Hybrid Spectral Transform Diagrams

E. M. Clarke[1] M. Fujita[2] W. Heinle [3]

June 16, 1997

CMU-CS-97-149

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We give a uniform algebraic framework for computing hybrid spectral transforms in an efficient manner. Based on properties of the Kronecker product, we derive a set of recursive equations, which leads naturally to an algorithm for computing such transforms efficiently. As a result, many applications of transforms like the Walsh transform and the Reed-Muller transform, which were previously impossible because of memory constraints, have now become feasible. The same set of recursive equations also gives a new way of explaining hybrid transform diagrams, an efficient data-structure for integer valued boolean functions.

[1]School of Computer Science, Carnegie Mellon University, Pittsburgh
[2]Fujitsu Laboratories of America Inc., Santa Clara, CA.
[3]Institut für Informatik und angewandte Mathematik, University of Bern, Switzerland

This research was sponsored in part by the National Science Foundation under grant no. CCR-8722633, by the Semiconductor Research Corporation under contract 92-DJ-294, and by the Wright Laboratory, Aeronautical Systems Center, Air Force Materiel Command, USAF, the Advanced Research Projects Agency (ARPA) under grant F33615-93-1-1330, and the Swiss National Science Foundation within project 20-45717.95.

Keywords: (Multi- Terminal) Binary Decision Diagrams, Hybrid Transform Diagrams, (Hybrid) Spectral transfomation, Walsh transform, Reed-Muller transform, Spectrum of a boolean function.

1 Introduction

Spectral transformations of boolean functions [14, 17], like the Walsh or Reed-Muller transformations, have numerous applications in computer aided design, especially in the synthesis and testing of combinational circuits. Using a straightforward implementation, the complexity of computing these transformations grows rapidly in the number of variables of the boolean function. In the past, this has severely limited the usefulness of spectral techniques for industrial applications. New techniques for computing the spectrum of a boolean function using binary decision diagrams, have made it possible to compute concise representations for the transforms of functions with several hundred variables [10].

This has led to a proliferation of similar transforms including MTBDD [6], FDD [5], BMD [5], OKFDD [11, 12, 18], and others [17]. It is often difficult to understand, how these transforms are related. In this paper we consider a class of transforms, called *hybrid spectral transforms* [7, 8], which encompasses all of these transforms. Let F be a function, which maps boolean vectors of length n into some set D. The Q–spectrum of F is a linear transformation Qf of the vector representation $f = (\, F(0,\ldots,0) \quad \ldots \quad F(1,\ldots,1)\,)^T$ of F. We consider only spectral transformations Q, which are constructed as *Kronecker products* of certain elementary matrices. When we want to emphasize a particular hybrid spectral transformation Q, we call it a (hybrid) Q–transformation. We provide a uniform algebraic framework for reasoning about such transformations. This framework is based entirely on properties of the Kronecker product. We derive a set of recursive equations, which lead naturally to an algorithm for evaluating products of the form Qv where Q is given as a Kronecker product, and v is a vector. Typically the *hybrid spectral transformation* is given by a matrix Q. This matrix maps a vector representation of the original function to the Q*–spectrum* of F, the vector representation for the Q*–transform.* We show how various spectral transforms like the Walsh transform and the Reed-Muller transform can be computed efficiently using this algorithm. The same set of recursive equations gives a new way of explaining hybrid transform diagrams, an efficient data-structure for representing the Q–transform for D–valued boolean functions. The methodology developed in this paper also yields a concise classification of all known applicable spectral transformation diagrams.

This paper is organized as follows: Section 2 describes the notation we use for boolean functions. Section 3 introduces the Kronecker product, and gives an efficient algoritm for computing the product Qv of a matrix Q given as a Kronecker product and the vector v. Section 4 gives a uniform treatment of hybrid spectral transforms in terms of Kronecker products. We show in section 5, how such transformations can be represented concisely by using a generalization of the MTBDD. The paper concludes in section 6 with a summary and some directions for further research.

2 Binary tree indexing of boolean functions

Let $F : B^n \rightarrow D$ be a D–valued boolean function; assume that F is given in tabular form $\{(b, F(b)) \,|\, b \in B^n\}$. Since the vector $b = (b_0, \ldots, b_{n-1})$ is *encoded* by an integer x where

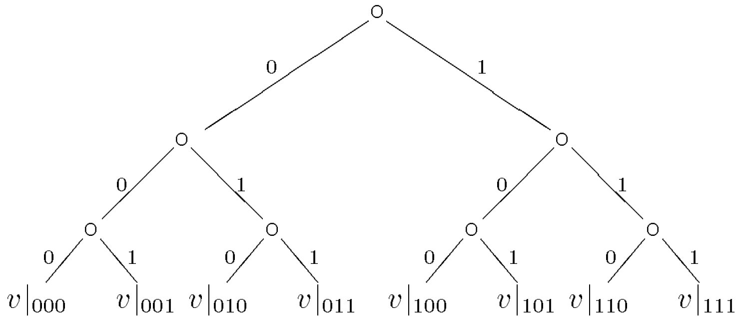

Figure 1: Decomposition of a vector using binary tree indexing

$x \in \{0, \ldots, 2^n - 1\}$, the list of values of F can be represented conveniently as the vector

$$f = (\, F(c(0)) \quad \ldots \quad F(c(2^n - 1)) \,)^T \quad .$$

Alternatively, f may be interpreted as a vector indexed by *boolean sequences* of length n. A *boolean sequence* is either empty (denoted ε), or of the form $c0$ or $c1$, where c is a boolean sequence. Let v be a vector of dimension 2^n, and c a boolean sequence of length $|c| \leq n-1$. Then, we use the following indexing scheme: $v|_c$ is defined such that

$$v|_\varepsilon = v \ , \quad \text{and} \quad v|_c = \begin{pmatrix} v|_{c0} \\ v|_{c1} \end{pmatrix} \quad ,$$

where $v|_{c0}$ and $v|_{c1}$ have the same length. This scheme is called *binary tree indexing*.

When this scheme is expressed in terms of indices, we obtain:

$v|_\varepsilon = v \quad ,$

Let $v|_c = (\, \tilde{v}_0 \quad \ldots \quad \tilde{v}_{2^r-1} \,)^T$ where $r = n - |c|$, then

$$v|_{c0} = (\, \tilde{v}_0 \quad \ldots \quad \tilde{v}_{2^{r-1}-1} \,)^T \quad ,$$

$$v|_{c1} = (\, \tilde{v}_{2^{r-1}} \quad \ldots \quad \tilde{v}_{2^r-1} \,)^T \quad .$$

Vectors indexed by boolean sequences in this manner can be interpreted as Binary Decision Trees, as illustrated in figure 1. The transition to Binary Decision Diagrams [4] then involves elimination of unnecessary nodes and sharing of common subtrees.

3 Computing Kronecker products efficiently

The *Kronecker product* $A \otimes B$ for a $(k \times l)$–matrix A and a $(m \times n)$–matrix B is defined as the following $(km \times ln)$–matrix:

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1l}B \\ a_{21}B & a_{22}B & \cdots & a_{2l}B \\ \vdots & \vdots & & \vdots \\ a_{k1}B & a_{k2}B & \cdots & a_{kl}B \end{pmatrix} \quad .$$

The Kronecker product is associative, but not commutative. Iterated Kronecker products are defined as usual:

$$\bigotimes_{k=0}^{0} A_k = A_0 \quad , \quad \bigotimes_{k=0}^{n} A_k = \bigotimes_{k=0}^{n-1} A_k \otimes A_n \quad .$$

The following identity relates the Kronecker product and ordinary matrix multiplication.

$$(A_1 \otimes B_1) \cdot (A_2 \otimes B_2) = A_1 \cdot A_2 \otimes B_1 \cdot B_2 \quad .$$

As a consequence, inversion distributes over the Kronecker product of nonsingular square matrices:

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1} \quad .$$

Next, we derive a recursive algorithm for computing $Q \cdot f$, where matrix Q is given as a Kronecker product

$$Q = \bigotimes_{i=0}^{n-1} Q_i$$

and binary tree indexing is used for the vector f. The algorithm is efficient because it avoids the construction of the Kronecker product. First, we consider the case where the Q_i is always a (2×2)–matrix. The elements of Q_i are denoted $(Q_i)_{j,k}$, where j and k take the values 0 and 1. The algorithm is based on the following recursive equations, which will be proved below. We assume, that the dimension of $f|_c$ is 2^{n-k}.
When $k = n - 1$:

$$\Big(\bigotimes_{j=k}^{n-1} Q_j \Big) \cdot f|_c = Q_{n-1} \cdot f|_c \quad ,$$

otherwise:

$$\Big(\bigotimes_{j=k}^{n-1} Q_j \Big) \cdot f|_c = (Q_k \otimes I_{n,k}) \cdot \begin{pmatrix} \Big(\bigotimes_{j=k+1}^{n-1} Q_j \Big) \cdot f|_{c0} \\ \Big(\bigotimes_{j=k+1}^{n-1} Q_j \Big) \cdot f|_{c1} \end{pmatrix}$$

In the proof of these equations, the following abbreviations are used (I is the (2×2)–identity matrix):

$$\tilde{Q} = \Big(\bigotimes_{j=k+1}^{n-1} Q_j \Big) \ , \quad I_{n,k} = \Big(\bigotimes_{j=k+1}^{n-1} I \Big) \ .$$

Note that $I_{n,k}$ is an identity matrix which has the same dimension as $\tilde{Q}$. The case when

$k = n-1$ is obvious. The other case is established by the following argument:

$$\begin{aligned}
\Big(\bigotimes_{j=k}^{n-1} Q_j\Big)\cdot f|_c &= Q_k \otimes \Big(\bigotimes_{j=k+1}^{n-1} Q_j\Big)\cdot f|_c \\
&= \begin{pmatrix} (Q_k)_{00}\cdot\tilde{Q} & (Q_k)_{01}\cdot\tilde{Q} \\ (Q_k)_{10}\cdot\tilde{Q} & (Q_k)_{11}\cdot\tilde{Q} \end{pmatrix}\cdot\begin{pmatrix} f|_{c0} \\ f|_{c1} \end{pmatrix} \\
&= \begin{pmatrix} (Q_k)_{00}\cdot\tilde{Q}\cdot f|_{c0} + (Q_k)_{01}\cdot\tilde{Q}\cdot f|_{c1} \\ (Q_k)_{10}\cdot\tilde{Q}\cdot f|_{c0} + (Q_k)_{11}\cdot\tilde{Q}\cdot f|_{c1} \end{pmatrix} \\
&= \begin{pmatrix} (Q_k)_{00}\cdot I_{n,k}\cdot\tilde{Q}\cdot f|_{c0} + (Q_k)_{01}\cdot I_{n,k}\cdot\tilde{Q}\cdot f|_{c1} \\ (Q_k)_{10}\cdot I_{n,k}\cdot\tilde{Q}\cdot f|_{c0} + (Q_k)_{11}\cdot I_{n,k}\cdot\tilde{Q}\cdot f|_{c1} \end{pmatrix} \\
&= \begin{pmatrix} (Q_k)_{00}\cdot I_{n,k} & (Q_k)_{01}\cdot I_{n,k} \\ (Q_k)_{10}\cdot I_{n,k} & (Q_k)_{11}\cdot I_{n,k} \end{pmatrix}\cdot\begin{pmatrix} \tilde{Q}\cdot f|_{c0} \\ \tilde{Q}\cdot f|_{c1} \end{pmatrix} \\
&= \Big(Q_k\otimes I_{n,k}\Big)\cdot\begin{pmatrix} \tilde{Q}\cdot f|_{c0} \\ \tilde{Q}\cdot f|_{c1} \end{pmatrix} \quad .
\end{aligned}$$

Using these equations, $Q\cdot f$ can be computed recursively by setting $k=0$ and $c=\varepsilon$. We illustrate how the recursive algorithm works by the following example. Let

$$f = (1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1)^T \quad ,$$

and $Q = Q_0 \otimes Q_1 \otimes Q_2$ where each Q_i is the matrix:

$$\begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \quad .$$

The computation of $Q\cdot f$ starts with the four 2–element subvectors $f|_{00}, f|_{01}, f|_{10}$ and $f|_{11}$, which are transformed seperately by Q_2. The results of these transformations are then assembled into two 4–element vectors, which are in turn transformed by $Q_1\otimes I_{3,1}$ into the result $Q\cdot f$.

$$\begin{array}{lll}
Q_2\cdot f|_{00} = \begin{pmatrix}1\\-1\end{pmatrix} \longrightarrow & & \\
 & Q_1\otimes I_{3,1}\cdot\begin{pmatrix}1\\-1\\1\\1\end{pmatrix} = \begin{pmatrix}2\\0\\0\\2\end{pmatrix} \longrightarrow & \\
Q_2\cdot f|_{01} = \begin{pmatrix}1\\1\end{pmatrix} \longrightarrow & & \\
 & & Q_0\otimes I_{3,0}\cdot\begin{pmatrix}2\\0\\0\\2\\3\\-1\\1\\1\end{pmatrix} = \begin{pmatrix}5\\-1\\1\\3\\1\\-1\\1\\-1\end{pmatrix} \\
Q_2\cdot f|_{10} = \begin{pmatrix}1\\-1\end{pmatrix} \longrightarrow & & \\
 & Q_1\otimes I_{3,1}\cdot\begin{pmatrix}1\\-1\\2\\0\end{pmatrix} = \begin{pmatrix}3\\-1\\1\\1\end{pmatrix} \longrightarrow & \\
Q_2\cdot f|_{11} = \begin{pmatrix}2\\0\end{pmatrix} \longrightarrow & &
\end{array}$$

Of course, when the algorithm is implemented, the Kronecker products $(Q_j\otimes I_{n,j})$ need not be constructed. $(Q_j\otimes I_{n,j})\cdot v$ can be computed by applying the (2×2)–matrix Q_j to the *blocks* $v|_0$ and $v|_1$ of the vector v. So, instead of matrix multiplication, actually the following operation is performed:

$$(Q_j \otimes I_{n,j}) \cdot \begin{pmatrix} v|_0 \\ v|_1 \end{pmatrix} = \begin{pmatrix} (Q_j)_{00} \cdot v|_0 + (Q_j)_{01} \cdot v|_1 \\ (Q_j)_{10} \cdot v|_0 + (Q_j)_{11} \cdot v|_1 \end{pmatrix}$$

This operation only involves computing linear combinations of the vectors $v|_0$ and $v|_1$ with the scalars $(Q_j)_{kl}$. The original matrix formulation, however, gives a more concise presentation of the algorithm.

This algorithm is easily generalized to Q–transforms over $\mathbb{Z}_l$, where the Q–transform is made up from $(l \times l)$–matrices Q_j. The recursive equations, however, also work for $(m \times l)$–matrices Q_j. In this case we have:

$$Q = \Big(\bigotimes_{j=0}^{n-1} Q_j\Big) \ , \quad I_{n,k} = \Big(\bigotimes_{j=k+1}^{n-1} I^m\Big) \ .$$

where I^m is the $(m \times m)$–identity matrix. Thus, $I_{n,k}$ is a square matrix of dimension $m^{(n-k)}$. The appropriate recursive equations are a natural generalization of the case when $l = 2$. When $k = n - 1$:

$$\Big(\bigotimes_{j=k}^{n-1} Q_j\Big) \cdot f|_c = Q_{n-1} \cdot f|_c \quad ,$$

otherwise:

$$\Big(\bigotimes_{j=k}^{n-1} Q_j\Big) \cdot f|_c = (Q_k \otimes I_{n,k}) \cdot \begin{pmatrix} \Big(\bigotimes_{j=k+1}^{n-1} Q_j\Big) \cdot f|_{c0} \\ \Big(\bigotimes_{j=k+1}^{n-1} Q_j\Big) \cdot f|_{c1} \\ \vdots \\ \Big(\bigotimes_{j=k+1}^{n-1} Q_j\Big) \cdot f|_{c(l-1)} \end{pmatrix}$$

These recursive equations can be used to evaluate the Q–transforms, as will be explained below.

4 Hybrid spectral transforms

In the remainder of the paper, we will restrict our attention to functions that map boolean vectors to $D = \mathbb{Z}_2$ or $\mathbb{Z}$. Such functions $F : B^n \to D$ may be uniformly expressed using the so-called *minterm-representation*. Let $\cdot$ and $+$ be the standard multiplication and addition operations on $\mathbb{Z}_2$ and $\mathbb{Z}$. Also, let $c : \{0, \ldots, 2^n - 1\} \to B^n$ be the encoding for boolean sequences from 2. The *minterm* $m(b)$ for a vector $b \in B^n$ is defined as follows:

$$m(b) = \prod_{i=0}^{n-1} t_i \ , \quad \text{where} \quad t_i = \begin{cases} x_i & \text{if } b_i = 1 \\ \overline{x_i} & \text{if } b_i = 0 \end{cases}$$

The *minterm representation* of $F : B^n \to D$ is given by:

$$F(x_0, \ldots, x_{n-1}) = \sum_{i=0}^{2^n-1} m(c(i)) \cdot f_i \quad .$$

This sum can be regarded the scalar product $m^T f$ of the *minterm*-vector

$$m = \begin{pmatrix} m(c(0)) & m(c(1)) & \ldots & m(c(2^n-1)) \end{pmatrix}^T$$

and

$$f = \begin{pmatrix} F(c(0)) & \ldots & F(c(2^n-1)) \end{pmatrix}^T$$

The minterm vector can be expressed as a Kronecker product:

$$m = \Big(\bigotimes_{j=0}^{n-1} (\, \overline{x_i} \quad x_i \,) \Big)^T$$

For example, when n is 3, the minterm vector is given by:

$$\begin{aligned} m^T &= (\, \overline{x_0} \quad x_0 \,) \otimes (\, \overline{x_1} \quad x_1 \,) \otimes (\, \overline{x_2} \quad x_2 \,) \\ &= (\, \overline{x_0}\,\overline{x_1}\,\overline{x_2} \quad \overline{x_0}\,\overline{x_1}\,x_2 \quad \overline{x_0}\,x_1\,\overline{x_2} \quad \overline{x_0}\,x_1\,x_2 \quad x_0\,\overline{x_1}\,\overline{x_2} \quad x_0\,\overline{x_1}\,x_2 \quad x_0\,x_1\,\overline{x_2} \quad x_0\,x_1\,x_2 \,) \quad . \end{aligned}$$

The concept of minterm representation is easily generalized to get other representations of the same function F:

$$F = m^T f = m^T I f = (m^T Q^{-1})(Q f)$$

where I is the appropriate identity matrix, and Q is some non-singular matrix. The vector Qf is called the *Q–spectrum of F* with respect to the *spectral transformation matrix Q*. Since Q is non-singular, the spectrum of a function provides a *canonical form* for the function. This is easy to see, since $Qf_1 = Qf_2$ iff $f_1 = f_2$. In general, any nonsingular matrix Q can be used for this purpose. However, in practice it is desirable for Q to have a regular structure:

$$Q = \bigotimes_{i=0}^{n-1} Q_i \quad .$$

For boolean functions it is useful to restrict spectral transforms to Kronecker products of (2×2)–matrices over $\{0, 1, -1\}$. There are twelve relevant cases, all other non-singular matrices are either scalar multiples or can be obtained by multiplication with a diagonal matrix.

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix} \quad \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \quad \begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix} \quad \begin{pmatrix} -1 & 1 \\ 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix}$$

The upper six matrices have been assigned names in the literature. They are (from left to right): The *Shannon*-matrix, *Reed-Muller* matrix, *Arithmetic* matrix, *Walsh* matrix, *negative Davio* matrix, and the *inverse negative Davio* matrix. The Shannon and Reed-Muller matrices are usually interpreted over $\mathbb{Z}_2$, whereas the arithmetic and Walsh matrix are defined over the integers [14]. Generally, two important cases of the Q–transformation are distinguished:

$Q = \bigotimes_k Q_0$, the *homogeneous transformation* [10],

$Q = \bigotimes_k Q_k$, not all Q_k–s are equal: the *heterogeneous*, or *hybrid transformation* [7].

Homogeneous transformations formed from the Walsh matrix are called Walsh-transformations. Likewise homogeneous transformations formed from the Reed-Muller matrix are called Reed-Muller transformations. Similar conventions are observed for the other matrices.

Choosing Kronecker products to define spectral transformation matrices for boolean functions has several advantages. First, using the recursive algorithm given in 3, the spectrum of any given function can be computed efficiently without actually ever constructing the huge transformation matrix. Enhancing the representation of the vector from *binary tree indexing* to a BDD-representation improves efficiency considerably due to massive sharing of subvectors. Second, the utilization of Kronecker products of (2×2)–matrices in the transformations results in a modular representation of the function, which generalizes the notion of MTBDD to Q–transform diagrams.

We illustrate the heterogeneous case with an example. Let $F : B^2 \to D$ be given by the table $F(0,1) = 0$, and $F(b_1, b_2) = 1$, otherwise. In this case, the vector f is given by: $f = (\,0 \quad 1 \quad 1 \quad 1\,)$. The minterm representation of F is

$$F(x_1, x_2) = \overline{x_1}\,\overline{x_2} \cdot 0 + \overline{x_1}x_2 \cdot 1 + x_1\overline{x_2} \cdot 1 + x_1 x_2 \cdot 1 \quad .$$

F is interpreted over the integers (thus $\overline{x} = 1 - x$). The spectrum of F with respect to the heterogeneous transformation $Q = Q_0 \otimes Q_1$ where Q_0 is the Arithmetic matrix and Q_1 the Walsh matrix is given as follows:

$$Qf = \left(\begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\right)\begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ -1 & -1 & 1 & 1 \\ -1 & 1 & 1 & -1 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ 1 \\ 1 \end{pmatrix}$$

According to the definition above, the Q–transform gives a new term representation for F:

$$\begin{aligned}
(m^T Q^{-1})(Qf) &= \left(\begin{pmatrix} \overline{x_0}\,\overline{x_1} \\ \overline{x_0}\,x_1 \\ x_0\,\overline{x_1} \\ x_0\,x_1 \end{pmatrix}^T \frac{1}{2}\begin{pmatrix} -1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & -1 & -1 & -1 \\ -1 & 1 & -1 & 1 \end{pmatrix}\right)\begin{pmatrix} 1 \\ -1 \\ 1 \\ 1 \end{pmatrix} \\
&= \frac{1}{2}\begin{pmatrix} 1 \\ 1 - 2x_1 \\ x_0 \\ x_0(1 - 2x_1) \end{pmatrix}^T \begin{pmatrix} 1 \\ -1 \\ 1 \\ 1 \end{pmatrix} \\
&= x_1 + x_0(1 - x_1) \\
&= x_1 + x_0\overline{x_1}
\end{aligned}$$

Of course, this representation is equivalent to the minterm representation.

5 Hybrid transform diagrams

The term-representation of F using a suitable Q–transform may be considerably shorter and more efficient to evaluate than its equivalent minterm representation. We develop BDD-like representations for Q–transforms, which allow efficient computation and evaluation of such transforms.

Let F be a D–valued boolean function in the variables $x_0, \ldots, x_{n-1}$, and

$$Q = \bigotimes_{i=0}^{n-1} Q_i \quad .$$

Then, the Q–transform of F is given by

$$F(x_0, \ldots, x_{n-1}) = m^T Q^{-1} \tilde{f}$$

where $\tilde{f} = Qf$ is the Q–spectrum of F. Inserting the definition of Q as well as the decomposition of m as mentioned earlier we obtain:

$$\begin{aligned} F(x_0, \ldots, x_{n-1}) &= \Big(\big(\bigotimes_{j=0}^{n-1} (\, \overline{x_j} \quad x_j \,) \big) \cdot \bigotimes_{i=0}^{n-1} Q_i^{-1} \Big) \cdot \tilde{f} \\ &= \Big(\bigotimes_{j=0}^{n-1} \big((\, \overline{x_j} \quad x_j \,) \cdot Q_j^{-1} \big) \Big) \cdot \tilde{f} \end{aligned}$$

using the distributivity property of Kronecker and matrix product. Now, for any given assignment to the variables, the value of the transform of F can be computed efficiently using the generalized recursive algorithm from section 3. Since $(\, x_i \quad \overline{x_i} \,) \cdot Q_i^{-1}$ is always a (1×2)–matrix, the identity matrices $I_{n,k}$ degenerate into scalars, which makes the recursion particularly simple. However, in order to compute different values of the same transform, data-structures are needed to represent the transform efficiently. The key to this representation also comes from the recursive equation at the end of section 3. Intuitively, this can be seen best by unfolding the recursion a few steps:

$$\Big(\bigotimes_{j=0}^{n-1} \big((\, \overline{x_j} \quad x_j \,) \cdot Q_i^{-1} \big) \Big) \cdot \tilde{f} =$$

$$\Big((\, \overline{x_0} \quad x_0 \,) \cdot Q_0^{-1} \Big) \cdot \begin{pmatrix} \Big((\, \overline{x_1} \quad x_1 \,) \cdot Q_1^{-1} \Big) \cdot \begin{pmatrix} \Big(\bigotimes_{j=2}^{n-1} \big((\, \overline{x_j} \quad x_j \,) \cdot Q_j^{-1} \big) \Big) \cdot \tilde{f}|_{00} \\ \Big(\bigotimes_{j=2}^{n-1} \big((\, \overline{x_j} \quad x_j \,) \cdot Q_j^{-1} \big) \Big) \cdot \tilde{f}|_{01} \end{pmatrix} \\ \Big((\, \overline{x_1} \quad x_1 \,) \cdot Q_1^{-1} \Big) \cdot \begin{pmatrix} \Big(\bigotimes_{j=2}^{n-1} \big((\, \overline{x_j} \quad x_j \,) \cdot Q_j^{-1} \big) \Big) \cdot \tilde{f}|_{10} \\ \Big(\bigotimes_{j=2}^{n-1} \big((\, \overline{x_j} \quad x_j \,) \cdot Q_j^{-1} \big) \Big) \cdot \tilde{f}|_{11} \end{pmatrix} \end{pmatrix}$$

Again, the product $\left(\left(\overline{x_j} \quad x_j \right) \cdot Q_j^{-1} \right) \cdot v$ can be regarded a block operation on $v|_0$ and $v|_1$:

$$\left(\left(\overline{x_j} \quad x_j \right) \cdot Q_j^{-1} \right) \cdot v = l_j(x_j) \cdot v|_0 + r_j(x_j) \cdot v|_1$$

where

$$\begin{aligned} l_j(x_j) &= \frac{1}{\det Q_j} \cdot \left((Q_j)_{11} x_j - (Q_j)_{10} \overline{x_j} \right) \\ r_j(x_j) &= \frac{1}{\det Q_j} \cdot \left((Q_j)_{00} \overline{x_j} - (Q_j)_{01} x_j \right) \end{aligned}$$

With this notation, the recursive evaluation of the Q–transform of F can be written in the following manner ($x = (x_0, \ldots, x_{n-1})$):

$$\mathrm{eval}(x, f|_c) = \begin{cases} l_{|c|}(x) \cdot f|_{c0} + r_{|c|}(x) \cdot f|_{c1} & \text{if } |c| = n-1 \\ l_{|c|}(x) \cdot \mathrm{eval}(x, f|_{c0}) + r_{|c|}(x) \cdot \mathrm{eval}(x, f|_{c1}) & \text{otherwise} \end{cases}$$

The value of F at x can be computed via $F(x) = \mathrm{eval}(x, \tilde{f}|_\varepsilon)$. The structure of this recursion obviously follows a binary tree-pattern, so, the natural choice to represent the Q–transform is an annotated binary tree, which we call the *Q–transform tree*. The leaves of a Q–transform tree contain the elements of the Q–spectrum, whereas the branches on level j are labeled with $l_j(x)$ on the left branch and $r_j(x)$ on the right branch.

An example best illustrates the situation. We compute the Q–transform tree for

$$f = \begin{pmatrix} 1 & -1 & 1 & -1 & 2 & -4 & 2 & -2 \end{pmatrix}^T \quad ,$$

where $Q = Q_0 \otimes Q_1 \otimes Q_2$, with the matrices

$$Q_0 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad , \quad Q_1 = \begin{pmatrix} 0 & 1 \\ -1 & 1 \end{pmatrix} \quad , \quad Q_2 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \quad .$$

The Q–spectrum Qf can be computed as described in section 3:

$$Qf = \begin{pmatrix} 3 & 0 & 0 & 2 & -1 & 0 & 0 & -2 \end{pmatrix}^T \quad .$$

The evaluation of the Q–transform follows an analoguous recursive scheme. This scheme is displayed below with the products $\left(\overline{x_j} \quad x_j \right) \cdot Q_j^{-1}$ already evaluated.

$$\begin{aligned} F(x_1, x_2, x_3) &= \begin{pmatrix} \frac{1}{2} & \frac{1}{2}(2x_0 - 1) \end{pmatrix} \begin{pmatrix} \begin{pmatrix} 1 & x_1 - 1 \end{pmatrix} \begin{pmatrix} \begin{pmatrix} 1 - 2x_2 & x_2 \end{pmatrix} \begin{pmatrix} 3 \\ 0 \end{pmatrix} \\ \begin{pmatrix} 1 - 2x_2 & x_2 \end{pmatrix} \begin{pmatrix} 0 \\ 2 \end{pmatrix} \end{pmatrix} \\ \begin{pmatrix} 1 & x_1 - 1 \end{pmatrix} \begin{pmatrix} \begin{pmatrix} 1 - 2x_2 & x_2 \end{pmatrix} \begin{pmatrix} -1 \\ 0 \end{pmatrix} \\ \begin{pmatrix} 1 - 2x_2 & x_2 \end{pmatrix} \begin{pmatrix} 0 \\ -2 \end{pmatrix} \end{pmatrix} \end{pmatrix} \\ &= 1 + x_0 - 2x_2 - 4x_0x_2 - 2x_1x_2 - 2x_0x_1x_2 \end{aligned}$$

The recursive scheme can be represented conveniently by a tree. We call such a tree a Q–transform tree. Figure 2 shows the Q–transform tree for the example above.

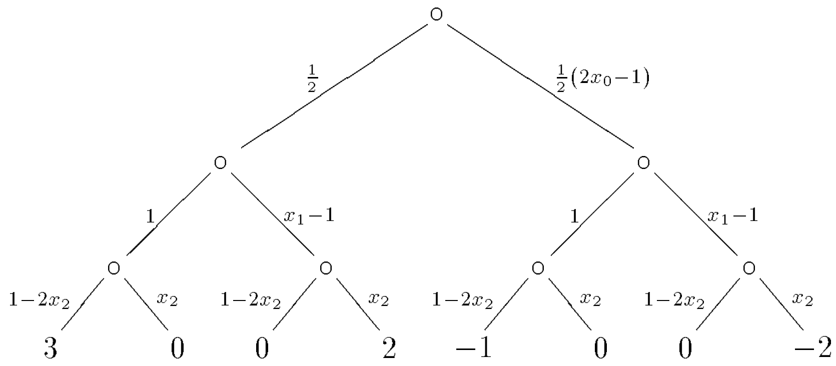

Figure 2: Q–transform tree for F

More formally, we define the Q–transform tree T for f, where the spectral transformation Q is given by a Kronecker product of (2×2)–matrices

$$Q = \bigotimes_{i=0}^{n-1} Q_i \quad ,$$

and the vector f is of dimension 2^n. For any $k \leq n-1$, and boolean sequence c of length less than $n-1$, the k,c–subtree $T_{k,c}$ of T is defined as follows.

the left subtree of $T_{k,c}$ is $T_{k+1,c0}$, the branch leading to it is labeled with $l_k(x)$;

the right subtree of $T_{k,c}$ is $T_{k+1,c1}$, the branch leading to it is labeled with $r_k(x)$;

$T_{n-1,c}$ is defined by:

the left subtree of $T_{n-1,c}$ is the leaf f_{c0}, the branch leading to it is labeled with $l_{n-1}(x)$;

the right subtree of $T_{n-1,c}$ is the leaf f_{c1}, the branch leading to it is labeled with $r_{n-1}(x)$;

T itself is its own subtree $T_{0,\varepsilon}$.

In the case of the Shannon-transformation, a decomposition of the function F itself along these lines is known as the *Shannon-expansion* w.r.t. the variable x_j:

$$F = (\, \overline{x_j} \quad x_j \,)\, (\, F|_{x_j=0} \quad F|_{x_j=1} \,)^T$$

For arbitrary Q–transformations we define the Q–expansion with respect to x_j:

$$F = \Big((\, \overline{x_j} \quad x_j \,)\, Q_j^{-1} \Big) \Big(Q_j \, (\, F|_{x_j=0} \quad F|_{x_j=1} \,)^T \Big) \quad .$$

These expansions just describe the node-operations on the Q–transform trees. Thus, each level of a Q–transform tree can be regarded as a kind of Q_j–expansion with respect to the variable x_j.

Homogeneous *Shannon*–transform trees are an interesting special case. Since the Shannon transformation matrix is the (2×2)–identity matrix, the labeling on the branches becomes particularly simple:

$$l_j(x) = \overline{x_j}\ , \quad r_j(x) = x_j\ .$$

Thus, at every node in such a tree, one of the two branches starting at that node is labeled 0 and the other 1. So, these trees actually are *decision* trees; this is not necessarily the case for other transforms.

The space required to represent boolean functions by Q–transform trees can be greatly reduced in certain situations. Obviously, space can be saved by converting these trees to *directed acyclic graphs*, where identical subtrees are shared. Additional reduction in space arises from the *elimination of unnecessary nodes.* A node in a tree is unnecessary, if both its subtrees are identical. Such a node can be eliminated, after an adjustment is made to the label of the branch preceding this node. Without loss of generality consider a tree where one node on the first level has identical subtrees. This tree is represented by the term:

$$\begin{pmatrix} l_j(x_j) & r_j(x_j) \end{pmatrix} \cdot \begin{pmatrix} \begin{pmatrix} l_{j+1}(x_{j+1}) & r_{j+1}(x_{j+1}) \end{pmatrix} \cdot \begin{pmatrix} B \\ B' \end{pmatrix} \\ \begin{pmatrix} l_{j+1}(x_{j+1}) & r_{j+1}(x_{j+1}) \end{pmatrix} \cdot \begin{pmatrix} A \\ A \end{pmatrix} \end{pmatrix}$$

Here, A, B and B' are the subtrees, where A occurs twice as a subtree of the same node. This expression can be simplified as follows:

$$\begin{aligned}
&\begin{pmatrix} l_j(x_j) & r_j(x_j) \end{pmatrix} \cdot \begin{pmatrix} \begin{pmatrix} l_{j+1}(x_{j+1}) & r_{j+1}(x_{j+1}) \end{pmatrix} \cdot \begin{pmatrix} B \\ B' \end{pmatrix} \\ \begin{pmatrix} l_{j+1}(x_{j+1}) & r_{j+1}(x_{j+1}) \end{pmatrix} \cdot \begin{pmatrix} A \\ A \end{pmatrix} \end{pmatrix} \\
&= \begin{pmatrix} l_j(x_j) & r_j(x_j) \end{pmatrix} \cdot \begin{pmatrix} \begin{pmatrix} l_{j+1}(x_{j+1}) & r_{j+1}(x_{j+1}) \end{pmatrix} \cdot \begin{pmatrix} B \\ B' \end{pmatrix} \\ l_{j+1}(x_{j+1})A + r_{j+1}(x_{j+1})A \end{pmatrix} \\
&= \begin{pmatrix} l_j(x_j) & r_j(x_j) \end{pmatrix} \cdot \begin{pmatrix} \begin{pmatrix} l_{j+1}(x_{j+1}) & r_{j+1}(x_{j+1}) \end{pmatrix} \cdot \begin{pmatrix} B \\ B' \end{pmatrix} \\ (l_{j+1}(x_{j+1}) + r_{j+1}(x_{j+1}))A \end{pmatrix} \\
&= \Big(l_j(x_j) \cdot \begin{pmatrix} l_{j+1}(x_{j+1}) & r_{j+1}(x_{j+1}) \end{pmatrix} \Big) \cdot \begin{pmatrix} B \\ B' \end{pmatrix} + r_j(x_j)(l_{j+1}(x_{j+1}) + r_{j+1}(x_{j+1}))A \\
&= \Big(l_j(x_j) \quad r_j(x_j)(l_{j+1}(x_{j+1}) + r_{j+1}(x_{j+1})) \Big) \cdot \begin{pmatrix} \begin{pmatrix} l_{j+1}(x_{j+1}) & r_{j+1}(x_{j+1}) \end{pmatrix} \cdot \begin{pmatrix} B \\ B' \end{pmatrix} \\ A \end{pmatrix}
\end{aligned}$$

Elimination of unnecessary nodes can be expressed graphically as shown in figure 3. The unreduced tree on the left corresponds to the original formula in the derivation. The tree on the right corresponds to the last formula in the derivation and shows the effect of the reduction.

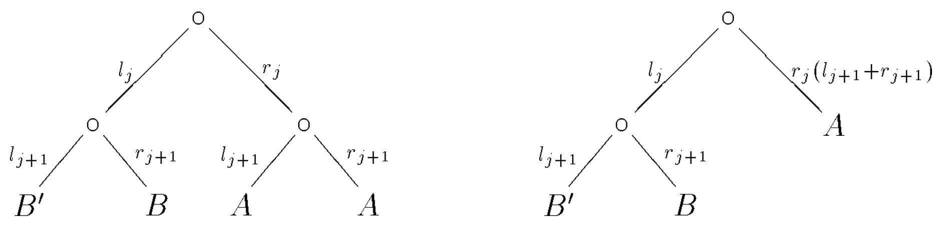

Figure 3: Elimination of unnecessary nodes

The *ordering of the variables* for Q–transform trees has not been considered so far. The variable ordering determines which reductions can be made, and can have a dramatic effect on the size of the final directed acyclic graph. Since this topic has been discussed extensively in the literature [13, 16], we will not discuss it further in this paper.

Now, for any hybrid Q–transformation, we define the corresponding *hybrid Q–transform diagram* to be the Q–transform tree together with the additional operations of node elimination, sharing of common subdiagrams and variable ordering. Various transform diagrams, that appear in the literature, can be classified using this terminology. For example, MTB-DDs can be understood as homogeneous Shannon–transform diagrams. Some of the most commonly used diagrams are listed in the table below.

Matrix	$\begin{pmatrix}1&0\\0&1\end{pmatrix}$	$\begin{pmatrix}1&0\\1&1\end{pmatrix}$	$\begin{pmatrix}0&1\\1&-1\end{pmatrix}$	$\begin{pmatrix}1&0\\-1&1\end{pmatrix}$	$\begin{pmatrix}1&1\\1&-1\end{pmatrix}$
Name	Shannon	Reed-Muller Positive Davio	Negative Davio	Arithmetic	Walsh
Expansion: $(\, l(x) \quad r(x)\,)\, v$	$(\,\overline{x} \quad x\,)\, v$	$(\,\overline{x}-x \quad x\,)\, v$	$(\,1 \quad \overline{x}\,)\, v$	$(\,1 \quad x\,)\, v$	$\frac{1}{2}(\,1 \quad \overline{x}-x\,)v$
Homogeneous Diagrams	BDD [1, 4] MTBDD [6]	FDD [5, 15]	—	BMD [5] ACDD [18]	WDD [18]
Heterogeneous Diagrams	←——— KDD [12] ———→ (Shannon to Negative Davio) ←——— Hybrid transform diagrams [7] ———→ (Shannon to Walsh)				

6 Summary and directions for future research

In this paper we provide a uniform algebraic framework for computing spectral transforms [10] and hybrid transform-diagrams [7, 8] in an efficient manner. The entire discussion is based on properties of the Kronecker product. We derive a set of recursive equations, which leads naturally to an algorithm for evaluating products of the form Qv where Q is given as a Kronecker product, and v is a vector. We show how various spectral transforms like the Walsh transform and the Reed-Muller transform can be computed efficiently using this algorithm. Many applications of these transforms in digital design which were impossible because of memory constraints, have now become feasible [10]. The same set of recursive equations also gives a new way of explaining hybrid transform diagrams, an efficient data-structure for D–valued boolean functions. The use of such diagrams has resulted in the development of new verification methods for computer arithmetic, like word level model checking [9].

It is clear that these ideas are not confined to applications in digital circuit design. D–valued boolean functions, expressed in terms of hybrid transform diagrams (in particular MTBDDs) can be used to represent large matrices [6]. The arguments of these functions are the binary representations of the row and column indices of the matrices. The applications of such methods in numerical analysis and linear algebra are obvious. Gaussian Elimination / LU decomposition with pivoting and iterative methods for solving systems of linear equations are typical examples[2, 6]. Many problems in graph theory like the all pairs shortest path problem for large graphs with weighted edges can also be formulated in terms of matrices and then solved by the methods presented in [2, 6]. Finally, Markov analysis and probabilistic model checking involve huge matrices, and may ultimately benefit from these techniques [2, 3].

References

[1] S. B. Akers. Binary decision diagrams. *IEEE Transactions on Computers*, C–27(6):509–516, June 1978.

[2] I. Babhaar, E. Frohm, C. Gaona, G. D. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Algebraic decision diagrams and their applications. In *Proc. IEEE/ACM ICCAD'93*, pages 188–191, November 1993.

[3] C. Baier, M. Kwiatkowska, M. Ryan, E. Clarke, and V. Hartonas-Garmhausen. Symbolic model checking for probabilistic processes. In *Proceedings, ICALP'97,*. LNCS, Springer Verlag, 1997.

[4] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, 1986.

[5] R. E. Bryant and Y. A. Chen. Verification of arithmetic functions with binary moment diagrams. In *Proc. 32nd ACM/IEEE DAC*, pages 535–541, June 1995.

[6] E. Clarke, M. Fujita, P. McGeer, J. Yang, and X. Zhao. Multi-terminal binary decision diagrams: An efficient data structure for matrix representation. In *IWLS '93: International Workshop on Logic Synthesis, Tahoe City*, May 1993.

[7] E. M. Clarke, M. Fujita, and X. Zhao. Hybrid decision diagrams — overcoming the limitations of MTBDDs and BMDs. In *Proceedings of the 1995 IEEE International conference on computer aided design*, pages 54–60. IEEE Computer Society Press, November 1995.

[8] E. M. Clarke, M. Fujita, and X. Zhao. Multi-terminal binary decision diagrams and hybrid decision diagrams. In T. Sasao and M. Fujita, editors, *Representations of discrete functions*, chapter 4, pages 93–108. Kluwer academic publishers, 1996.

[9] E. M. Clarke, K. Khaira, and X. Zhao. Word level model checking — a new approach for verifying arithmetic circuits. In *Proceedings os the 33rd ACM/IEEE Design Automation Conference*. IEEE Computer society press, June 1993.

[10] E. M. Clarke, K. L. McMillan, X. Zhao, M. Fujita, and J. Yang. Spectral transforms for large boolean functions with applications to technology mapping. In *Proceedings of the 30th ACM/IEEE Design automation Conference*, 54–60, June 1993. IEEE Computer Society Press.

[11] R. Drechsler and B. Becker. OKFDDS–algorithms, applications and extensions. In T. Sasao and M. Fujita, editors, *Representations of discrete functions*, chapter 7, pages 163–190. Kluwer academic publishers, 1996.

[12] R. Drechsler, A. Sarabi, M. Theobald, B. Becker, and M.A. Perkowski. Efficient representation and manipulation of switching functions based on ordered Kronecker Functional Decision Diagrams. pages 415–419, June 1994.

[13] M. Fujita, Y. Matsungara, and T. Kakuda. On variable ordering of binary decision diagrams for the application of multi-level logic synthesis. In *Proc. IEEE EDAC'91*, pages 50–54, February 1991.

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

[15] U. Kebschull, E. Schubert, and W. Rosenstiel. Multilevel logic synthesis bassed on functional decision diagrams. In *IEEE EDAC'92*, pages 43–47, march 1992.

[16] R. Rudell. Dynamic variable ordering for ordered binary decision diagrams. In *Proc. IEEE/ACM ICCAD'93*, pages 42–47, November 1993.

[17] T. Sasao and M. Fujita, editors. *Representations of discrete functions*. Kluwer academic publishers, 1996.

[18] R. S. Stanković, T. Sasao, and C. Moraga. Spectral transform decision diagrams. In T. Sasao and M. Fujita, editors, *Representations of discrete functions*, chapter 3, pages 55–92. Kluwer academic publishers, 1996.