Decision Diagram Based Techniques for the Haar Wavelet Transform

Jeffery P. Hansen
Toshiba System ULSI Engineering Laboratory
580-1 Horikawa-cho, Saiwai-ku,
Kawasaki 210, Japan
hansen@sdel.toshiba.co.jp

Masatoshi Sekine
Toshiba Advanced Semiconductor Device Laboratories
1, Komukai-Toshiba-cho, Saiwai-ku,
Kawasaki 210, Japan
sekine@ull.rdc.toshiba.co.jp

Abstract

This paper describes a new data structure called the "Haar Spectral Diagram" (or HSD) useful for representing the Haar spectrum of boolean functions. An alternative ordering of Haar coefficients is used to represent the Haar transform matrix in terms of a Kronecker product yielding a natural decision-diagram based representation. The resulting graph is a point-decomposition of the Haar spectrum using "0-element" edge values. For incompletely specified functions, the Haar spectrum represented as an HSD is shown to require no more nodes than the ROBDD for the same function, and for completely specified functions, the HSD is shown to be isomorphic to the ROBDD.

1 Introduction

In recent years there has been considerable renewed interest in applying spectral techniques to boolean functions. This has been mainly due to the development of decision-diagram(DD)[2] [1] based techniques for operating on the spectra of large boolean functions[3]. Up to now, much of the recent DD-based work on spectra for boolean functions has used the Walsh transform[3][5][8][11]. Applications have included technology mapping using Chow parameters[3], and spectral translation[8]. One problem with the Walsh transform is that all of the coefficients reflect global information of the function. While adequate when the characteristics of the function are homogeneous, it breaks down for functions such as ALUs, with localized characteristics determined by a small number of input lines.

The problem of localized characteristics also occurs in digital signal processing problems where a speech sample or an image has different characteristics in different parts of the sample. Wavelets, with their multi-resolution properties, have recently been demonstrated to be quite effective when applied to these types of data as opposed to earlier Fourier- and Z-transform-based approaches.

Based on the success of wavelet techniques in the signal processing domain, and on early work with the Haar transform for logic synthesis[9], spectral-based digital logic design research has shifted toward using the Haar transform [4][6] [7]. In this paper we propose new decision-diagram-based methods for computing and operating on the Haar transform.

2 The Haar Transform

The Haar transform[10] is a multi-resolution wavelet transform based on the set of Haar functions $H_i^q(k)$ where $0 \leq k < 1$. Except for H_0^0 which is 0 for all values in the defined range, each of the functions takes on the value $|\sqrt{2}|^{i-1}$ in the first half of some window, the value $-|\sqrt{2}|^{i-1}$ in the second half of the window, and 0 outside the window. In applications involving boolean functions, the Haar transform is usually normalized by retaining only the sign for non-zero values, and by arranging the functions into a $2^n \times 2^n$ matrix where n is the number of inputs. For example, the traditional Haar transform matrix for a 3-input boolean function is:

$$T^3 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

The Haar transform $S^n(f)$ of an n-input function f is then the 2^n element vector:

$$S^n(f) = T^n Y^n(f) \tag{1}$$

where $Y^n(f)$ is a 2^n element vector representing the truth table of the function.

There are two commonly used encodings for mapping truth table entries to the numeric values of this vector. In the "R" encoding, logic-0, logic-1 and "don't care" are mapped to 0, 1 and $\frac{1}{2}$; in the "S" encoding they are mapped to 1, -1 and 0. We choose the

"S" encoding in our work because it can be more naturally extended to p-valued logic (by using p evenly spaced points around the unit circle[10]) and because the spectrum of the compliment of a function can be obtained simply by negation of the spectrum without an additional adjustment to the first element.

Unlike the Walsh transform in which each of the coefficients contains global information about the transformed function, most of the coefficients of the Haar transform contain local information. In fact, only the first two Haar transform coefficients contain global information. The remaining coefficients are a measure of the correlation between the function f, and some variable x_i, sometimes called an "extended literal" within some window function $w(x_0, \ldots, x_{i-1})$ representing a single cube.

Individual Haar coefficients can be computed as the inner product of the corresponding Haar function, and the transformed function. This inner product can be efficiently computed using BDD-based methods by counting input combinations which "match" a given window function, and by comparing the number of matching paths to the various terminal nodes which pass through the 1-edges of the node for the extended literal with those that pass through the 0-edges[6].

3 Recursive Form of Haar Transform Matrix

The problem with computing single coefficients is that even if they can be computed efficiently, there are still 2^n coefficients for an n-input function making operations such as finding the maximum valued coefficient or counting the coefficients having a particular value prohibitively expensive. While the recursive structure of the Hadamard-Walsh matrix makes a compact point-decomposition based decision-diagram representation of the Walsh transform relatively strait-forward[11], the standard Haar transform matrix does not lend itself well to this. For these reasons, we propose a new ordering of the coefficients, determined by an alternate Haar matrix which can be defined in terms of a Kronecker product. We first define the $2^n \times 2^n$ "DC" matrix D^n in which the first row is all ones, and all other elements are zero, then, we separate the Haar transform matrix into this DC part and the remaining "AC" part A^n as:

$$T^n = A^n + D^n \tag{2}$$

We now define the matrix A^n for the "AC" part to be:

$$A^n = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \circledast A^{n-1} + \begin{bmatrix} 0 & 0 \\ 1 & -1 \end{bmatrix} \circledast D^{n-1} \tag{3}$$

where $A^0 = 0$. For $n = 3$, this modified Haar matrix has the form:

$$T^3 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \tag{4}$$

This matrix is exactly the same as the standard Haar transform matrix except that the rows have been reordered. The coefficient s_i of this modified Haar transform is interpreted by considering the binary encoding of i. Bit j of the index i corresponds to the input variable x_j. The lowest non-zero bit determines the variable used as the extended literal, and the bits above that bit are a binary encoding of the cube used as the window function with bit values of 1 for non-complimented literals and 0 for complimented literals. For example, consider the spectral parameter s_{44} of a 6-input function f. Since the binary encoding of 44 is 101100, the lowest non-zero bit is bit 2 indicating x_2 is the extended literal, and $x_5\overline{x}_4x_3$ is the window function.

4 The Haar Spectral Diagram

Using the modified Haar matrix, the Haar transform of an n-input boolean function f can be written as:

$$S^n(f) = A^nY^n(f) + D^nY^n(f) \tag{5}$$

Since the first row of A^n is all zeros, and only the first row of D^n is non-zero, $\tilde{S} = AY$ represents the AC components of the spectrum. By Equation (3) we can separate this vector into its low half and high half as:

$$\mathcal{L}(\tilde{S}) = A^{n-1}\mathcal{L}(Y) \tag{6}$$

$$\mathcal{H}(\tilde{S}) = A^{n-1}\mathcal{H}(Y) + D^{n-1}(\mathcal{L}(Y) - \mathcal{H}(Y)) \tag{7}$$

We now use these equations as the basis for a decision-diagram which we call the Haar Spectral Diagram(HSD). While an HSD can be used to represent any numeric vector, unless stated otherwise, we will assume it is being used to represent the Haar transform of some boolean function. An HSD is an acyclic directed graph having a single terminal node, and with all other nodes labeled with a variable and having two edges leaving the node labeled "0" and "1". Each of these variable nodes split the represented vector into a low and a high half. A variable ordering restriction analogous to the restriction used in ROBDDs is also imposed. The 1-edge of each non-terminal node, and

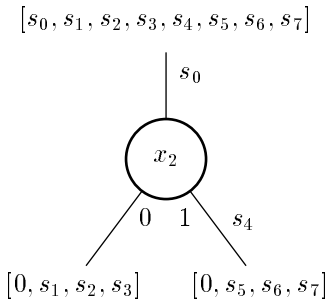

Figure 1: An HSD Node

the top edge of the graph have a "0-element" value. The edge value represents the first element of the vector to which it is applied. That is, an edge value is only applied when the "0" branch is taken for all nodes below it on a path, including any skipped variables or "cross points"[11]. In practice this implies that no more than one non-zero value can be applied on any path.

A node in an HSD is redundant when both the low and high edges point to the same node, and when the edge value is zero. The deleted nodes are treated as "cross points" in operations on HSDs. That is the deleted nodes while not physically present in the graph, are treated as if present for the purpose of any computations. In implementing an HSD package, the standard ROBDD practices of using a unique table[1] to ensure there is exactly one node with the same index, children and edge value, as well as a compute table to avoid repeated computations are also used.

Figure 1 shows a node in an HSD. The arc leading into this node represents an eight element vector. The first element of this vector s_0 is stored as the edge value leading into the node. The "0" arc of the node points to the low half of the vector, except that the first element is unused since it has been moved up as the edge value of the parent node. The "1" branch of the node points to the top half of the vector. Again, the first element of the top half s_4 is stored as an edge value.

The HSD is an ideal representation for the Haar transform of a boolean function since the equations for the low half (6) and high half (7) of the spectrum fit naturally onto the definition of the HSD. The right-hand side of the addition in (7) is a vector in which only the first element is non-zero mapping naturally onto the HSD concept of a 0-element edge. Similarly, the right-hand side of (2) is also a vector in which only the first element is non-zero which also maps to the 0-element edge concept for the top node.

5 Relation Between HSD and ROBDD

Note that the resulting graph, like the ROBDD, is a "point decomposition". In a point decomposition, the value of an element in the low or high half, can be determined only from information in that half of the graph. The advantage of a point decomposition is greatly improved efficiency for many graph operations.

Two acyclic graphs are said to be isomorphic when the have the same structure, or more precisely($\mathtt{var}(g)$, $\mathcal{L}^*(g)$ and $\mathcal{H}^*(g)$ are respectively the variable index, the low and high edges of node g):

Definition 1 *ROBDD graph G_R with nodes r_i and HSD graph G_H with nodes h_j are* isomorphic *($G_R \simeq G_H$) if and only if there is mapping function σ such that for any terminal node r_i of G_R, $\sigma(r_i)$ maps to the unique terminal node of G_H, and for any pair of non-terminal nodes the mapping is one-to-one and $h_j = \sigma(r_i)$ holds if and only if $\mathtt{var}(h_j) = \mathtt{var}(r_i)$, $\mathcal{L}^*(h_j) = \sigma(\mathcal{L}^*(r_i))$ and $\mathcal{H}^*(h_j) = \sigma(\mathcal{H}^*(r_i))$.*

Note that in this definition we ignore the fact that while an ROBDD has two terminal nodes "1" and "0", an HSD has only one terminal node and simply declare all terminal nodes as isomorphic to each other. One of the most important properties of the HSD is stated in the following theorem:

Theorem 1 *For any completely specified function f having a truth table represented by ROBDD $G_R(f)$ and Haar spectrum represented by HSD $G_H(f)$ each with the same variable ordering, $G_R(f)$ and $G_H(f)$ are isomorphic.*

Proof: Assume a variable ordering $\pi = [x_{n-1}, \ldots, x_0]$. Also assume we have functions g and h over the variables $x_{i+1}, \ldots, x_0$, and assume that $G_R(g) \simeq G_H(g)$ and $G_R(h) \simeq G_H(h)$ hold. Now form the function $f = \overline{x}_i g + x_i h$. If $g = h$, node x_i in the ROBDD for this new function is redundant and the graph will reduce to $G_R(g)$. Similarly since $g = h$ implies $AY(g) = AY(h)$ and a 1-edge value of 0, the x_i node in the HSD is also redundant and will be reduced. If $f \neq g$, then $G_R(h)$ will be a graph with top node x_i and with 0- and 1-edges $G_R(g)$ and $G_R(h)$. By equations (6) and (7), $G_H(f)$ will be a graph with top node x_i and ignoring edge values, 0- and 1-edges $G_H(g)$ and $G_H(h)$. By the definition of isomorphism, this implies that given the assumptions above $G_R(f) \simeq G_H(f)$. Now since terminal nodes are defined to be isomorphic to each other, and since any boolean function can be constructed in the manner described and has a unique ROBDD, by induction the stated theorem holds. □

An example HSD/ROBDD for the function:

$$x_4 x_3 \overline{x}_2 x_1 + x_4 \overline{x}_2 x_0 + \overline{x}_4 x_3 \overline{x}_2 \overline{x}_1 \overline{x}_0 + \overline{x}_4 \overline{x}_3 x_2 \overline{x}_1 \overline{x}_0$$

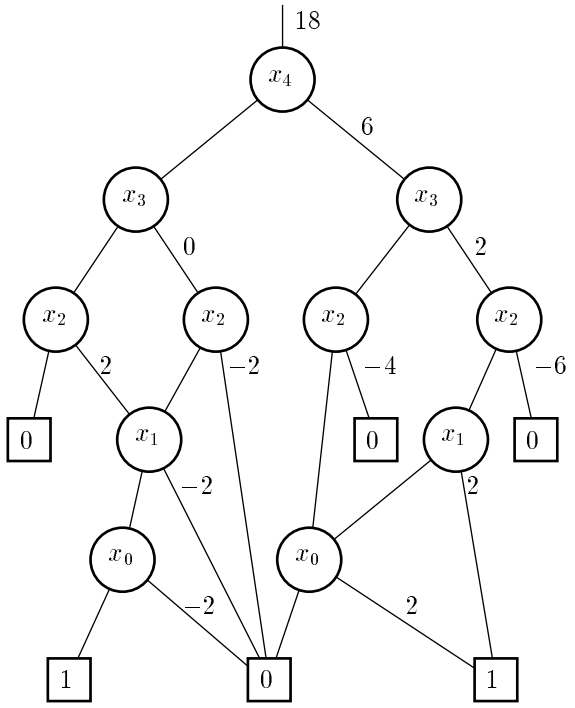

Figure 2: HSD/ROBDD for a Five Input Function

is shown in Figure 2. Since the ROBDD is isomorphic to the HSD, the same graph can be interpreted as either an HSD or an ROBDD. The terminal nodes are marked to correspond to the ROBDD terminal values, but are ignored in the HSD interpretation. The "1" edges are indicated by marking them with their HSD edge value. For completely specified functions, the terminal node values for the ROBDD can also be recovered from the HSD edge values. When the "1" edge value of a node is negative, any terminal node from the "1" edge is logic-0, and any terminal node from the "0" edge is a logic-1. When the "1" edge value is positive, the reverse is true. Furthermore, it can be shown that if the "1" edge value of a node is 0, then that node cannot have a terminal node as a child, since any such node would removed by the deletion rule.

It is also possible to consider a "negation" edge property in addition to the 0-element edge property. A negation edge corresponds to a compliment edge in a ROBDD since under the "S" encoding the Haar spectrum of the compliment of a function is simply the negation of the spectrum of the uncomplimented function. Like with ROBDDs, one edge must be chosen for negation edges, but it does not matter which is chosen. However, if a negation edge is applied to the 1-edge, the negation should be applied above the 0-element bias value to maintain isomorphism with the ROBDD. When using a negation edge, it is also necessary to restrict edge values to being non-negative (or non-positive) to maintain a unique representation.

The internal nodes of an HSD are not necessarily the Haar spectra of the corresponding sub-functions in the ROBDD. In order to obtain the proper Haar spectra, it is necessary to adjust the top 0-element bias edge value. In particular, the top edge value for the Haar spectrum of $f_{x_{n-1}}$ must be changed to $\frac{1}{2}(v_t - v_h)$ where v_t is the old top edge value, and v_h is the edge value on the "1" edge leaving the top node. For the spectrum of $f_{\overline{x}_{n-1}}$, the top value must be changed to $\frac{1}{2}(v_t + v_h)$.

Since an HSD is isomorphic to the ROBDD for the same function, it is only necessary to compute the edge value to obtain an HSD. The top edge value is $\sum f$, and the edge value for a node x_i is $\sum f_{\overline{x}_i} - \sum f_{x_i}$ where $\sum f$ is the sum of the minterm values of f under the appropriate encoding. If we only care about computing the Haar spectral coefficients of completely specified boolean functions, it is not always necessary to have an explicit implementation for the HSD. Assuming the values of $\sum$ are cached in a compute table, the second computation of an edge value can take place in $O(1)$ time. In essence, the ROBDD can simply be interpreted as an HSD with the edge values being stored in the compute table.

In the case of incompletely specified functions the HSD can actually be smaller than the ROBDD. This occurs because of an additional condition under which nodes can be shared in an HSD. Since the DC component of the spectra is stored as an edge value, if two spectra differ only in the DC bias, then they will be represented by the same node differing only in the bias value of the edge leading to that node. For completely specified functions, the only case this can happen is for the degenerate case of the constant functions 1 and 0. However, consider the one input incompletely specified functions represented by the vectors $Y_1 = [1, 0]$ and $Y_2 = [0, -1]$. The spectra of these functions are $S_1 = [1, 2]$ and $S_2 = [-1, 2]$ which differ only in the DC bias causing both functions to be represented with the same HSD node, even though they would be separate nodes in an ROBDD.

6 Operations on HSDs

In this section we describe several basic operations on HSDs. For the most part, the actual implementation is obvious and we only give the relevant mathematic basis. It is assumed that a compute table is used for all calculations. We will use the notation (v, g) to indicate an HSD edge with value v and node g, and the notation $[g_l, g_h]$ to indicate creation of an HSD node with edge 0 being g_l and edge 1 being g_h. In most cases, the index of the node is obvious from context.

The product of a scalar c and an HSD (v, g) is given by $(c \neq 0)$:

$$c(v,g) = (cv, [c\mathcal{L}(g), c\mathcal{H}(g)]) \quad (8)$$

That is, the constant is multiplied into the top edge value, and the scalar multiply operation is applied recursively to the children of the top node. In general, the resulting graph is no longer the Haar transform of some simple boolean function, but the structure of the graph does not change as a result of the operation (except for the special case of $c = 0$).

The sum of two HSDs (v_1, g_1) and (v_2, g_2) is:

$$(v_1 + v_2, [\mathcal{L}(g_1) + \mathcal{L}(g_2), \mathcal{H}(g_1) + \mathcal{H}(g_2)]) \quad (9)$$

Cross points must be considered in the calculation, so in general a parameter indicating the level must be passed to the addition function. The resulting graph is in general, no longer the Haar transform of a simple boolean function, and may have nodes in which the 1-edge values are 0, but have a terminal node as a child.

The maximum coefficient of an HSD $\texttt{maxh}(g)$ can be computed as:

$$\texttt{maxh}((v,g)) = \max(v, \texttt{maxh}(\mathcal{L}(g)), \texttt{maxh}(\mathcal{H}(g))) \quad (10)$$

If the coefficient index is also required, the function can be modified to keep track of the index as it descends, and to store the index value whenever the edge value v is the maximum value. A minimum operation can be implemented in a similar manner.

The number of coefficients of (v, g) having the value t can be computed as:

$$C_t((v,g)) = C_t(\mathcal{L}(g)) + C_t(\mathcal{H}(g)) + \text{eq}(v,t) \quad (11)$$

where $\text{eq}(v,t)$ returns 1 if $v = t$ and 0 if $v \neq t$. Cross points must be considered implementing this function. In particular, since $\mathcal{L}(g) = \mathcal{H}(g)$ at a cross point, only one recursive call is necessary with the result being multiplied by 2. Alternatively, the value returned from a child can be multiplied by 2^i where i is the number of cross points on the edge.

Many other operations are also possible. For example, the authors have developed an algorithm for computing $S(f \oplus g)$ directly from $S(f)$ and $S(g)$. Unfortunately, the equations describing this are extremely complex, and it is likely to be more efficient to do such operations in the boolean rather than spectral domain.

Since an HSD is isomorphic to the corresponding ROBDD, orderings minimizing the number of nodes in an ROBDD will also minimize the number of nodes in an HSD. However, the variable ordering affects the information contained in Haar transform coefficients. Usually it is best to put "controlling" inputs, such as the function select lines on an ALU, near the top of the graph so the Haar spectrum of the individual subfunctions can be accessed separately. This ordering also tends to produce the smallest ROBDDs/HSDs.

7 Conclusion

We have shown in this abstract a new ordering for coefficients in the Haar transform matrix which can be recursively decomposed making it possible to define a new type of decision diagram, the HSD, optimized for representing the Haar transform. We have also shown that the HSD for a boolean function is isomorphic to its ROBDD. We believe that the multiresolution properties of the Haar transform lends itself well to the analysis of functions composed of multiple sub-functions, and anticipate that it will play in important role in logic synthesis for such functions.

References

[1] Brace, K.S., Rudell, R.L and Bryant, R.E., "Efficient Implimentation of a BDD Package", From *27th ACM/IEEE Design Automation Conference* , pp. 40-5 (1990)

[2] Bryant, R,E., "Graph-Based Algorithms for Boolean Function Manipulation", *IEEE Transactions on Computing* , C-38, 8, pp. 677-91, (August 1986)

[3] Clarke, E.M., et. al., "Spectral Transforms for Large Boolean Functions with Applications to Technology Mapping", From *30th ACM/IEEE Design Automation Conference*, pg. 54-60 (1993)

[4] Falkowski, B.J. and Chang, C.H., "A Novel Paired Haar Based Transform: Algorithms and Interpretations in Boolean Domain", From *36th Midwest Symposium on Circuits and Systems*, pp. 1101-1104, (1993)

[5] Falkowski, B.J. and Chang, C.H., "Efficient Algorithms for the Calculation of Arithmetic Spectrum from OBDD and Synthesis of OBDD from Arithmetic Spectrum for Incompletely Specified Boolean Functions", From *IEEE International Symposium on Circuits and Systems*, vol. 1, pg. 197-200 (1994)

[6] Falkowski, B.J. and Chang, C.H., "Efficient Algorithms for Forward and Inverse Transformations Between Haar Spectrum and Binary Decision Diagram Representations of Boolean Functions", From *13th IEEE International Phoenix Conference on Computers and Communications*, pp. 497-503 (1994)

[7] Falkowski, B.J. and Rahardja, S., "Sign Haar Transform", From *27th IEEE Int. Symp. on Circuits and Systems*, vol. 2, pp, 161-164 (1994)

[8] Hansen, J.P. and Sekine, M., "Synthesis by Spectral Translation using Boolean Decision Diagrams", From *33rd Design Automation Conference*, pg. 248-253 (1996)

[9] Hurst, S.L., "The Haar Transform in Digital Network Synthesis", From *IEEE 11th International Symposium on Multi-Valued Logic*, pg. 10-18 (1981)

[10] Hurst, S.L., Miller, D.M. and Muzio J.C., "Spectral Techniques in Digital Logic", Academic Press (1985)

[11] Stanković, R.S., "Some Remarks about Spectral Transform Interpretation of MTBDDs and EVBDDs", From *The Asia and South Pacific Design Automation Conference*, pg. 385-390 (1995)