4

MULTI-TERMINAL BINARY DECISION DIAGRAMS AND HYBRID DECISION DIAGRAMS

Edmund M. CLARKE[1]
Masahiro FUJITA[2]
Xudong ZHAO[1]

[1] *School of Computer Science*
Carnegie Mellon University
Pittsburgh, PA 15213, USA

[2] *Fujitsu Labs of America Inc.*
3350 Scott Blvd. Bldg. #34
Santa Clara, CA 95054-3104, USA

Abstract— Functions that map vectors with binary values into the integers are important for the design and verification of arithmetic circuits. We demonstrate how multi-terminal binary decision diagrams (MTBDDs) can be used to represent such functions concisely. The Walsh transform and Reed-Muller transform have numerous applications in computer-aided design, but the usefulness of these techniques in practice has been limited by the size of the binary valued functions that can be transformed. We show how to compute the MTBDD representations of the Walsh transform and Reed-Muller transform for functions with several hundred variables. Bryant and Chen have proposed binary moment diagrams (BMDs) for representing the class of functions that we have considered. We discuss the relationship between these methods and describe a generalization called hybrid decision diagrams which is often much more concise.

4.1 INTRODUCTION

Large integer matrices arise naturally in the design and verification of arithmetic circuits. In this chapter, we describe how to represent and manipulate such matrices efficiently using **multi-terminal binary decision diagrams** (MTBDDs) [7]. An MTBDD is like an ordinary binary decision diagram ex-

T. Sasao et al. (eds.), *Representations of Discrete Functions*

cept that the terminal nodes can be arbitrary integer values instead of just 0 and 1. Previously, we have demonstrated how MTBDDs can be used to represent functions that map vectors with binary values into the integers. Our representation for integer matrices is based on this technique. An integer matrix with dimensions $2^m \times 2^n$ can be treated as a function that maps vectors with binary values of length $m+n$ into the integers. Various matrix operations can be performed by operations on the corresponding integer functions.

The Walsh transform and the Reed-Muller transform [10] have numerous applications in computer aided design, particularly in synthesis and testing of circuits. Unfortunately, the usefulness of these techniques in practice has been limited by the size of the binary valued functions that can be handled by the transform. Since these transforms are given as vectors with length of 2^n where n is the number of variables in the function, currently available techniques limit the functions to less than 20 variables. Since the Walsh matrix and the Reed-Muller matrix have simple recursive definitions, they can be encoded efficiently by MTBDDs. In this manner, we can compute concise representations for the transforms of functions with several hundred variables.

Recently, Bryant and Chen [5] have proposed binary moment diagrams (BMDs) for representing functions that map vectors with binary values into the integers. We show that the BMD of a function is the MTBDD that results from applying the inverse integer Reed-Muller transformation [11] to the function. The transformation can be computed using the techniques that we have developed for manipulating large matrices. The transformation matrix in this case is the Kronecker product [2] of a number of identical 2×2 matrices. We show that the Kronecker products of other 2×2 matrices behave in a similar way. In fact, the transformations obtained from Kronecker products of other matrices will in many cases be more concise than the BMD. We have further generalized this idea so that the transformation matrix can be the Kronecker product of different matrices. In this way, we obtain a representation, called hybrid decision diagram (HDD), that is more concise than either the MTBDD or the BMD.

Our chapter is organized as follows: Section 4.2 gives the basic properties of MTBDDs that are used in the remainder of the chapter. Section 4.3 shows how the results of the previous section can be used to implement standard operations like addition and multiplication of very large integer matrices. Section 4.4 describes how BDDs can be obtained for recursively defined integer matrices and shows how to compute the spectral transforms for binary valued functions. In Section 4.4 we also illustrate the power of this representation by computing the transforms of several very large binary valued functions. Section 4.5

describes the relationship between BMDs and the inverse integer Reed-Muller transformation. This section also introduces Kronecker product and shows how it can be used to generalize BMDs. The next section introduces hybrid decision diagrams and provides experimental evidence to show the usefulness of this representation. The chapter concludes in Section 4.7 with a brief summary and a discussion of directions for future research.

4.2 MULTI-TERMINAL BINARY DECISION DIAGRAMS

Ordered binary decision diagrams (BDDs) are a canonical representation for binary valued functions proposed by Bryant [4]. They are often substantially more compact than traditional normal forms such as conjunctive normal form and disjunctive normal form. They can also be manipulated very efficiently. Hence, BDDs have become widely used for a variety of CAD applications, including symbolic simulation, verification of combinational logic and, more recently, verification of sequential circuits.

A BDD is similar to a binary decision tree, except that its structure is a directed acyclic graph (DAG) rather than a tree, and there is a strict total order placed on the occurrence of variables as one traverses the graph from root to leaf. Algorithms of linear complexity exist for computing BDD representations of $\neg f$ and $f \vee g$ from the BDDs for the functions f and g.

Let $f : B^m \rightarrow Z$ be a function that maps vectors with binary values of length m into integers. Suppose $n_1, \ldots, n_N$ are the possible values of f. The function f partitions the space B^m of vectors with binary values into N sets $\{S_1, \cdots, S_N\}$, such that $S_i = \{\bar{x} \mid f(\bar{x}) = n_i\}$. Let f_i be the characteristic function of S_i. We say that f is in *normal form* if $f(\bar{x})$ is represented as $\sum_{i=1}^{N} f_i(\bar{x}) \cdot n_i$. This sum can be represented as a BDD with integers as its terminal nodes. We call such DAGs **Multi-terminal BDDs** (MTBDDs) [1, 7].

Any arithmetic operation $\odot$ on MTBDDs can be performed in the following way.

$$\begin{aligned} h(\bar{x}) &= f(\bar{x}) \odot g(\bar{x}) \\ &= \sum_{i=1}^{N} f_i(\bar{x}) \cdot n_i \odot \sum_{j=1}^{N'} g_j(\bar{x}) \cdot n'_j \end{aligned}$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{N'} f_i(\bar{x})g_j(\bar{x})(n_i \odot n'_j)$$
$$= \sum_{k=1}^{N''} \left(\sum_{n_i \odot n'_j = n''_k} f_i(\bar{x})g_j(\bar{x}) \right) n''_k.$$

Since the f_i's are mutually disjoint and the g_j's are mutually disjoint, the $f_i g_j$'s are also mutually disjoint. Therefore, the summations $\sum_{n_i \odot n'_j = n''_k} f_i(\bar{x})g_j(\bar{x})$ are mutually disjoint binary valued functions.

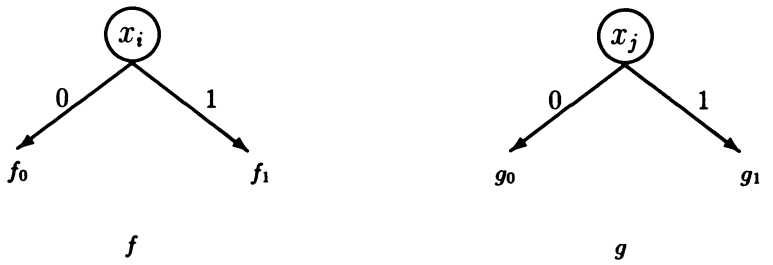

Figure 4.2.1 BDDs for f and g.

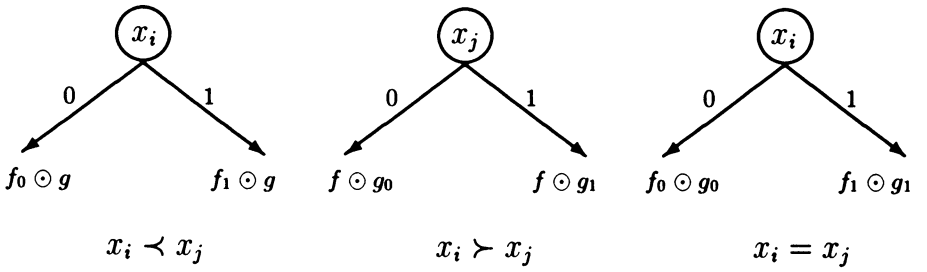

Figure 4.2.2 BDD of $f \odot g$.

We now give an efficient algorithm for computing $f(\bar{x}) \odot g(\bar{x})$.

- If f is a leaf, then for each leaf of g, apply $\odot$ with f as the first argument.
- If g is a leaf, then for each leaf of f, apply $\odot$ with g as the second argument.

- Otherwise, f and g have the form in Figure 4.2.1, and the BDD for $f \odot g$, depending on the relative order of x_i and x_j, is given in Figure 4.2.2. We use the notation $x_i \prec x_j$ to mean that variable x_i precedes variable x_j in the total ordering for the BDD variables.

The resulting diagram may not be in normal form. In order to convert it into normal form, a *reduction* phase is needed. The algorithm for this phase is essentially identical to the reduction phase in Bryant's algorithm for constructing BDDs [4] .

Functions that map vectors with binary values into the integers can also be represented as arrays of BDDs. Each BDD corresponds to one bit of the binary representation of the function value. In general, it is quite expensive to perform operations using this representation.

4.3 MATRIX OPERATIONS

Let M be a $2^k \times 2^l$ matrix over Z. It is easy to see that M can be represented as a function $M : B^{k+l} \rightarrow Z$, such that $M_{ij} = M(\bar{x}, \bar{y})$, where $\bar{x}$ is the binary vector encoding i and $\bar{y}$ is the binary vector encoding j. Therefore, matrices with integer values can be represented as integer valued functions using the MTBDD representation in Section 4.2. We need the following operations for integer matrices for computing the spectral transforms: *absolute value*, *scalar multiplication*, *addition*, *sorting a vector of integers*, *summation over one dimension*, and *matrix multiplication*. The first three operations are trivial and will not be discussed in this chapter.

- Summing matrices over one dimension

  It is sometimes desirable to obtain a 2^n vector from a $2^n \times 2^m$ matrix that each element in the vector is the summation of the corresponding column, i.e. $M'_i = \sum_{j=0}^{2^m-1} M_{ij}$. When the matrices are expressed in terms of integer valued functions, the equation becomes $M'(\bar{x}) = \sum_{\bar{y}} M(\bar{x}, \bar{y})$, where $\sum_{\bar{y}}$ means "sum over all possible assignments to $\bar{y}$". In practice, $\sum_{\bar{y}} M(\bar{x}, \bar{y})$ can be computed as:

$$\sum_{y_1 y_2 \cdots y_m} M(\bar{x}, y_1, y_2, \ldots, y_m)$$

$$= \sum_{y_1 y_2 \cdots y_{m-1}} \sum_{y_m} M(\bar{x}, y_1, y_2, \ldots, y_m)$$
$$= \sum_{y_1 y_2 \cdots y_{m-1}} (M(\bar{x}, y_1, y_2, \ldots, y_{m-1}, 0) + M(\bar{x}, y_1, y_2, \ldots, y_{m-1}, 1)).$$

In this way, each variable in $\bar{y}$ is eliminated by performing an addition.

This operation can also be used to sum the elements of a vector and to obtain a two dimensional matrix from a three dimensional matrix by summing over one dimension. Although this operation works well in many cases, the worst case complexity can be exponential in the number of variables.

- Sorting vectors

  Frequently, it is useful to rearrange the elements in a vector so that they are in non-decreasing order. When the number of different values in the vector is not very large, the sorted vector can be represented concisely without using MTBDDs. In order to uniquely determine a sorted vector, we only need to know the set of different values and the number of occurrences of each value. Thus, the sorted vector can be represented as a list with length m, where m is the number of different values. Each element in the list contains the value and number of its occurrences.

  It is easy to find the set of different values, since it is only necessary to collect all of the terminal nodes in the MTBDD. The number of occurrences N_k of a possible value c_k can be calculated as $N_k = \sum_{i=0}^{2^n-1}$(if $V_i = c_k$ then 1 else 0), where V_i is the ith element of the vector. The operation of summation over a vector discussed previously can be applied to compute this sum. Although, in general, the complexity of the summation operation does not have a satisfactory upper bound, summation over a vector takes time linear in the size of the MTBDD representing the vector. Thus the complexity of the sorting operation is linear in both the number of distinct values in the vector and the size of the MTBDD representation of the vector.

- Matrix multiplication

  Suppose that two matrices A and B have dimensions $2^k \times 2^l$ and $2^l \times 2^m$, respectively. Let $C = A \times B$ be the product of A and B, C will have dimension $2^k \times 2^m$. If we treat A and B as integer valued functions, we can compute the product matrix C as

$$C(\bar{x}, \bar{z}) = \sum_{\bar{y}} A(\bar{x}, \bar{y}) B(\bar{y}, \bar{z}),$$

using the summation operation discussed above. In general, the complexity of this operation can also be exponential in the number of variables.

4.4 SPECTRAL TRANSFORMATIONS OF BINARY VALUED FUNCTIONS

Two of the most commonly used transformations in digital circuit design are the Walsh transform and the Reed-Muller transform [10]. In this section, we will show how the MTBDD based techniques described previously can be used to compute concise representations of the spectra for these transformations.

The Walsh matrix W_n has the recursive definition:

$$W_0 = 1 \qquad W_n = \begin{bmatrix} W_{n-1} & W_{n-1} \\ W_{n-1} & -W_{n-1} \end{bmatrix}.$$

Each element of the matrix is determined by its row and column coordinates. We will encode the 2^n columns by variables $y_n, \ldots, y_1$ and the 2^n rows by the variables $x_n, \ldots, x_1$. W_n can be represented as an integer valued function:

$$\begin{aligned} & W_n(y_n, \ldots, y_1, x_n, \ldots, x_1) \\ = & \begin{cases} W_{n-1}(y_{n-1}, \ldots, y_1, x_{n-1}, \ldots, x_1) & \text{if } (x_n y_n \neq 1) \\ -W_{n-1}(y_{n-1}, \ldots, y_1, x_{n-1}, \ldots, x_1) & \text{if } (x_n y_n = 1) \end{cases} \\ = & W_{n-1}(y_{n-1}, \ldots, y_1, x_{n-1}, \ldots, x_1) \\ & \cdot(\text{if } x_n y_n = 1 \text{ then } -1 \text{ else } 1). \end{aligned}$$

The above recursive definition can be expressed by an MTBDD as shown in Figure 4.4.1.

The Walsh transform maps a binary valued vector f with length 2^n to an integer vector of length 2^n, denoted by W_f, in which each component is between -2^n to 2^n. The transform can be easily expressed using the Walsh matrix, $W(f) = W_n \times (1 - 2f)$ [10]. For example, the vector $[0,1,1,1,1,0,0,0]^T$ is mapped into $[0,0,0,0,-4,4,4,4]^T$.

Likewise, the Reed-Muller matrix has the recursive definition:

$$R_0 = 1 \qquad R_n = \begin{bmatrix} R_{n-1} & 0 \\ R_{n-1} & R_{n-1} \end{bmatrix}$$

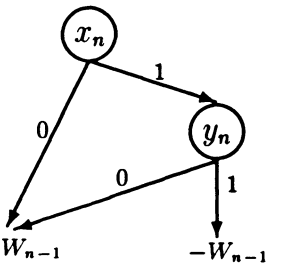

Figure 4.4.1 MTBDD for W_n.

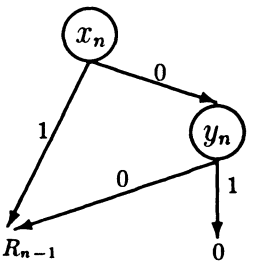

Figure 4.4.2 MTBDD for R_n.

which can be expressed by

$$\begin{aligned} & R_n(y_n, \ldots, y_1, x_n, \ldots, x_1) \\ = \ & \text{if } ((\neg x_n) \cdot y_n) \text{ then } 0 \\ & \text{else } R_{n-1}(y_{n-1}, \ldots, y_1, x_{n-1}, \ldots, x_1) \end{aligned}$$

and has the MTBDD representation in Figure 4.4.2.

The Reed-Muller transformation [11] maps a binary valued vector of length 2^n into another binary valued vector of the same length. This transformation can be expressed by the matrix multiplication $R(f) = R_n \times f$. However, during the matrix multiplication, integer addition is replaced by XOR in order to perform the modulo 2 arithmetic. For example, the vector $[0,1,1,1,1,0,0,0]^T$ is mapped into $[0,1,1,1,1,0,0,0]^T$.

Table 4.4.1 Experimental results for spectral transformations.

example circuit					Walsh coef.		R-M coef.	
circuit	# of inputs	output name	# of gates	|BDD|	|MTBDD|	time	|MTBDD|	time
c1908	33	9	880	3607	1850	44	27748	184
c3540	50	361	1669	520	15985	171	4679	8.2
c5315	178	813	2307	1397	7069	328	2647	25
adder_{50}	100	C_{50}	250	151	7456	23	249	2.3
adder_{100}	200	C_{100}	500	301	29906	128	499	11

When the number of variables is large, the transformations can be computed by representing the matrices and the vectors as MTBDDs and matrix operations can be performed as described in Section 4.3 and Section 4.4.

To illustrate the power of these techniques, we have computed the Walsh transformation and Reed-Muller transformation for some large combinatorial circuits, including two adders and some of the ISCAS benchmarks (Table 4.4.1). The examples were run on a DEC-5000 and run time is shown in seconds. We use the notation |BDD| to indicate the size of a BDD. A similar convention is used for MTBDDs.

4.5 KRONECKER TRANSFORMATIONS

Recently, Bryant and Chen[5] have developed a new representation for functions that map vectors with binary values to integer values. This representation is called the binary moment diagram (BMD) of the function. Instead of the Shannon expansion $f = xf_1 + (1-x)f_0$, they use the expansion $f = f_0 + xf'$, where f' is equal to $f_1 - f_0$. After merging the common subexpressions, a DAG representation for the function is obtained. They prove in their paper that this gives a compact representation for certain functions which have exponential size if represented by MTBDDs directly.

There is a close relationship between this representation and the inverse integer Reed-Muller transformation. The matrix for the inverse integer Reed-Muller transformation is defined recursively by

$$S_0 = 1 \qquad S_n = \begin{bmatrix} S_{n-1} & 0 \\ -S_{n-1} & S_{n-1} \end{bmatrix}$$

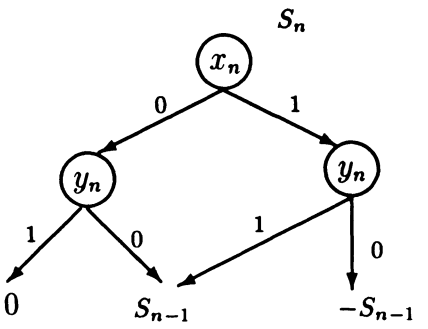

Figure 4.5.1 MTBDD for inverse integer Reed-Muller transformation matrix.

which has a linear MTBDD representation as shown in Figure 4.5.1. Let $\bar{i} \in B^n$ be the binary representation of the integer $0 \leq i < 2^n$. A function $f : B^n \rightarrow Z$ can be represented as a column vector where the value of the i-th entry is $f(\bar{i})$. We will not distinguish between a function and its corresponding column vector. The inverse integer Reed-Muller transformation can be obtained by multiplying the transformation matrix and the column vector $S(f) = S \times f$ using the technique described in the previous section.

Theorem 4.5.1 *The MTBDD of the inverse integer Reed-Muller transform of f is isomorphic to the BMD of f.*

The Kronecker product of two matrices is defined as follows:

$$A \otimes B = \begin{pmatrix} a_{11} & \dots & a_{1m} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nm} \end{pmatrix} \otimes B = \begin{pmatrix} a_{11}B & \dots & a_{1m}B \\ \vdots & & \vdots \\ a_{n1}B & \dots & a_{nm}B \end{pmatrix}.$$

The inverse integer Reed-Muller matrix can be represented as the Kronecker product of n identical 2×2 matrices:

$$S_n = \begin{pmatrix} S_{n-1} & 0 \\ -S_{n-1} & S_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \otimes S_{n-1}$$

$$= \underbrace{\begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \otimes \ldots \otimes \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}}_{n}.$$

The inverse integer Reed-Muller transformation is not the only method that can be used to reduce the size of the BDD representation. Other transformations that are defined as Kronecker products of 2 × 2 matrices may also provide concise representations for functions mapping vectors with binary values into integers. In particular, Reed-Muller matrix R_n and Walsh matrix W_n can be represented as Kronecker products shown below:

$$\begin{aligned} R_n &= \begin{pmatrix} R_{n-1} & 0 \\ R_{n-1} & R_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \otimes R_{n-1} \\ &= \underbrace{\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \otimes \ldots \otimes \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}}_{n} \quad \text{and} \\ W_n &= \begin{pmatrix} W_{n-1} & W_{n-1} \\ W_{n-1} & -W_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes W_{n-1} \\ &= \underbrace{\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \ldots \otimes \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}}_{n}. \end{aligned}$$

In fact, the Kronecker product of any non-singular 2 × 2 matrices can be used as a transformation matrix and will produce a canonical representation for the function. We call such transformations **Kronecker transformations.** If the entries of the 2 × 2 matrix are restricted among $\{0, 1, -1\}$, there are six interesting matrices

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ -1 & 1 \end{pmatrix},$$
$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}, \text{ and } \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}.$$

All other matrices are either singular or would produce BDDs that are isomorphic to one of the six matrices.

We have applied these transformations to the functions discussed in paper[5]. The transformation can be partitioned into two groups of three each. The MTBDDs of the results after applying the transformations in the same group have the same complexity. Let $X = \sum_{i=0}^{n} x_i 2^i, Y = \sum_{j=0}^{m} y_j 2^j, X_j = \sum_{i=0}^{n_j} x_{ij} 2^i$,

the sizes of the results after the Kronecker transformation are shown in Table 4.5.1. The six base matrices can be divided into 2 classes, the first class consists of matrices $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, $\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$, and $\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$; the second class consists of matrices $\begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$, $\begin{pmatrix} 0 & 1 \\ -1 & 1 \end{pmatrix}$ and $\begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$.

Table 4.5.1 Size of BDDs after Kronecker transformations.

class	X	X^2	XY	X^k	$\prod_{j=0}^{k} X_j$
1	$O(2^n)$	$O(2^{2n})$	$O(2^{n+m})$	$O(2^{kn})$	$O(\prod_{j=0}^{k} 2^{n_j})$
2	$O(n)$	$O(n^2)$	$O(nm)$	$O(n^k)$	$O(\prod_{j=0}^{k} n_j)$

The possibility of using BMDs to represent binary valued functions is discussed in [5]. In general, the BMD does not appear to be better than the ordinary BDD for representing binary valued functions. In order to see why this is true, consider the binary Reed-Muller transformation [11], in which operations are performed over Z_2 instead of the integers. The DAG representation of this transformation is sometimes called the Functional Decision Diagram or FDD[9]. This transformation can be obtained by applying the modulo 2 operations to all of the terminal nodes of the BMD. Consequently, the size of FDD is always smaller than the size of the BMD. Since the inverse binary Reed-Muller transformation is the same as the binary Reed-Muller transformation, the FDD for the binary Reed-Muller transformation for f is identical to the original BDD for f. Therefore, for every function f such that $|\text{FDD}_f| < |\text{BDD}_f|$, there exists another function f' which is the binary Reed-Muller transform of f such that $|\text{BDD}_{f'}| < |\text{FDD}_{f'}|$. In particular, both the BMD and the FDD representations for the middle bit of a multiplier are still exponential.

4.6 HYBRID DECISION DIAGRAMS

In the previous sections, we have discussed transformations that can be represented as the Kronecker product of a number of identical 2×2 matrices. If the transformation matrix is a Kronecker product of different 2×2 matrices, we still have a canonical representation of the function. We call transformations obtained from such matrices **hybrid transformations.**

A similar strategy has been tried by Becker [8]. However, his technique only works for the binary domain. When using his technique, all of the transformation matrices, the original function and the resulting function must have binary values. Our technique, on the other hand, works over the integers. By allowing integer values, we can handle a wider range of functions. Moreover, we can obtain larger reduction factors since we have more choices for transformation matrices.

We can apply this idea to reduce the size of MTBDD representation of functions. Since there is no known polynomial algorithm to find the hybrid transformation that minimizes MTBDD size, we use a greedy algorithm to reduce the size. If we restrict the entries in the matrix to the set $\{0, 1, -1\}$, then there are six matrices we can try. For each variable, we select the matrix that gives the smallest MTBDD size. The MTBDDs obtained from such transformations are called hybrid decision diagrams (HDDs).

Although a hybrid transformation can be performed by matrix multiplication, there is a more efficient way of computing it. It can be shown that [2]

$$\bigotimes_{i=0}^{k} A_i = \prod_{i=0}^{k} (I_{2^{i-1}} \otimes A_i \otimes I_{2^{k-i}}),$$

where each A_i is a 2×2 matrix and I_k is the identity matrix of size $k \times k$. A transformation of the form $(I_{2^{i-1}} \otimes A_i \otimes I_{2^{k-i}})$ is called a *basic transformation.* Let $A_i = \begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix}$, and let g be a function represented as a MTBDD, then the basic transformation $g' = (I_{2^{i-1}} \otimes A_i \otimes I_{2^{k-i}}) \times g$ can be computed as

$$\begin{aligned} g' = \quad & \text{if } x_i \text{ then } a_{10}\ g|_{x_i=0} + a_{11}\ g|_{x_i=1} \\ & \text{else } a_{00}\ g|_{x_i=0} + a_{01}\ g|_{x_i=1}. \end{aligned}$$

As a result of this observation, the Kronecker transformation can be performed by a series of basic transformations. Moreover, it can be proved that the order of the basic transformations does not effect the final result.

Suppose the transformation matrix for a hybrid decision diagram can be represented as $\bigotimes_{i=0}^{k} \begin{pmatrix} a_{i1} & a_{i2} \\ a_{i3} & a_{i4} \end{pmatrix}$. Then the hybrid decision diagram nodes at level i is shown in Figure 4.6.1.

We have tried to represent the ISCAS85 benchmark circuits using hybrid decision diagrams 4.6.1. In some cases we have been able to reduce the size of

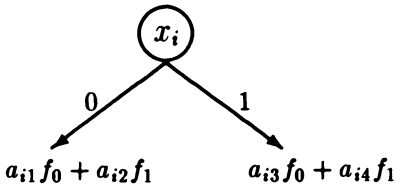

Figure 4.6.1 A hybrid decision diagram node at level i.

Table 4.6.1 Experimental results for hybrid transformations of some ISCAS85 circuits.

circuit	without reordering			with reordering		
circuit	|BDD|	|BMD|	|HDD|	|BDD|	| BMD |	|HDD|
c1355	9419	1217689	2857	4407	478903	1518
c1908	3703	140174	1374	1581	154488	632
c5315	679593	2820	521	108	5106	107

BDD representation by a factor of 1300. However, reductions of this magnitude usually occur when the original function has a bad variable ordering. If dynamic variable ordering is used, then our method gives a much smaller reduction factor.

We have tried several techniques to increase the number of possible matrices. The first technique involves increasing the number of entries in the matrices. This can be accomplished by allowing the entries to take larger values or by using the complex numbers $\{0, 1, -1, i, -i, 1+i, 1-i, i-1, -i-1\}$. Unfortunately, neither extension improved the results significantly.

The second technique involves using transformation matrices that are Kronecker products of larger matrices. For example, we have tried hybrid transformations based on 4×4 matrices instead of 2×2 matrices. Although we have been able to reduce the BDD size even further using this technique, the time it takes to find such transformations is much bigger since the number of possibilities is considerably larger.

Note that our technique can achieve comparable and sometimes better results than dynamic variable reordering. Thus, in some cases, it can serve as an

alternative to dynamic variable reordering. We conjecture that the combination of both techniques together may result in reductions that neither technique can achieve alone.

In order to make the techniques described in the previous sections more useful, it is desirable to be able to perform various arithmetic operations and arithmetic relations on hybrid decision diagrams. This problem is discussed in [6] with detail.

4.7 SUMMARY AND DIRECTIONS FOR FUTURE RESEARCH

In this chapter, we have used MTBDDs to represent functions that map vectors with binary values into integers. We have also shown how to represent large integer matrices concisely and perform standard matrices operations such as scalar multiplication, matrix addition and matrix multiplication.

The Walsh and Reed-Muller transforms are given by matrices that have simple recursive definitions. Because of this, the transforms can be computed efficiently using MTBDDs. In fact, we are able to find the transforms of binary valued functions with several hundred variables.

We discuss the relationship between spectral transforms and binary moment diagrams and describe a generalization called the hybrid decision diagram which is often much more concise. We have also discussed a method to generalize the hybrid decision diagrams by using permutations.

In [7], we show how our technique for computing the Walsh transform can be used in technology mapping. Permutation and complementation of input variables does not change the sorted absolute values of the Walsh spectrum of a binary valued function. Thus, by comparing the Walsh spectra of two binary valued functions, we obtain a necessary condition for determining if one can be changed to the other by these operations.

There are other possible applications of the techniques discussed in this chapter. MTBDDs enable us to represent and manipulate very large matrices efficiently. Some potential applications include image compression, numerical solution of partial differential equations and computation of limit state probabilities for Markov Chains. Since hybrid decision diagrams tend to be more concise than

multi-terminal BDDs, they may prove even more useful for this type of application.

REFERENCES

[1] R. I. Bahar, E. A. Frohm, C. M. Gaona, G. D. Hachtel, E. Macii, A. Pardo, and F. Somenzi, " Algebraic Decision Diagrams and their applications," *Proceedings of the 1993 IEEE International Conference on Computer Aided Design*, pp. 188-191, IEEE Computer Society Press, November 1993.

[2] R. Bellman, *Introduction to Matrix Analysis*, chapter 12, pp. 231-248, McGraw-Hill, 1970.

[3] J. Bern, C. Meinel, and A. Slobodova, " Efficient OBDD-based Boolean manipulation in CAD beyond current limits," *Proceedings of the 32nd ACM/IEEE Design Automation Conference*, pp. 408-413, IEEE Computer Society Press, June 1995.

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

[5] R. E. Bryant and Y. A. Chen, " Verification of arithmetic functions with Binary Moment Diagrams," *Proceedings of the 32nd ACM/IEEE Design Automation Conference*, pp. 535-541, IEEE Computer Society Press, June 1995.

[6] E. M. Clarke, M. Fujita, and X. Zhao, " Hybrid Decision Diagrams - overcoming the limitations of MTBDDs and BMDs," *Proceedings of the 1995 Proceedings of the IEEE International Conference on Computer Aided Design*, pp. 159-163, IEEE Computer Society Press, November 1995.

[7] E. M. Clarke, K. McMillan, X. Zhao, M. Fujita, and J. Yang, " Spectral transforms for large Boolean functions with applications to technology mapping," *Proceedings of the 30th ACM/IEEE Design Automation Conference*, pp. 54-60, IEEE Computer Society Press, June 1993.

[8] 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," *Proceedings of the 32nd ACM/IEEE Design Automation Conference*, pp. 415-419, IEEE Computer Society Press, June 1994. (Also Chapter 7 of this book).

[9] R. Drechsler, M. Theobald, and B. Becker, " Fast OFDD based minimization of fixed polarity Reed-Muller expressions," *Proceedings of the 1994 European Design Automation Conference*, pp. 2-7, IEEE Computer Society Press, June 1994.

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

[11] D. E. Muller, " Application of Boolean algebra to switching circuit design and error detection", *IRE Trans.*, 1:6–12, pp. 6-12, 1954.