Spectral Decision Diagrams Using Graph Transformations*

Mitchell Thornton
Department of Electrical and Computer Engineering
Mississippi State University
Mississippi State, MS 39762
mitch@ece.msstate.edu

Rolf Drechsler
Siemens AG
Corporate Technology
81730 Munich, Germany
rolf.drechsler@mchp.siemens.de

Abstract

Spectral techniques are powerful methods for synthesis and verification of digital circuits. The advances in DD representations for discrete valued functions in terms of computational efficiency can be exploited in the calculation of the spectra of Boolean functions. The classical approach in computing the spectrum of a function by taking advantage of factored transformation matrices as used in the "Fast Fourier Transform" may be reformulated in terms of DD based graph algorithms resulting in a complete representation of the spectrum. The relationship between DD based interpretations and the linear algebra based definitions of spectral methods are described.

1 Introduction

The proliferation of the use of spectral techniques for signal analysis and linear systems analysis and design motivates researchers to look for ways to apply these methods to digital systems. The pioneering work of Karpovsky [14] and Lechner [15] are generally regarded as the basis of subsequent work in this field.

Many digital system design tasks can be performed in the spectral domain. The principles of spectral methods have been applied to many areas in digital systems engineering. Some of these include synthesis [14] [9] [13] [16] [20] [22] [4] [18], partitioning techniques [14] [15][1] [26] [27], testing [6] [12] [19] [24], function classification [13] [8], verification [25] and others. It has been shown that certain problems such as disjoint decomposition [1] [26] and function classification [13] cannot be solved with less complexity in the Boolean domain than in the spectral domain. Unfortunately, many of these techniques have not seen practical application since the computation of the spectrum was too costly. The relevance of the result presented here is that the spectrum may be computed directly using graph algorithms without resorting to exponentially large numerical calculations in many instances.

*This work was supported by the NSF under grants CCR-0000891 and INT-0096008 and DaaD grant 315/PPP/gü-ab.

The *Decision Diagram* (DD) structure allows for the compact representation of discrete functions in the Boolean domain (BDDs) [2] and the spectral domain as a *Spectral Decision Diagram* (SDD). It is possible to formulate the spectral transformation operation as a graph traversal algorithm resulting in the direct conversion of a BDD into a SDD [17] [23]. This can be accomplished without using graph-based vector-matrix operations such as those described in [4][10] by utilizing the 2×2 transform matrix and traversing the BDD functional representation applying the 2×2 transformation to each vertex visited in the BDD. Once all vertices have been transformed, the resulting structure is the SDD of the function.

From [7], it is observed that one characterizing factor of a DD is the decomposition type existing at each internal vertex. Given a BDD representing a function to be transformed, it is possible to visit each vertex and to change (transform) the vertex decomposition into a type representing the desired spectrum. When all such vertices have been transformed, the result is a SDD representing the spectrum. This observation is the basis for developing graph algorithms for the determination of the spectrum of a Boolean function and allows relationship between DD and spectral interpretations to be easily seen.

Since the vertices are transformed individually utilizing the 2×2 transformation matrix, this type of spectrum computation is the graph-algorithm based method of the "fast" transform discussed in [5] for the discrete Fourier transform. Each vertex transformation represents the application of a "butterfly" operation. Such operations are termed as "butterflies" due to the shape of the signal flow graph representing the basic operation. It is possible that the application of the butterfly to a particular DD vertex can lead to adding or removing other vertices from the DD undergoing a transformation. This is to be expected as it is clear that a BDD is not necessarily topologically identical to SDDs in the spectral domain.

2 Transformations of Shannon Trees

It is insightful to present the "fast" transformation technique based on DDs by first considering the case of transformations over trees. Although this technique is impractical

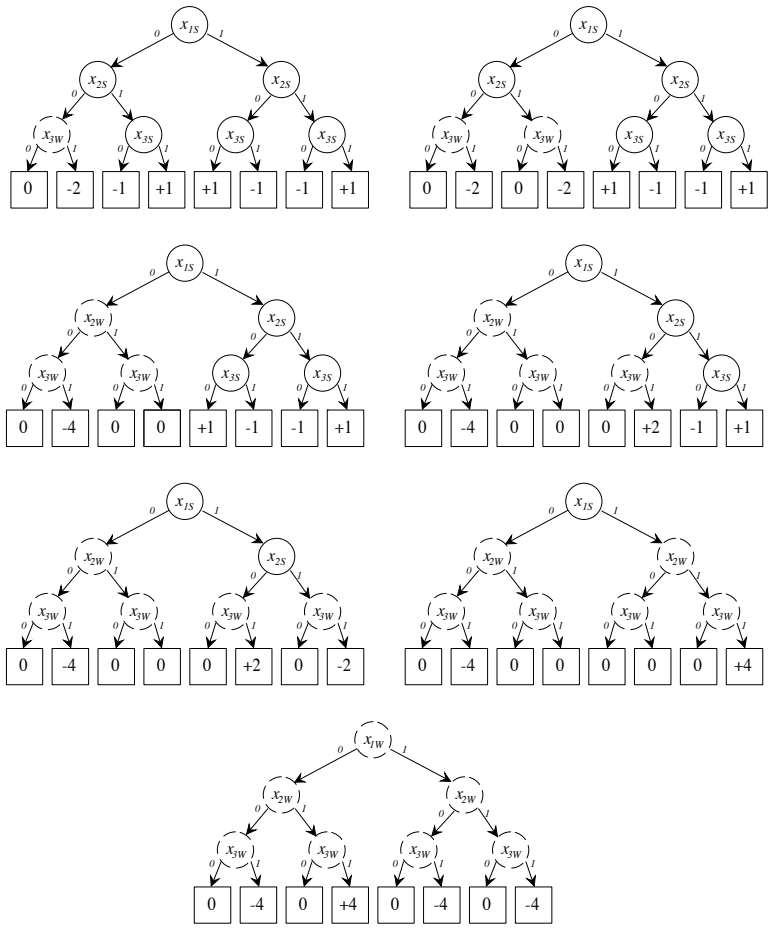

Figure 2. Example of a Series of Shannon Trees Undergoing a Walsh Transformation

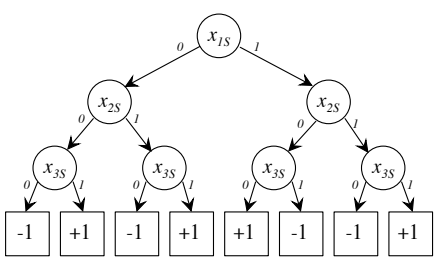

Figure 1. Example Shannon Tree with Integer-Valued Terminal Vertices

for implementation, it does allow for the basis of the method to be easily explained.

A Shannon tree is a binary tree representation of a fully-specified single output function. Unlike the BDD representation, Shannon trees are complete and consist of $2^n - 1$ non-termianl vertices and 2^n terminal vertices. The decomposition type at each non-terminal obeys the Shannon decomposition relationship. As an example of the Shannon tree for the three-variable function, $f = \overline{x_1}\,\overline{x_3} + x_2\overline{x_3} + x_1\overline{x_2}x_3$, consider the diagram in Figure 1.

Note that the terminal vertices are labeled with integers +1 and -1 as opposed to the Boolean values 0 and 1. This labeling allows us to apply the one-variable transform (or butterfly operation) directly. Consider the Walsh transform with Hadamard ordering. This transform can be represented in terms of Kronecker products as discussed in [13]. Note that in terms of graph operations, the one-variable transform simply replaces the subtree $low(v)$ with that of $low(v) + high(v)$ and correspondingly replaces the subtree, $high(v)$, with that of $low(v) - high(v)$. This is the butterfly operation for the Walsh transform [21] as phrased in terms of graph operations. It is easy to see this is the case upon examination of Equation 1.

$$\begin{bmatrix} low(v_W) \\ high(v_W) \end{bmatrix} = \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix} \begin{bmatrix} low(v_S) \\ high(v_S) \end{bmatrix} \qquad (1)$$

After the butterfly operation is performed on the Shannon decomposed vertex, v_S, it is transformed into the Walsh decomposed vertex, v_W. The variable ordering in the tree is important since each butterfly operation may only be ap-

plied of the subtrees have already been transformed. Initially, transforming the terminal vertices to the integers ± 1 allows for the non-terminal nodes at the bottom of the tree to be transformed. By successively applying the transformations in a bottom-up manner, the Shannon tree is transformed into a Walsh tree and the Walsh spectrum is present in Hadamard order from left to right. Figure 2 illustrates the state of the tree at various points in the application of the transformation from the Shannon to the Walsh decomposed tree.

The analogy between this type of transformation and the commonly known fast transform butterfly diagrams is shown in Figure 3 where the Shannon tree is shown with the corresponding butterfly diagram rotated 90° and appearing under the tree.

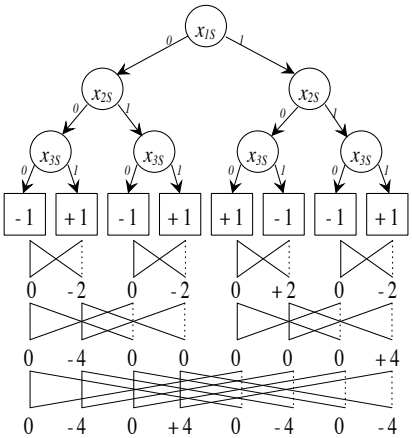

Figure 3. Example of Shannon Tree Undergoing a Walsh Transformation with Butterfly Diagram

This technique can be stated in a succinct form as a top-down, depth-first search algorithm as:

```
Walsh_Transform (f)  {
  if ((low(f)==(terminal
         || walsh_vertex)) &&
      (high(f)==(terminal
         || walsh_vertex))) {
           Walsh_Butterfly(f);
           return; }
  else if ((low(f)!=(terminal
         || walsh_vertex)))
           Walsh_Transform(low(f));
  else if ((high(f)!=(terminal
         || walsh_vertex)))
           Walsh_Transform(high(f)); }

Walsh_Butterfly (f)  {
  low(f) = low(f) + high(f);
  high(f) = low(f) - high(f); }
```

3 DD Algorithm for Fast Transformation

The tree based algorithm offers no computational advantage over the direct computation of the spectrum using matrix algebra since the size of the tree is exponential in the number of dependent function variables. In order to take advantage of shared topological isomorphic subgraphs as are found in reduced DD structures, the tree based algorithm must be modified to account for the case when non-terminal variables are present along a path without subsequent valued level indices. This case never occurs in a tree, but often results in a reduced DD. As an example, consider the case where the function $f = \overline{x_1}\,\overline{x_3} + x_2\overline{x_3} + x_1\overline{x_2}x_3$ is to be transformed to the Walsh domain. Figure 4 contains a diagram representing the reduced BDD of this function with variable order, $x_1 \prec x_2 \prec x_3$. As is easily seen, the path specified by $x_1 = 0$ and $x_3 = 0$ skips the intermediate variable, x_2. This occurs due to the fact that x_2 is redundant in this case because both 0-cubes $\overline{x_1}\,\overline{x_2}\,\overline{x_3}$ and $\overline{x_1}x_2\overline{x_3}$ are members of the set, $ON(f)$, and are present in the reduced BDD as the path representing the merged cube, $\overline{x_1}\,\overline{x_3}$. However, in modifying the decomposition of non-terminal vertex, x_1, from a Shannon to a Walsh type, the absence of a vertex representing variable x_2 cannot be ignored and must be inherently included.

Consider a portion of the BDD shown in Figure 4. This is the part of the BDD corresponding to the path $x_1 = 0$ and $x_3 = 0$. In Figure 5, a subscript is used with each variable to indicate if the corresponding vertex is decomposed using the Shannon or Walsh relationships. Initially, the bottom vertex labeled with variable x_{3S} in Figure 4 undergoes a transformation yielding the vertex x_{3W} as shown in the DD in Figure 5. This DD is a hybrid case consisting of both Shannon and Walsh vertices. In this initial case, the bottom vertex labeled x_{3S} in Figure 4 pointed directly to terminal vertices and the transformation is performed in the same manner as is done for the tree-based approach.

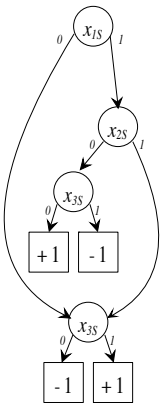

Figure 4. Reduced BDD for the Example Function

Next, the top vertex, x_{1S}, in Figure 4 must be transformed. However the level value of x_{1S} differs by that for x_{3W} by more than one (i.e. $3 - 1 = 2$) indicating that intermediate variables in the ordering are not present in the DD. Due to the decomposition relation for the Walsh n=vertex, the "skipped" vertex must be considered. By applying the the transformation to "skipped" x_{2S} resulting in x_{2W}, we see that the resulting DD does indeed contain vertex x_{2W}. Furthermore, this case can be stated as a general decomposition rule as originally described in [17]. Whenever a path contains to adjacent vertices with level values that differ by two or more, the "skipped" variable, v, must be re-introduced into the resulting Walsh spectral DD with $high(v)$ pointing to terminal value 0 and $low(v)$ pointing to the original subgraph multiplied by two. This occurs since the Walsh decomposition when applied to two identical subtrees, G, results in $G + G$ and $G - G = 0$. Similar transformation rules apply to other spectral transforms that can be described in terms of Kronecker products.

As a complete example, the function under consideration above can be completely transformed by undergoing a series of vertex transformation during a graph traversal as shown in Figure 5. The tree based algorithm outlined above may be modified to operate over DD structures by adding a check for non-sequential index values in a path during traversal and then performing the appropriate transformation.

3.1 Edge Value Attributed DDs

It is noted that the use of attributed edges may also be incorporated into these algorithms resulting in more compact DDs and hence, reduced transformation computation time. As an example, negative-edge attributes in the BDD representation of the function can be extended into arithmetic sign attributes for the spectral MTBDD. Also, edge-values can be used to further reduce the size of the spectral DD allowing it to be represented as a BMD or hybrid MTBDD/BMD as described in [3]. In [11], it was shown that the SDD with edge attributes containing Haar spectral coefficients is isomorphic to the BDD representing the function in the Boolean domain.

4 Conclusion

An algorithm for computing the spectral transformation of a Boolean function represented as a BDD is presented. The technique is based upon the factored matrix formulation used in the development of the FFT. Butterfly operations are implemented in terms of graph addition and subtraction operations resulting in a technique that is implemented through the use of graph manipulations only. This method takes advantage of the compactness inherent in DD structures and can be more effective for Boolean function transformations than traditional approaches. This technique can be used for any transformation that has a Kronecker-product based transformation matrix resulting in a SDD with the coefficients appearing in a natural order.

References

[1] R.L. Ashenhurst. The decomposition of switching functions. In *Int'l Symp. on Theory Switching Funct.*, pages 74–116, 1959.

[2] R.E. Bryant. Graph - based algorithms for Boolean function manipulation. *IEEE Trans. on Comp.*, 35(8):677–691, 1986.

[3] E.M. Clarke, M. Fujita, and X. Zhao. Hybrid decision diagrams - overcoming the limitations of MTBDDs and BMDs. In *Int'l Conf. on CAD*, pages 159–163, 1995.

[4] E.M. Clarke, K.L. McMillan, X. Zhao, M. Fujita, and J. Yang. Spectral transforms for large Boolean functions with application to technology mapping. In *Design Automation Conf.*, pages 54–60, 1993.

[5] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex fourier series. *Math. Computation*, 19:297–301, 1965.

[6] T. Darmala. Generalized transforms for multiple valued circuits and their fault detection. *IEEE Trans. on Comp.*, 41:1101–1109, 1992.

[7] R. Drechsler and B. Becker. *Binary Decision Diagrams - Theory and Implementation*. Kluwer Academic Publishers, 1998.

[8] C. R. Edwards. The application of the rademacher-walsh transform to boolean function classification and threshold logic synthesis. *IEEE Trans. on Comp.*, pages 48–62, 1975.

[9] C. R. Edwards. The design of easily tested circuits using mapping and spectral techniques. *Radio and Electronic Engineer*, 47, no. 7:321–342, 1977.

[10] M. Fujita, J. C.-Y. Yang, E. M. Clarke, X. Zhao, and P. McGeer. Fast spectrum computation for logic functions using binary decision diagrams. In *Int'l Symp. Circ. and Systems*, pages 275–278, 1995.

[11] J.P. Hansen and M. Sekine. Decision diagram based techniques for the haar wavelet transform. In *International Conference on Information, Communication & Signal Processing*, pages 59–63, 1997.

[12] T. C. Hsiao and S. C. Seth. An analysis of the use of rademacher-walsh spectrum in compact testing. *IEEE Trans. on Comp.*, 33:931–937, 1984.

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

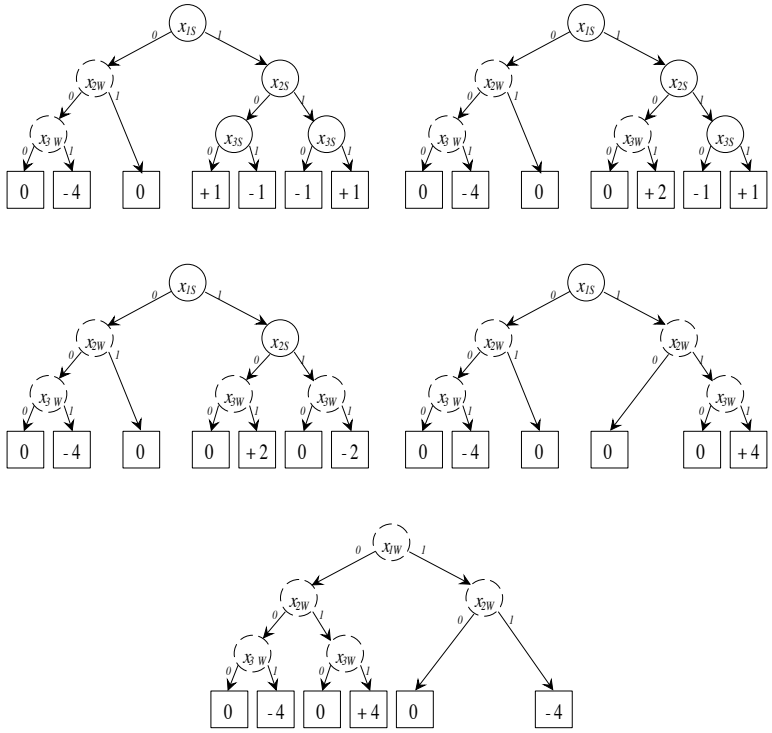

Figure 5. BDD Undergoing Walsh Transformation for the Example Function

[14] M. Karpovsky. *Finite Orthogonal Series in the Design of Digital Devices*. Wiley and JUP, 1976.

[15] R. J. Lechner. Harmonic analysis of switching functions. In A. Mukhopadhyay, editor, *Recent Developments in Switching Theory*, pages 121–228. Academic Press, 1971.

[16] A. M. Lloyd. A consideration of orthogonal matrices, other than the rademacher-walsh types, for the synthesis of digital networks. *J. Electronics*, 47:205–212, 1979.

[17] D. M. Miller. Graph algorithms for the manipulation of boolean functions and their spectra. In *Congressus Numerantium*, pages 177–199, Winnipeg, Canada, 1987.

[18] D. M. Miller. A spectral method for Boolean function matching. In *European Design & Test Conf.*, page 602, 1996.

[19] D. M. Miller and J. C. Muzio. Spectral fault signatures for single stuck-at faults in combinational networks. *IEEE Trans. on Comp.*, 33:765–768, 1984.

[20] M.A. Perkowski, M. Driscoll, J. Liu, D. Smith, J. Brown, L. Yang, A. Shamsapour, M. Helliwell, and B. Falkowski. Integration of logic synthesis and high-level synthesis into the DIADES design automation system. In *Int'l Symp. Circ. and Systems*, pages 718–751, 1989.

[21] J. L. Shanks. Computation of the fast walsh-fourier transform. *IEEE Trans. on Comp.*, 18:457–459, 1969.

[22] M. Stanković, Z. Tošić, and S. Nikolić. Synthesis of maitra cascades by means of spectral coefficients. *IEE Proceedings*, 130, no. 1:101–108, 1983.

[23] R. S. Stanković, T. Sasao, and C. Moraga. Spectral transforms decision diagrams. In T. Sasao and M. Fujita, editors, *Representation of Discrete Functions*, pages 55–92. Kluwer Academic Publishers, 1996.

[24] A. K. Susskind. Testing by verifying walsh coefficients. *IEEE Trans. on Comp.*, 32:198–201, 1983.

[25] M. Thornton, R. Drechsler, and W. Günther. Logic circuit equivalence checking using haar spectral coefficients and partial bdds. In *VLSI Design, to appear*, 2000.

[26] V. M. Tokmen. Disjoint decomposability of multiple valued functions by spectral menas. In *Int'l Symp. on Multi-Valued Logic*, pages 88–93, 1980.

[27] D. Varma and E. A. Trachtenberg. Design automation tools for efficient implementation of logic functions by decomposition. *IEEE Trans. on CAD*, 8:901–916, 1989.