	$\begin{bmatrix} 0 & 1 & 0 \end{bmatrix}$
2	$\langle 2\|$	$\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$

network representation such as a netlist. Section 3 also contains examples that demonstrate of how the matrices are used for computation of system responses. Concluding remarks are provided in Section 4.

2. Background and Notation

2.1. Dirac Notation

"Bra-ket" notation was originally devised as a concise representation of vectors and associated operations to aid in quantum mechanical system calculations [5]. Due to the conciseness of this notation, we employ its use in this paper. Column vectors are referred to as "kets" and the notation for column vector $\mathbf{x}$ is $|x\rangle$. Likewise, a row vector $\mathbf{y}$ is referred to as "bra-y" and is written as $\langle y|$. The inner (or "dot") product of two vectors $\mathbf{x}$ and $\mathbf{y}$ is written as $\langle x|y\rangle$ and the outer (or "Kronecker") product as $|x\rangle\langle y|$.

2.2. Ternary Logic Constants

Ternary logic switching models commonly use an integral encoding $\{0, 1, 2\}$ to represent logic values. In the model presented here, we represent ternary logic values as three-dimensional vectors, where each logic value j is represented by a row vector whose j^{th} component is unity-valued and all other remaining components are zero-valued. Table 1 shows the correspondence of the integral and vector logic values. The choice of using row vectors as models for ternary logic values is arbitrary and column-vectors could have been chosen. As will become apparent in a later section, the choice of row-vectors for ternary constants results in the logic network transfer function being expressed as a matrix that closely resembles the switching model truth table.

We use the notation $\mathbb{H}$ to represent a Hilbert vector space whose elements are three-dimensional row vectors. The 3^n-dimensional Hilbert space is denoted as $\mathbb{H}^n$ and may be constructed through use of the outer product operation, denoted by the $\otimes$ operator, as shown in Equation 1 where n is any natural number, $n \in \mathbb{N}$.

$$\mathbb{H}^n = \mathbb{H} \otimes \mathbb{H} \otimes \ldots \otimes \mathbb{H} = \bigotimes_{i=1}^{n} \mathbb{H} \qquad (1)$$

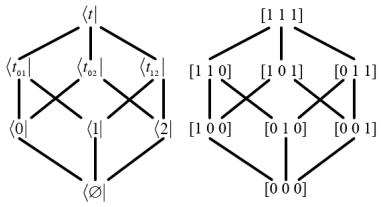

Figure 1. Hasse Diagrams of Ternary Vector Constants

In addition to the vector constants $\langle 0|$, $\langle 1|$, and $\langle 2|$, five other constants are defined and used, denoted as $\langle t|$, $\langle t_{01}|$, $\langle t_{02}|$, $\langle t_{12}|$, and $\langle \varnothing|$. $\langle t|$ represents the simultaneous presence of $\langle 0|$, $\langle 1|$, and $\langle 2|$ and is given by $\langle t| = \langle 0| + \langle 1| + \langle 2|$ where the + operator denotes vector space addition. Likewise, the constants $\langle t_{12}|$, $\langle t_{02}|$, and $\langle t_{01}|$ represent the simultaneous presence of two logic values $\langle t_{01}| = \langle 0| + \langle 1|$, $\langle t_{02}| = \langle 0| + \langle 2|$, and $\langle t_{12}| = \langle 1| + \langle 2|$. In contrast, $\langle \varnothing|$ represents the absence of all logic values and is given as $\langle \varnothing| = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$. $\langle \varnothing|$ should not be confused with the logic-0 value, $\langle 0| = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$. The collection of vector constants can be expressed as a Hasse diagram as shown in Figure 1 [2]. Two versions of the Hasse diagram are depicted using bra- and the row-vector notation. The five additional constants can arise during calculations using transfer function representations of logic networks.

2.3. Network Elements

Any of a variety of two- and one-place operators can be defined and represented by logic gates within a ternary logic network. For the algebraic construction to enable representation of all possible functions, some primitive set of operators is required such that all other possible operators can be expressed as a function of the primitives and constants [2]. In this work, we utilize the ternary MIN, MAX, and J_i operators as logic primitives.

3. Transfer Functions

The transfer function is an expression that yields the output response of the corresponding network when multiplied by an input stimulus. A particular input stimulus is represented by a row vector of dimension 3^n where n is the number of parallel network inputs. Individual network input values are used to compute the overall stimulus vector through the relationship in Equation 1. Lemma 3.1 describes a characteristic that is useful in the derivation of the transfer function.

Lemma 3.1: Linear Independence: Consider a ternary logic network with n inputs. Two distinct network input assignments $(\langle x_i|, \langle x_j|) \in \mathbb{H}^n$ are linearly independent vectors when $i \neq j$.

Proof: $\langle x_i|$ and $\langle x_j|$ are represented as $\langle d_n d_{n-1} \dots d_0|$, where each $d_i \in \{0, 1\}$ represents a particular network input logic value. Using the relation in Equation 1, the input vectors are expanded as $\langle d_n| \otimes \langle d_{n-1}| \otimes \dots \otimes \langle d_0|$. Expressed as a row vector, $\langle x| = \begin{bmatrix} 0 & 0 & \dots & 1 & \dots & 0 \end{bmatrix}$ where the single unity-valued vector component exists in a different location for $\langle x_i|$ and $\langle x_j|$ since $i \neq j$. Therefore $\langle x_i|x_j\rangle = 0$ and the norm of x_i and x_j is $L_2(x_i) = L_2(x_j) = 1$, satisfying the definition of linear independence. When $i = j$, $\langle x_i|x_j\rangle = 1$ and this only occurs when $\langle x_i| = \langle x_j|$. Expressed mathematically,

$$\mathbf{x}_i \cdot \mathbf{x}_j = \langle x_i|x_j\rangle = \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases} \tag{2}$$

□

Lemma 3.2: Input-Output Response: The output response of a logic network due to a particular input assignment $\langle x_i|$ is represented by $\langle f_i|$. The output response vector is obtained as the product of $\langle x_i|$ with $|x_i\rangle\langle f_i|$.

Proof: The lemma statement is expressed as
$(\langle x_i|)(|x_i\rangle\langle f_i|) = \langle x_i|x_i\rangle\langle f_i|$.
From Lemma 3.1, $\langle x_i|x_i\rangle = 1$, thus
$(\langle x_i|)(|x_i\rangle\langle f_i|) = (1)\langle f_i| = \langle f_i|$. □

Theorem 3.3: Transfer Function: The transfer function representing the input-output relationship of a logic network f is of the form of a matrix $\mathbf{T}$ and is given by Equation 3.

$$\mathbf{T} = \sum_{i=1}^{3^n} |x_i\rangle\langle f_i| \tag{3}$$

Proof: By definition, the transfer function yields the logic network output response vector $\langle f_i|$ when multiplied with the corresponding logic network input stimulus $\langle x_i|$. Multiplying $\mathbf{T}$ with the j^{th} logic network stimulus vector $\langle x_j|$ yields

$$\langle x_j|\mathbf{T} = \langle x_j| \left(\sum_{i=1}^{3^n} |x_i\rangle\langle f_i| \right) = \sum_{i=1}^{3^n} \langle x_j|x_i\rangle\langle f_i|$$

Using Equation 2,
$\langle x_j|\mathbf{T} = \langle f_j|$ □

3.1. Transfer Matrix Derivation

The transfer matrix as derived in the previous section allows for a means to derive the system response for one or more logic network input vectors through a single vector-matrix multiplication operation. Modern Electronic Design Automation (EDA) tools generally implement this operation through the use of discrete event simulation algorithms in conjunction with a description of the logic network. For the transfer matrix approach to be a practical alternative to EDA approaches, the transfer matrix $\mathbf{T}$ must be obtained in an efficient manner.

The transfer matrix $\mathbf{T}$ can be derived in several ways. We focus on two approaches here; derivation of $\mathbf{T}$ when the switching function behavior is given, and alternatively, when a low-level characterization of the network is provided in the form of a logic network or netlist.

3.1.1. Transfer Functions from Switching Models.

For switching function specifications, common representations include cube lists [6] or graphical representations such as the Binary Decision Diagrams (BDD) [7] or their generalization as Multiple-valued Decision Diagrams (MDD) [8]. These various descriptions allow specific corresponding input and output responses to be directly obtained.

A direct approach for the calculation of $\mathbf{T}$ is to form the outer product terms expressed in Equation 3 and sum them together. Unfortunately, this approach requires the formulation of an exponential number (3^n) matrices and then finding their sum. A better approach is to use the existing switching function representation as an implicit representation of the transfer matrix $\mathbf{T}$. This approach is viable and allows for system response calculations to be accomplished through the use of algorithms that implement the vector-matrix product operation using the structure of the implicitly represented transfer matrix $\mathbf{T}$.

Observation 3.4: Transfer Function/Truth Table Isomorphism: A truth table specification of a ternary switching function is isomorphic to the corresponding transfer function. □

To further illustrate Observation 3.4, consider a switching function representation comprised of a single *MIN* operation, $f = x \cdot y$. Using the direct approach for evaluating the transfer function of a *MIN* function transfer matrix, denoted as $\mathbf{A}$, results in the following calculation.

$$\begin{aligned} \mathbf{A} = {} & |00\rangle\langle 0| + |01\rangle\langle 0| + |02\rangle\langle 0| + |10\rangle\langle 0| \\ & + |11\rangle\langle 1| + |12\rangle\langle 1| + |20\rangle\langle 0| + |21\rangle\langle 1| \\ & + |22\rangle\langle 2| \end{aligned}$$

Expanding the outer product terms into 3×9 matrices and summing together yields

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

Examination of the structure of $\mathbf{T}_{MIN}$ reveals that each row vector is simply the vector representation of the *MIN* truth table output column. Thus, a truth table representation can be used as an implicit representation of the transfer matrix and algorithms can be formulated that implement the vector-matrix product operation to compute the output response. Since the transfer matrix representation is isomorphic to the truth table, decision diagram (DD) structures are sufficient to represent the transfer matrix such as those described in [8]. Furthermore, the matrix operations can be efficiently implemented using DDs as described in [9]. Thus, the complexity of transfer matrix representation is identical to that of using DDs.

3.1.2. Transfer Functions from Netlists. Many modern EDA analysis tasks require the derivation of a network response using only a low-level description such as a netlist. Methods requiring a switching behavioral model can prove problematic since the conversion of a netlist representation to a switching model representation can incur unacceptable complexity. Even when state-of-the-art representations such decision diagrams are used, parsing the netlist into such a structure can require excessive amounts of memory. A commonly cited example is a BDD representation of a fixed-point multiplication circuit. It has been proven that BDD representations of multiplier circuits grow exponentially with operand wordsize regardless of the variable orderings.

Another commonly encountered analysis task requires analysis of an internal partition or subcircuit within a netlist. Extraction of the switching model for an internal subcircuit can also be difficult in some cases. For these reasons, there is a need to devise an efficient means of determining netlist responses to various stimuli and motivates us to determine a means to efficiently extract the corresponding transfer function directly from a netlist representation while avoiding the intermediate step of first extracting a switching model, and then determining the transfer matrix.

The transfer matrix can be determined directly from a netlist representation of a logic network through two

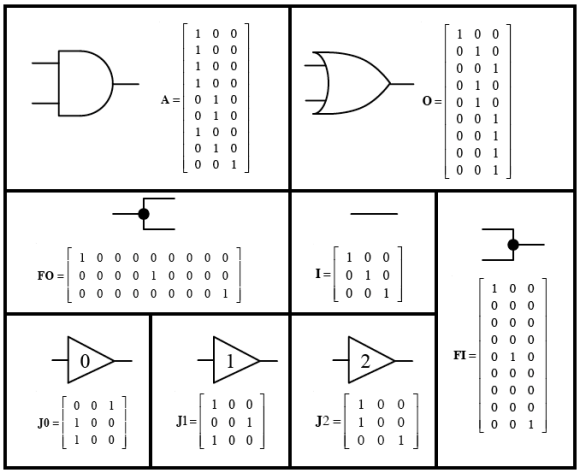

Figure 2. Ternary Network Element Transfer Matrices

traversals. In the first traversal, the netlist is partitioned into a series of cascaded stages. The second traversal requires computation of the transfer matrix for each stage and the final step derives the overall transfer matrix as the direct matrix product of each cascade stage transfer matrix. The partitioning step determines cuts through the netlist such that all elements in each partition operate in parallel. Each partition is then treated as a separate netlist whose inputs and outputs are combined using the outer product operation in accordance with Equation 1 where individual three-dimensional vectors are combined into a single 3^n-dimension vector. Because the partitions are chosen as serial cascade stages, the output of the prior cascade serves as the input to the subsequent stage, and the overall transfer matrix is simply the direct product of the partition matrices.

This technique can be implemented through use of a library of transfer matrices for the atomic logic network elements such as the *MIN*, *MAX*, and J_i gates. Additional atomic transfer matrices are needed in the library that correspond to the single line (pass-through), fanout, and fanin structures since they will appear as parallel elements in some of the network partitions. Figure 2 contains the transfer matrices for a library composed of these elements. The transfer matrix for the fanin gate **FI** has $\varnothing$ components for the cases where the two inputs are different logic values.

Example 3.5: Transfer Matrix from Netlist: Consider the logic network in Figure 3 with input stimulus of $\langle x_1| = \langle 1|$ and $\langle x_2| = \langle 2|$. The uppermost diagram shows the partitioned cascaded stages denoted by ϕ_1, ϕ_2, and ϕ_3. Each partition is comprised of a set of

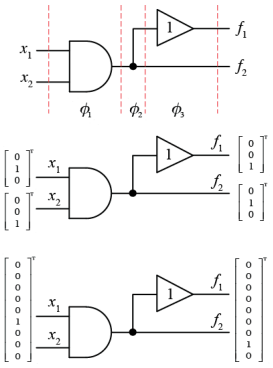

Figure 3. Partitioned Ternary Logic Network

parallel network elements and can be characterized by individual transfer matrices $\mathbf{T}_{\phi_1}$, $\mathbf{T}_{\phi_2}$, and $\mathbf{T}_{\phi_3}$. The logic network diagram in the center of Figure 3 depicts a specific input stimulus written as individual vectors. From Equation 1, the network input values can be combined into a single vector as shown on the bottom-most diagram. The corresponding network output response is shown on the right side of the network diagrams, both as individual vectors and as a combined vector formed as the outer product of the individual vectors.

The output of ϕ_1 is produced by a *MIN* gate, therefore $\mathbf{T}_{\phi_1} = \mathbf{A}$. Stage ϕ_2 is comprised of a single fanout network element and has a transfer matrix $\mathbf{T}_{\phi_2} = \mathbf{FO}$. Stage ϕ_3 is comprised of two network elements; a J_1 literal selection gate and a single pass through conductor. Just as the input vectors are combined into a single vector using the outer product, so is the overall transfer matrix $\mathbf{T}_{\phi_3} = \mathbf{J_1} \otimes \mathbf{I}$.

$$\mathbf{T}_{\phi_3} = \mathbf{J_1} \otimes \mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

□

3.2. Switching Domain Response

As implied in Theorem 3.3, the transfer matrix can be used to compute the output response of the characterized logic network through multiplication with a network stimulus vector.

Theorem 3.6: Network Response: The output response $\langle f_j|$ of a logic network characterized by transfer matrix $\mathbf{T}$ can be calculated by multiplying the input stimulus vector $\langle x_j|$ with $\mathbf{T}$ as expressed in Equation 4.

$$\langle f_j| = \langle x_j|\mathbf{T} \tag{4}$$

Proof: Multiplying Equation 3 with input stimulus $\langle x_j|$

$$\langle x_j|\mathbf{T} = \langle x_j| \left(\sum_{i=1}^{3^n} |x_i\rangle\langle f_i| \right)$$

$$= \sum_{i=1}^{3^n} \langle x_j|x_i\rangle|f_i\rangle$$

From Lemma 3.1, the inner product $\langle x_j|x_i\rangle = 0$ for the $3^n - 1$ cases in Equation 4 where $i \neq j$. Furthermore, $\langle x_j|x_i\rangle = 1$ for the single case in Equation 4 where $i = j$. When $i = j$, $\langle f_i| = \langle f_j|$, hence $\langle x_j|\mathbf{T} = \langle \mathbf{f_j}|$. □

Corollary 3.7: Multiple system Response: The system response due to multiple logic network input vectors can be calculated through a single multiplication operation by forming the input stimulus vector using the logic value $\langle t|$ and using the transfer matrix $\mathbf{T}$.

Proof: Three different input stimuli can be expressed in a single combined logic network input vector, $\langle x_{comb}|$ by specifying one of the network inputs as $\langle t|$ resulting in that particular input having logic values $\langle 0|$, $\langle 1|$, and $\langle 2|$ simultaneously. By definition of the logic value $\langle t|$, the combined input vector can be expanded as $\langle x_{comb}| = \langle x_0| + \langle x_1| + \langle x_2|$ where each $\langle x_i|$ represents a logic input vector with one input assigned logic value i. Multiplying the input vector $\langle x_{comb}|$ with $\mathbf{T}$ results in the logic network response $\langle f_0| + \langle f_1| + \langle f_2|$ where $\langle f_i|$ denotes the network response for input stimulus $\langle x_i|$. This result can be generalized by setting more than one input to value $\langle t|$. Furthermore, it is desired to restrict one of the inputs to simultaneously be a subset of logic values $\{0, 1, 2\}$, a new value akin to $\langle t|$ can be defined and used. □

Example 3.8: Calculating Single Network Response: The logic network depicted in Figure 3 is partitioned into three stages and the overall transfer matrix, $\mathbf{T}$, is computed as $\mathbf{T} = \mathbf{T}_{\phi_3}\mathbf{T}_{\phi_2}\mathbf{T}_{\phi_1}$.

$$\mathbf{T} = \mathbf{T}_{\phi_3}\mathbf{T}_{\phi_2}\mathbf{T}_{\phi_1} = (\mathbf{A})(\mathbf{FO})(\mathbf{J1} \otimes \mathbf{I})$$

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

To determine the logic network response $\langle f_{12}|$ for an input stimulus of $\langle x_1 x_2| = \langle 12|$, the network inputs are expressed as a single vector, $\langle 12| = \langle 1| \otimes \langle 2|$ and is multiplied with the transfer matrix $\mathbf{T}$ in accordance with Theorem 3.6.

$$
\begin{aligned}
&\langle f_{12}| = \langle 12|\mathbf{T} \\
&= \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}^T \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \\
&= \langle 2| \otimes \langle 1| = \langle 21| \qquad \square
\end{aligned}
$$

Here, we arbitrarily choose to perform outer product operations from the topmost signal to the bottommost when forming the transfer matrices for partitions and to perform the direct matrix multiplication operations using the transfer matrix closest to the network inputs as the leftmost operand. The network response to multiple input stimuli can be computed with a single evaluation of the transfer matrix through use of the $\langle t|$ value where $\langle t| = \langle 1| + \langle 1| + \langle 2|$ as proven in Corollary 3.7.

Example 3.9: Calculating Multiple Network Responses: The total network response due to all possible network input values $\langle f_{tot}|$ is calculated as

$$
\begin{aligned}
&\langle f_{tot}| = \langle tt|\mathbf{T} \\
&= \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}^T \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 5 & 0 & 1 & 0 & 0 & 0 & 0 & 3 & 0 \end{bmatrix} \\
&= 5 \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\
&+ \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\
&+ 3 \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \\
&= 5(\begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}) \\
&+ (\begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \otimes \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}) \\
&+ 3(\begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}) \\
&= 5\langle 00| + \langle 02| + 3\langle 21| \qquad \square
\end{aligned}
$$

The result of Example 3.9 indicates that only three different network responses are possible and that $\langle 00|$ occurs five times, $\langle 02|$ occurs once, and $\langle 21|$ occurs three times. Instead of computing the total system response, this technique can be used to find partial system responses through the use of input stimuli vectors with a subset of inputs assigned the values $\langle t|$, $\langle t_{01}|$, $\langle t_{02}|$, or $\langle t_{12}|$.

4. Conclusion

A model of a ternary switching function as a transformation over vector spaces is presented. The characteristic transfer matrix for a particular switching function is introduced and methods for obtaining it from either a switching function specification or a netlist are provided. Inclusion of additional constants that cover more than one logic value are described and used to obtain multiple system responses through a single calculation with the transfer matrix.

References

[1] C. T. Chen, *Linear System Theory and Design*. Holt, Rinehart and Winston, 1984.

[2] D. M. Miller and M. A. Thornton, *Multiple Valued Logic: Concepts and Representations*. Morgan & Claypool Publishers, 2007.

[3] M. A. Thornton and J. Dworak, "Ternary logic network justification using transfer matrices," *Proceedings. IEEE International Symposium on Multiple-Valued Logic*, 2013.

[4] M. A. Thornton and T. W. Manikas, "Spectral response of ternary logic netlists," *Proceedings. IEEE International Symposium on Multiple-Valued Logic*, 2013.

[5] P. A. M. Dirac, "A new notation for quantum mechanics," *Proc. of the Cambridge Philosophical Society*, vol. 54, p. 416, 1939.

[6] T. Sasao, *Switching Theory for Logic Synthesis*. Kluwer Academic Publishers, 1999.

[7] R. E. Bryant, "Graph-Based algorithms for boolean function manipulation," *IEEE Trans. Comput.*, vol. C-35, no. 8, pp. 677 –691, Aug. 1986.

[8] D. M. Miller and R. Drechsler, "Implementing a multiple-valued decision diagram package," in *Proceedings. 1998 28th IEEE International Symposium on Multiple-Valued Logic*, May 1998, pp. 52 –57.

[9] E. M. Clarke, M. Fujita, P. C. McGeer, K. McMillan, J. C. Yang, and X. Zhao, "Multi-terminal binary decision diagrams: An efficient data structure for matrix representation," in *Proceedings. IEEE Int. Workshop on Logic Synthesis*, 1993, pp. 1–15.