Ternary Logic Network Justification Using Transfer Matrices

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

Abstract

A linear algebraic method is developed that allows for logic network justification problems to be solved. The method differs from previous techniques that require learning or solution space search techniques in that all possible justification solutions are determined through a single vector-matrix product calculation. The logic network is represented by a matrix that is defined as the "justification" matrix. It is shown that the justification matrix is simply the transpose of the network transfer matrix and is thus easily obtained through a traversal of the network netlist. Example justification calculations are provided.

1. Introduction

Many logic network design and analysis tasks involve the determination of a set of input stimuli b, given a network model and an output response a. In other words, we must *justify* the values assigned to the outputs by finding an appropriate set of assignments to the inputs. Several justification algorithms and heuristics have been proposed previously. Here, we describe an algebraic model representation for a ternary logic network and use it to find a solution to the justification problem. Specifically, given the set of logic values a and a representation of a ternary logic network in the form of a transfer matrix $\mathbf{T}$, we find the corresponding network input values b that produce the desired output values a.

Logic network justification is useful in multiple design and analysis applications, including synthesis, verification, and test. For example, justification is a critical component of automatic test pattern generation (ATPG) algorithms, including the D-algorithm [1], PODEM [2], and FAN [3]. It is also inherently related to the satisfiability problem (SAT). Furthermore, reverse logic implications are a special case where *all possible* solutions require that a particular logic value (or set of logic values) be assigned to a particular input (or set of inputs).

Justification during ATPG is historically performed by successively assigning logic values to upstream gates or primary inputs and then propagating the results downstream through forward implication until the desired downstream assignment has been justified. However, reconvergent fanout may produce conflicts among selected value assignments. This requires that previously assigned nodes be assigned a different value and thus requires *backtracking*. Similarly, a widely applied technique for finding implications involves the construction of a binary learning tree whose traversal, in conjunction with a recursive learning algorithm, allows assignment of logic values at various logic network nodes [4]. In the learning approach, a backtracking algorithm is also used.

Unlike many previous approaches, the approach described here allows multiple justification solutions to be determined simultaneously through a single computation without recursion or backtracking. It is based on a transfer matrix representation of the logic network introduced in [5]. In that paper, a ternary logic network or behavioral description was represented by a characterizing matrix, and the input stimuli and corresponding output response were represented as vectors. This representation allows the input and output vectors to be related as a linear transform, where the characterizing matrix represents the input-output behavior of the logic system and is thus a "transfer function" model [6]. Because the transfer function is in the form of a matrix, we refer to the representation as either a "transfer function" or equivalently a "transfer matrix."

In this paper, we define the "justification matrix" and show it to be equivalent to the *transpose* of the transfer matrix. Thus, efficient methods for determination of the transfer matrix will suffice for efficiently determining the justification matrix.

The remainder of the paper is organized as follows.

In Section 2, we briefly review the notation and concepts from [5] used to describe the transfer function model for ternary logic systems. Section 3 expands upon the transfer function model and introduces the notion of a "justification matrix." Section 4 contains an example of using a justification matrix and illustrates how multiple solutions to the justification problem may be determined with a single computation. Conclusions are provided in Section 5.

2. Transfer Matrix Models

A transfer matrix model for ternary logic systems is introduced and described in detail in [5]. Here, we provide an overview of the most important concepts as background for the development and use of the "justification matrix."

Ternary switching functions are traditionally modeled with an algebraic structure consisting of a discrete set of logic constants $\{0, 1, 2\}$ and a set of switching operators such as *MIN*, *MAX*, and *Literal Selection* [7]. Ternary logic systems may be modeled in a variety of ways, including systems of switching logic equations over an algebra such as the Post algebra, as truth tables, as decision diagrams, or as netlists. In contrast, the transfer matrix model captures the input-output behavior of a ternary logic network in the form of a characteristic matrix $\mathbf{T}$ that defines a linear transformation of a vector representing an input assignment to a corresponding vector representing the function value. The transfer matrix approach models the behavior of a ternary logic system within a linear algebraic framework as opposed to a more commonly used switching algebra such as the Post algebra.

To characterize a switching network in a linear algebraic framework, logic value assignments are represented by vector quantities. We use the vector notation introduced by Dirac [8] and map the commonly used scalar logic values $\{0, 1, 2\}$ to corresponding three-dimensional vectors $\{\langle 0|, \langle 1|, \langle 2|\}$. The components of the vector are the scalars $\{0, 1\}$ and are explicitly defined as $\langle 0| = [1, 0, 0]$, $\langle 1| = [0, 1, 0]$, and $\langle 2| = [0, 0, 1]$. We also use the notion of a null vector $\langle \varnothing| = [0, 0, 0]$ representing the absence of $\langle 0|$, $\langle 1|$, and $\langle 2|$ and the vectors $\langle t_{01}| = [1, 1, 0]$, $\langle t_{02}| = [1, 0, 1]$, $\langle t_{12}| = [0, 1, 1]$, and $\langle t| = [1, 1, 1]$ that represent covers of more than one logic value. The complete set of logic values then forms a lattice as depicted in the Hasse diagram in Fig. 1

The Kronecker or outer product operation, denoted by $\otimes$, is used to combine individual logic vector values into a single vector. Because the outer product operation is non-commutative, the order in which the logic values are combined is important for the sake of consistency within the computations, but the particular order is arbitrary. As an example, two logic values $\langle 2|$ and $\langle 1|$ may be represented as the single vector $\langle 2| \otimes \langle 1| = [0, 0, 1] \otimes [0, 1, 0] = [0, 0, 0, 0, 0, 0, 0, 1, 0]$.

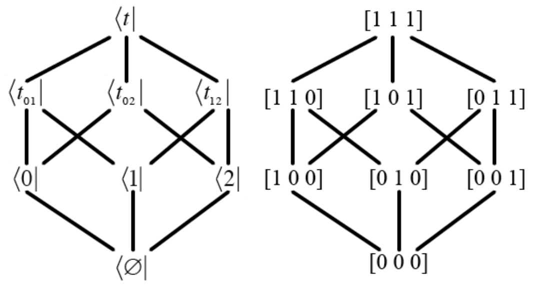

Figure 1. Hasse Diagrams of Ternary Vector Constants

As described in [5], the transfer matrix $\mathbf{T}$ of a ternary logic system represents all possible input-output behavior of the single vectors representing a variable assignment and their corresponding single output vectors. The matrix $\mathbf{T}$ may be formed as the matrix sum of all outer products of the input-output vector pairs as given in Equation 1 where $|x_i\rangle$ represents the i^{th} variable assignment resulting in the i^{th} function value $\langle f_i|$.

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

The explicit use of Equation 1 requires a summation of 3^n matrices for an n-variable function. Fortunately, the transfer matrices $\mathbf{T}$ are isomorphic to the function truth table, thus any compact representation of the truth table such as decision diagrams [9] [10] may be used to represent the transfer matrix. As shown in [5], it is also possible to efficiently determine the transfer matrix through a traversal of a logic network representing the ternary switching function.

Using these formulations, a method for computing the output response of a ternary logic network given an input stimulus was derived in [5]. Although the method is based on ternary logic systems, it is easily extended to systems of other logic values including binary and systems of higher radix $p > 3$. Furthermore, the use of covering logic values $\langle t_{01}|$, $\langle t_{02}|$, $\langle t_{12}|$, and $\langle t|$ enable multiple stimuli represented by the vector $\langle x|$ to be specified and the resulting set of of output responses $\langle f|$ obtained through a single vector-matrix product calculation as given in Equation 2.

$$\langle f| = \langle x|\mathbf{T} \tag{2}$$

3. Justification Matrices

Justification can be viewed as the inverse problem of output response determination. Therefore, given an output response vector $\langle f|$ and a specification of the logic network $\mathbf{T}$, a solution $\langle x|$ that produces the desired output response can be computed as given in Equation 3.

$$\langle x| = \langle f|\mathbf{T}^{-1} \tag{3}$$

In general the number of logic network inputs n is not equal to the number of outputs m. In the general case where $n \neq m$, the transfer matrix $\mathbf{T}$ is not square, and a unique inverse $\mathbf{T}^{-1}$ does not exist. Furthermore, in many cases where $n = m$, the corresponding transfer matrix $\mathbf{T}$ is not of full rank and unique $\mathbf{T}^{-1}$ matrices do not exist in this case as well. For those cases where $\mathbf{T}$ is square and of full rank, the corresponding logic network is said to be logically reversible and $\mathbf{T}$ is a bijective linear transformation. Based on these observations, a logic network characterized by transfer matrix $\mathbf{T}$ can be classified according to four possibilities.

1) $n = m$ and $\mathbf{T}$ is of full rank. A unique inverse $\mathbf{T}^{-1}$ exists and the logic network is logically reversible.
2) $n = m$ and $\mathbf{T}$ is not of full rank. A unique inverse $\mathbf{T}^{-1}$ does not exist and the corresponding linear system of equations in Equation 3 is either over- or under-specified.
3) $n > m$, the logic network has more inputs than outputs and the $\mathbf{T}$ matrix represents an over-specified system, thus multiple solutions exist for Equation 3.
4) $n < m$, the logic network has more outputs than inputs and the $\mathbf{T}$ matrix represents an under-specified system, thus a unique solution is not possible.

Here, we are concerned with the general case where $\mathbf{T}$ is not of full rank hence, Equation 3 cannot be used since a unique $\mathbf{T}^{-1}$ does not exist. There are various techniques available for the general solution of Equation 3 when the matrix $\mathbf{T}$ is non-square or when it is not of full rank based on the use of generalized inverses [11]. In [12], the Moore-Penrose pseudo-inverse is described as a means for computing $\mathbf{T}^{-1}$ for binary logic net lists. We use the pseudo-inverse in Equation 3 for the purpose of computing the justification of a ternary logic network.

a	*b*	*MIN(a,b)*
0	0	0
0	1	0
0	2	0
1	0	0
1	1	1
1	2	1
2	0	0
2	1	1
2	2	2

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

Figure 2. *MIN* Gate Truth Table and Transfer Matrix

For over specified systems, the pseudo-inverse can be used to provide a solution that has a minimal or least squared error and for under-specified systems a best-fit solution is provided with minimal least squared error. In the case of justification solutions for logic networks, a single satisfying solution obtained through use of the pseudo-inverse is acceptable.

The pseudo-inverse of a matrix $\mathbf{T}$ is denoted as $\mathbf{T}^{+}$ and is computed in two different manners depending upon whether the system is over- or under-specified. The pseudo-inverse $\mathbf{T}^{+}$ is calculated according to Equation 4 and depends upon the Hermitian transpose $\mathbf{T}^{*}$.

$$\mathbf{T}^{+} = \begin{cases} (\mathbf{T}^{*}\mathbf{T})^{-1}\mathbf{T}^{*} \\ \mathbf{T}^{*}(\mathbf{T}^{*}\mathbf{T})^{-1} \end{cases} \tag{4}$$

The transfer matrix elements characterizing ternary logic networks are always real valued, thus the Hermitian transpose is simply the transpose of $\mathbf{T}$, thus $\mathbf{T}^{*} = \mathbf{T}^{T}$. Using this observation, Equation 4 can be rewritten as given in Equation 5.

$$\mathbf{T}^{+} = \begin{cases} (\mathbf{T}^{T}\mathbf{T})^{-1}\mathbf{T}^{T} \\ \mathbf{T}^{T}(\mathbf{T}^{T}\mathbf{T})^{-1} \end{cases} \tag{5}$$

Equations 2 and 5 can be used as the basis for solving a justification problem. Example 3.1 illustrates how Equation 5 can be used to find the input stimulus vector $\langle x|$ given an output response $\langle f| = \langle 1|$ for a simple logic network consisting of a two-input *MIN* gate.

Example 3.1: A two-input *MIN* gate can be characterized by a truth table and corresponding isomorphic transfer matrix $\mathbf{A}$ as shown in Figure 2.

Using Equation 5, the pseudo-inverse $\mathbf{T}^{+}$ becomes:

$$\mathbf{A}^{+} = \begin{bmatrix} \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 & 0 & \frac{1}{5} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Given that an output response of the *MIN*-gate is $\langle 1| = [0,1,0]$, the corresponding input stimulus is computed as $\langle x| = \langle f|\mathbf{A}^{+}$:

$$\langle x| = [0,1,0]\begin{bmatrix} \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 & 0 & \frac{1}{5} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$
$$= \tfrac{1}{3}[0,0,0,0,1,1,0,1,0] = \tfrac{1}{3}\langle 11| + \tfrac{1}{3}\langle 12| + \tfrac{1}{3}\langle 21|$$

Thus, it is observed that three different input stimuli can result in $\langle f| = \langle 1|$ and that those input stimuli are $\langle 11|$, $\langle 12|$, $\langle 21|$. □

The result of the computation in Example 3.1 is an output response vector composed of nine elements. To determine the individual input stimuli, the output vector must be factored. Factoring of outer products can, in general, be complex; however due to the fact that input vector components are restricted to $\{0,1\}$, the factors can be easily obtained by observing the indices of the non-zero components and expressing the indices in radix-3. The non-zero output response elements occur at vector indices $(4)_{10} = (11)_3$, $(5)_{10} = (12)_3$, and $(7)_{10} = (21)_3$.

To illustrate the use of other values within the Hasse diagram in Figure 1, consider Example 3.2.

Example 3.2: Given that an output response of the *MIN*-gate is either at logic-0 or at logic-1 ($\langle f| = \langle t_{01}| = [1,1,0]$), the corresponding input stimulus is computed as $\langle x| = \langle f|\mathbf{A}^{+}$:

$$\langle x| = [1,1,0]\begin{bmatrix} \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 & 0 & \frac{1}{5} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{3} & \frac{1}{3} & \frac{1}{5} & \frac{1}{3} & 0 \end{bmatrix}$$
$$= \tfrac{1}{5}\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$
$$+ \tfrac{1}{3}\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$
$$= \tfrac{1}{5}(\langle 00| + \langle 01| + \langle 02| + \langle 10| + \langle 20|)$$
$$+ \tfrac{1}{3}(\langle 11| + \langle 12| + \langle 21|)$$ □

The use of Equation 5 allows for the $\mathbf{T}^{+}$ matrix to be computed and used for computation of the input stimulus vector $\langle x|$ based upon a logic network response vector. Theorem 3.5 provides a result concerning the structure of the pseudo-inverse matrices.

Definition 3.3: Output Matrix: The 3×3 matrix $\mathbf{F}_i$ is defined as the "output matrix" representation of a ternary logic network or the "output matrix" response of a ternary logic function resulting from an input stimulus or variable assignment $\langle x_i|$.

$\mathbf{F_i} = |f_i\rangle\langle f_i|$

Lemma 3.4: Output Matrix Structure The output matrix of a ternary function $\mathbf{F}_i$ in response to a single variable assignment $\langle x_i|$ is of the form of a 3×3 matrix consisting of row vectors that are either $\langle\varnothing|$, $\langle 0|$, $\langle 1|$, or $\langle 2|$. Furthermore, only a single row vector $\langle 0|$, $\langle 1|$, or $\langle 2|$ will be present in the output matrix and the row vector $\langle i|$ will only be present in output matrix row i, the other two rows will contain $\langle\varnothing|$. □

Theorem 3.5: Pseudo-inverse Matrix Structure: The pseudo-inverse of a transfer matrix representing a ternary logic network is proportional to its transpose with a proportionality constant $\mathbf{P}$ where $\mathbf{P}$ is a 3×3 diagonal matrix.

Proof: Equation 5 contains a term $\mathbf{T}^T\mathbf{T}$ resulting in a 3×3 matrix of the following form:

$$\mathbf{T}^T\mathbf{T} = \begin{bmatrix} N_0 & 0 & 0 \\ 0 & N_1 & 0 \\ 0 & 0 & N_2 \end{bmatrix}$$

Where N_i is the integral number of times that the ternary function represented by transfer matrix $\mathbf{T}$ evaluates to one of the logic values $\{0,1,2\}$. We denote $\mathbf{T}^T\mathbf{T} = \mathbf{P}^{-1}$. From the isomorphic relation between the function truth table and the corresponding transfer matrix $\mathbf{T}$, it is observed that $\mathbf{P}^{-1}$ can be expressed as the summation of all output matrices.

$$\mathbf{P}^{-1} = \mathbf{T}^T\mathbf{T} = \sum_{n=1}^{3^n} |f_i\rangle\langle f_i|$$

Due to the structure of the output matrices, $\mathbf{P}^{-1}$ is diagonal and its inverse is of the form:

$$\mathbf{P} = (\mathbf{T}^T\mathbf{T})^{-1} = \begin{bmatrix} \frac{1}{N_0} & 0 & 0 \\ 0 & \frac{1}{N_1} & 0 \\ 0 & 0 & \frac{1}{N_2} \end{bmatrix}$$

From the form of Equation 5 and $\mathbf{P}$, the pseudo-inverse of the transfer matrix $\mathbf{T}$ is:

$$\mathbf{T}^{+} = \begin{cases} \mathbf{P}\mathbf{T}^T \\ \mathbf{T}^T\mathbf{P} \end{cases}$$ □

The result of Theorem 3.5 allows us to define the "justification matrix" $\mathbf{T}^J$. The justification matrix can be used to obtain the input stimulus vector of a ternary logic network or function given the corresponding output vector by evaluating Equation 6.

$$\langle x| = \langle f|\mathbf{T}^J \tag{6}$$

Equation 6 can be used for determining the input stimulus $\langle x|$ in the same way as that of Equation 3 with $\mathbf{T}^J$ used in place of $\mathbf{T}^{-1}$ with the result differing only in that the fractional constants $\frac{1}{N_0}$, $\frac{1}{N_1}$, and $\frac{1}{N_2}$ are omitted from the result. The advantage of using $\mathbf{T}^J$ to compute the justification solution is that it is easily obtained since it is the transpose of the transfer matrix $\mathbf{T}$. Therefore, the methods described in [5] for deriving the transfer matrix of logic networks and ternary logic functions are sufficient for computing the justification matrix $\mathbf{T}^J$.

Definition 3.6: Justification Matrix The "justification matrix" $\mathbf{T}^J := \mathbf{T}^T$ is the transpose of the transfer matrix $\mathbf{T}$ representing a ternary logic network or ternary logic function. □

4. Example Justification Computations

Consider the example ternary truth table and logic network depicted in Figure 3. The logic network is partitioned into cascades of parallel components as shown by the vertical dashed lines. Each serial cascade stage is labelled as ϕ_3, ϕ_2, and ϕ_1 with corresponding transfer matrices $\mathbf{T}_{\phi_3}$, $\mathbf{T}_{\phi_2}$, and $\mathbf{T}_{\phi_1}$. $\mathbf{T}_{\phi_3}$ is the transfer matrix of a two-input *MAX* gate denoted as $\mathbf{O}$ in [5]. $\mathbf{T}_{\phi_2}$ is the transfer matrix representing a fanout point denoted as $\mathbf{FO}$ from [5]. Stage ϕ_1 is composed of a J_0 literal selection gate in parallel with a single conducting path, thus $\mathbf{T}_{\phi_1} = \mathbf{J0} \otimes \mathbf{I}$.

The calculation of the partial product $(\mathbf{T}_{\phi_3}\mathbf{T}_{\phi_2})$ is: $\mathbf{T}_{\phi_3}\mathbf{T}_{\phi_2} = (\mathbf{O})(\mathbf{FO})$

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

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

The calculation of $\mathbf{T}_{\phi_1}$ is: $\mathbf{T}_{\phi_1} = \mathbf{J0} \otimes \mathbf{I}$

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

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

The overall logic network transfer matrix is computed as the direct product of the cascade stage transfer

x_1	x_2	f_1	f_2
0	0	2	0
0	1	0	1
0	2	0	2
1	0	0	1
1	1	0	1
1	2	0	2
2	0	0	2
2	1	0	2
2	2	0	2

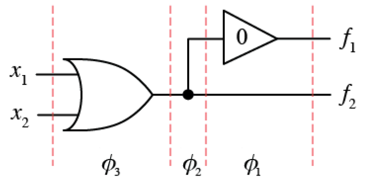

Figure 3. Example Truth Table and Logic Network

matrices, $\mathbf{T} = \mathbf{T}_{\phi_3}\mathbf{T}_{\phi_2}\mathbf{T}_{\phi_1} = (\mathbf{O})(\mathbf{FO})(\mathbf{JO} \otimes \mathbf{I})$ resulting in Equation 7.

$$\mathbf{T} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{7}$$

From Definition 3.6, the justification matrix corresponding to the network in Figure 3 is $\mathbf{T}^J = \mathbf{T}^T$ and is given in Equation 8.

$$\mathbf{T}^J = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{8}$$

The justification matrix $\mathbf{T}^J$ can be used to determine the input stimulus $\langle x|$ for a given output response $\langle f| = \langle 00|$ as shown in Example 4.1.

Example 4.1: Justification Calculation Example:
$\langle x_1 x_2| = \langle 00|\mathbf{T}^J = (\langle 0| \otimes \langle 0|)\mathbf{T}^J$
$= (\begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 \end{bmatrix})\mathbf{T}^J$
$= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{T}^J$
$= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
$= \langle \varnothing| \otimes \langle \varnothing| = \langle \varnothing\varnothing|$ □

Thus, the justification calculation in Example 4.1 indicates that there are no possible input values resulting in an output response of $\langle f| = \langle 00|$ as can be confirmed by examining the truth table in Figure 3.

Another example is to consider the solution of the justification $(\langle f_1 f_2| = \langle 01| \rightarrow \langle x_1 x_2| = \langle AB|)$. The solution of this justification problem is to determine the values A and B. Example 4.2 solves this justification problem using the justification matrix in Equation 7.

Example 4.2: Justification Calculation Example 2: To find the satisfying values A and B satisfying $(\langle f_1 f_2| = \langle 01| \rightarrow \langle x_1 x_2| = \langle AB|)$, Equation 6 is used with the justification matrix in Equation 8 as follows.

$$\langle x_1 x_2| = \langle 01|\mathbf{T}^J = (\langle 0| \otimes \langle 1|)\mathbf{T}^J$$
$$= (\begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 & 0 \end{bmatrix})\mathbf{T}^J$$
$$= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{T}^J$$
$$= \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$
$$= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$
$$+ \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$
$$= (\begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 & 0 \end{bmatrix})$$
$$+ (\begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 & 0 \end{bmatrix})$$
$$= (\langle 0| \otimes \langle 1|) + (\langle 1| \otimes \langle t_{01}|) = \langle 01| + \langle 1t_{01}| \qquad \square$$

The solution of the justification problem in Example 4.2 demonstrates how the partially covering logic value t_{01} is used. Because t_{01} indicates a value that is both simultaneously logic value 0 and 1, the justification solution covers three distinct input values; namely $\langle x_1 x_2| = \langle 01|$, $\langle x_1 x_2| = \langle 10|$, and $\langle x_1 x_2| = \langle 11|$. The resultant vector is first rewritten as a sum of two vectors and then decomposed. The single justification calculation of Example 4.2 results in all three possible cases of input stimuli that result in a logic network response of $\langle 01|$.

5. Conclusion and Future Effort

The transfer matrix representation $\mathbf{T}$ of a ternary logic network allows the network to be characterized by a matrix representing the input-output behavior. A new matrix $\mathbf{T}^J$ termed the "justification matrix" is introduced and is shown to be the transpose of the transfer matrix, $\mathbf{T}^J = \mathbf{T}^T$. The justification matrix can be used to determine the corresponding input stimulus of a ternary logic network given the function transfer matrix and an output response vector. Furthermore, the output response or input stimulus vectors can be formulated such that they contain logic values simultaneously representing more than one logic assignment through the use of $\langle t_{01}|$, $\langle t_{02}|$, $\langle t_{12}|$, and $\langle t|$.

The transfer matrix $\mathbf{T}$ and corresponding justification matrix $\mathbf{T}^J$ are isomorphic to the function truth table, thus any means for efficiently representing truth tables may also serve as representations of the corresponding transfer matrices. Furthermore, transfer and justification matrices for subcircuits or portions of logic networks may be computed directly by traversing a graphical representation of the network thus avoiding the need to first determine an entire network truth table. This result allows for an alternative means to solve the justification problem which has numerous applications in logic network design and analysis tasks.

References

[1] J. Roth, "Diagnosis of automata failures: A calculus and a method," *IBM Journal of Research and Development*, vol. 10, no. 4, pp. 278–291, 1966.

[2] P. Goel, "An implicit enumeration algorithm to generate tests for combinational logic circuits," *IEEE Trans. Comput.*, vol. C-30, no. 3, pp. 215–222, 1981.

[3] H. Fujiwara and T. Shimono, "On the acceleration of test generation algorithms," *IEEE Trans. Comput.*, vol. C-32, no. 12, pp. 1137–1144, 1983.

[4] W. Kunz and D. K. Pradhan, "Recursive learning: An attractive alternative to the decision tree for test generation in digital circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 13, no. 9, pp. 1143 –1158, Sep. 1994.

[5] M. A. Thornton, "A transfer function model for ternary switching logic circuits," in *Proc. of International Symposium on Multiple-Valued Logic*, May 2013.

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

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

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

[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.

[10] 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.

[11] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Johns Hopkins University Press, 1996.

[12] M. A. Thornton, "Spectral analysis of digital logic using circuit netlists," in *Proceedings. 2011 Conference on Computer Aided System Theory*, 2011, pp. 414–415.