Spectral Analysis of Digital Logic Using Circuit Netlists

Mitchell A. Thornton
Southern Methodist University
Dallas, Texas, USA

1.0 Introduction

Spectral analysis of digital logic switching circuits has been applied to areas such as synthesis, verification, and testing [HMM:05]. A challenge is the large complexity of the representative switching functions whose discrete range set contains an exponential number of values with respect to the size of the domain variable set. Methods such as decision diagrams and other fast transform techniques have been applied to overcome this problem [SA:03, TDM:01]. In this paper, we present a technique for spectral analysis that transforms the circuit directly instead of the underlying switching function.

Traditionally, digital logic circuits are modeled using the axioms and postulates of binary-valued Boolean algebra and discrete scalar-valued switching functions. In the approach discussed 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. Furthermore, the spectral transformations 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. By applying spectral transformations in this smaller grained fashion, problems associated with excessive memory usage can be avoided.

The remainder of the paper is organized as follows. First, we will review and introduce the vector space model of switching functions including their spectral transforms. Next, we will illustrate the application of these principles using a small example logic circuit. We will show how the problem of circuit implication maps to the use of the Moore-Penrose pseudoinverse and how spectral transforms are obtained. Finally, we conclude with observations of the approach and outline future related research topics.

2.0 Circuit Model Formulation

The Nobel Prize winning physicist Paul Dirac made many fundamental contributions to the field of quantum mechanics and in one of his well-known papers, he introduced a notation for quantum mechanical calculations that continues to be widely used today [Dir:39]. This notation, sometimes called 'bra-ket' notation is used to describe the state of a quantum system. Because the state of a quantum system is mathematically represented as a 1st-order tensor, or vector, bra-ket notation may also be used to denote vectors. We shall use bra-ket notation in this paper for vectors.

The column vector $\mathbf{a}$ 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 $\mathbf{a}$ written as $\|\mathbf{a}\|$ can be expressed in bra-ket notation as $\langle a|a\rangle$ and the inner product of two vectors $\mathbf{a}$ and $\mathbf{b}$ can be written as $\langle a|b\rangle$. The outer product, or tensor product of two 1st-order tensors $\mathbf{a}$ and $\mathbf{b}$ 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 column vectors $|0\rangle=[1\ 0]^T$ and $|1\rangle=[0\ 1]^T$. 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. As an example, the transfer matrix for a two-input AND gate is expressed as:

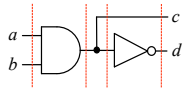
$$\mathbf{T}_{\text{AND}} = |0\rangle\langle 00| + |0\rangle\langle 01| + |0\rangle\langle 10| + |1\rangle\langle 11| = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The output response of a logic gate can then be determined as the product of the transfer matrix and the input logic vector. As an example, the output response of an AND gate when two logic 1-values are present at the inputs can be computed as:

$$\mathbf{T}_{\text{AND}}(|1\rangle \otimes |1\rangle) = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle$$

Transfer matrices can be easily 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. The following example illustrates the application of this principle and contains a fanout point that also must be modeled as a transfer matrix.



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

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

2.1 Logic Circuit Implication

Logic circuit implication is used in many synthesis, verification, and test algorithms. Implication can be formulated here by utilizing the Moore-Penrose pseudoinverse of the circuit transfer matrix. It is necessary to use the pseudoinverse since the transfer matrices are, in general, non-square. Due to the properties of logic gate transfer matrices, the following lemma holds.

Lemma: The psuedoinverse of a logic gate transfer matrix is equal to its' transpose. □

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

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

$$= |00\rangle + |01\rangle + |10\rangle$$

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

2.2 Logic Circuit Spectral Formulation

The spectral formulation of an AND gate can be modeled by using the transform of the transfer matrix. Although any desired linear transform can be used, we use the Fourier transform over GF(2) as an example here. In past work this transform is sometimes referred to as the Walsh or Hadamard transform, **H**. In keeping with convention, logic-0 is represented as +1 and logic-1 is -1. These values are used because they are equivalent to the two maximally-spaced square roots of unity along the real axis of the unit circle in the complex plane. The spectral transform of the AND gate, $\mathbf{W}_{\text{AND}}$, is calculated as:

$$\mathbf{W}_{\text{AND}} = \mathbf{H}\mathbf{T}_{\text{AND}} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} -1 & -1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 \\ -2 & -2 & -2 & 0 \end{bmatrix}$$

The spectral formulation of the implication matrix can be similarly computed as the transform of the pseudoinverse (transpose) of the transfer matrix.

3.0 Conclusions and Future Work

The formulation of combinational logic networks in the spectral domain leads to the notion of transfer matrices that can be used to calculate the input-output behavior analogous to transfer functions used in classical linear systems analysis. It is shown that the implication problem is conveniently accomplished in the vector space formulation by using the transpose of the logic gate transfer matrices. It is also shown that the spectral formulation of a logic netlist can be obtained by transforming the individual logic gate transfer matrices.

Past logic circuit spectral methods have mainly focused on the transforms of the underlying switching functions. The approach presented here allows the logic networks themselves to be directly represented in the spectral domain.

Future work will involve the development of logic circuit synthesis, verification, and test algorithms in the spectral domain using the spectral formulation of circuit netlists directly. The goal will be to determine methods in the spectral domain that are computationally beneficial as compared to currently implemented design automation methods that use netlists modeled with scalar Boolean algebra systems.

REFERENCES

[Dir:39] Dirac, P.A.M., "A new notation for quantum mechanics," *Proceedings of the Cambridge Philosophical Society*, vol. 35, p. 416, 1939.

[TDM:01] Thornton, M.A., Drechsler, R., and Miller, D.M., **Spectral Techniques in VLSI CAD**, Kluwer Academic Publishers, ISBN 0-7923-7433-9, July 2001.

[SA:03] Stankovic, R. and Astola, J., **Spectral Interpretation of Decision Diagrams**, Springer-Verlag, ISBN 0-387-95545-3, 2003.

[HMM:85]Hurst, S.L., Miller, D.M., and Muzio, J.C., **Spectral Techniques in Digital Logic**, Academic Press, 1985.