Bi-Decomposition of Multi-Valued Relations

Alan Mishchenko [α] **Bernd Steinbach** [β] **Marek Perkowski** [α]

[α]Portland State University
Department of Electrical and Computer Engineering
Portland, OR 97207, USA
[alanmi,mperkows]@ee.pdx.edu

[β]Freiberg University of Mining and Technology
Institute of Computer Science
D-09596 Freiberg, Germany
steinb@informatik.tu-freiberg.de

Abstract

This paper discusses an approach to decomposition of multi-valued functions and relations into networks of two-input gates implementing multi-valued MIN and MAX operations. The algorithm exploits both the incompleteness of the initial specification and the flexibilities generated in the process of decomposition. Experimental results over a set of multi-valued benchmarks show that this approach outperforms other approaches [5,6] *in the quality of final results and CPU time.*

Introduction

Decomposition of multi-valued relations has important applications in data mining and machine learning [1]. Potentially, multi-valued decomposition can also be used in the design of multi-valued hardware devices, software compilers, and asynchronous logic synthesis [2].

Several approaches to multi-valued functional decomposition have been proposed [3-7]. Approach [3] uses Labeled Rough Partitions to perform Curtis decomposition for MV relations. Approach [4] introduces the concept of bi-decomposition for MV relations and [5] develops this concept further. New algorithms for MV decomposition presented in [6,7] rely on Multi-Valued Decision Diagrams (MDDs).

This paper discusses MV decomposition based on

- **Binary-Encoded Multi-Valued Decision Diagrams** BEMDDs have been introduced in [8] to solve the problem of MV support minimization. This hybrid data structure combines the efficiency of BDDs [9] for implicit representation with the expressive power of MDDs [6,7] for manipulation of MV relations.
- **Bi-Decomposition**
  This concept has been under development for the last decade [10,4,5]. Recently, it has been successfully applied to Boolean functions [11].

Essentially, bi-decomposition consists in recursive splitting of large logic blocks into compositions of three smaller logic blocks interconnected as shown in Fig. 1. Blocks A and B typically have reduced complexity (expressed in terms of smaller support or more don't-cares) compared to the initial logic block. Block C always has two inputs. In the binary case, block C is an elementary Boolean function, while in the multi-valued case, it is a MAX or MIN gate.

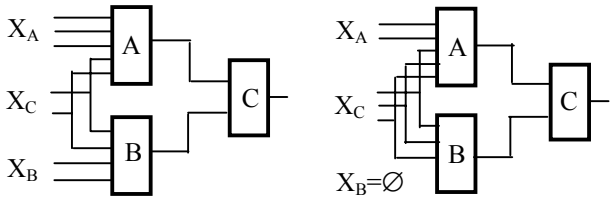

Fig. 1. Schematic representation of the two types of bi-decomposition: strong (left) and weak (right)

The following sections present our approach.

Section 1 introduces BEMDDs as a vehicle for manipulation of multi-valued relations. Section 2 introduces operations on MV relations. Section 3 lists conditions for the existence of MAX- and MIN-bi-decomposition and formulas to derive relations of blocks A and B. Section 4 outlines the main bi-decomposition algorithm. Section 5 gives experimental results. Section 6 concludes the paper.

1 BEMDDs

As stated in the introduction, we advocate the use of BEMDDs as the representation of choice for multi-valued relations. The motivation is that BEMDDs are efficient for large functions and lead to improved decomposition procedures compared to other representations: multi-valued cubes, Labeled Rough Partitions [3], edge-valued BDDs, and pure MDDs [6,7].

In BEMDDs, multi-valued variables are encoded using the smallest sets of binary variables. A k-valued variable requires at least $\lceil \log2(k) \rceil$ binary variables to uniquely encode its values. (Here the vertical bars stand for the closest larger integer.)

For example, a 5-valued variable A is encoded using the set of three binary variables $\{a_1, a_2, a_3\}$. In the simplest case, the set of all possible values of variable A, {0, 1, 2, 3, 4}, is encoded using the set of binary cubes:

$$\{\bar{a}_1\bar{a}_2\bar{a}_3,\ \bar{a}_1\bar{a}_2a_3,\ \bar{a}_1a_2\bar{a}_3,\ \bar{a}_1a_2a_3,\ a_1\bar{a}_2\bar{a}_3\}.$$

If k is not an integer power of two, the minimum length binary encoding results in $(2^{\lceil \log2(k) \rceil} - k)$ spare minterms. It is possible to leave them unused and account for them in the decomposition routines. In this case, it is necessary to remember that the domain of binary variables encoding the inputs is limited to only those minterms that provide codes for the values of multi-valued variables. Experimentally, we found that this approach increases the BEMDD size and makes traversal routines more complicated.

From the practical point of view, it is better to distribute the unused minterms between the values of the function. For example,

consider the two encodings of ten values using sixteen minterms of the four binary variables shown in Fig. 2.

	00	01	11	10
00	0	4	8	8
01	1	5	9	9
11	3	7	10	10
10	2	6	10	10

	00	01	11	10
00	0	2	7	4
01	0	2	8	4
11	1	3	10	6
10	1	3	9	5

Fig. 2. Two ways to encode ten values using binary variables

B\A	0	1
0	0,1,2	2
1	1	0,1,2
2	1,2	0,1

Fig. 3. The truth table for relation R(A,B)

Currently, as the default encoding of values, we use the encoding shown on the left of Fig. 2.

For multi-valued relations, we use a similar encoding scheme to represent the output values. The variables used for this purpose are called *output-value-encoding variables* (or simply, *value variables*). Typically, the value variables are ordered below other variables in the BEMDD.

Algorithms in this paper are reordering-independent. This means that, for the purpose of multi-valued bi-decomposition, the binary variables encoding input or output values can move freely during reordering, not even limited to the boundaries of variable groups encoding the same multi-valued variables.

Definition. Let the domain of a multi-valued variable A_i be D_{Ai}. A multi-valued relation R over a set of multi-valued input variables $\{A_i\}$, with the multi-valued output Z having domain D_z, is a mapping from the Cartesian product of the input variable domains into the domain of subsets of the output values:

$$R(A_1,A_2,\ldots,A_n): D_{A1}\times D_{A2}\times\ldots\times D_{An} \rightarrow 2^{Dz}.$$

Definition. The set of minterms (vertices) of the combined input domain of R such that the output of R is the set of all possible output values, is called the *don't-care set*. The set of other minterms of the domain of R is called the *care set*.

Once the input and output multi-valued variables are encoded, the BEMDD representing a multi-valued relation is constructed as a binary relation. This relation puts in correspondence encoded input values with the appropriate encoded output values.

For example, shown in Fig. 3 is a three-valued-output multi-valued relation depending on binary variable A and ternary variable B. (All examples in this paper use relations depending on the same two variables.) This relation can be represented as a binary relation over five binary variables (three variables for inputs and two variables for the output).

v_1v_2 \ ab_1b_2	000	001	011	010	110	111	101	100
00	1	0	0	0	1	1	1	0
01	1	1	1	1	1	1	1	0
11	1	0	1	1	0	0	1	1
10	1	0	1	1	0	0	1	1

Fig. 4. The Karnaugh map for the binary relation encoding multi-valued relation R(A,B) in Fig. 3.

Suppose the binary variable **a** encodes multi-valued variable A, variables b_1 and b_2 encode B, while variables v_1 and v_2 encode the output values of R(A,B). If ternary values { 0,1,2 } are encoded as { 00,01,1- }, the binary relation $R(a,b_1,b_2,v_1,v_2)$, which represents R(A,B), is expressed using variables a, b_1, b_2, v_1, and v_2 as follows:

$$R(a,b_1,b_2,v_1,v_2) = \bar{a}\bar{b}_1\bar{b}_2 + \bar{a}\bar{b}_1 b_2\bar{v}_1 v_2 + \bar{a}b_1(v_2+v_1) + a\bar{b}_1\bar{b}_2 v_1 + a\bar{b}_1 b_2 + ab_1\bar{v}_1 .$$

Each of the six terms in the above formula is obtained directly by encoding a cell of the multi-valued Karnaugh map in Fig. 3. The Karnaugh map and BEMDD for the resulting binary relation R are shown in Figs. 4 and 5.

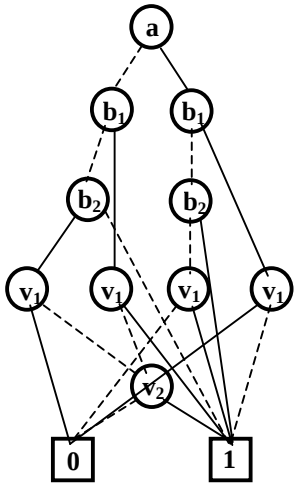

Fig. 5. The BEMDD for multi-valued relation R(A,B).

2 Operations on MV Relations

This section introduces important theoretical concepts and operations used in the procedures that check the existence of MAX- and MIN-bi-decomposition and derive multi-valued relations for logic blocks A and B.

Interval Relation

The bi-decomposition algorithm discussed in this paper is applicable to multi-valued relations satisfying the property of *function interval* [5]. Below this property is restated with some adaptation to the terminology accepted in this paper.

Definition. A multi-valued relation is an *interval relation* if in each vertex (minterm) of the domain, the output values form a contiguous range.

For example, relation R(A,B) in Fig. 3 is an interval relation, but if, for example, in the vertex (0,0) instead of having value set {0,1,2} it had value set {0,2}, it would not be an interval relation because values 0 and 2 do not form a contiguous range.

From the practical point of view, limiting our attention to only interval relations does not seem to restrict the applicability of the decomposition method because the majority of multi-valued benchmarks satisfy this property. For those that do not satisfy, below we propose a simple way to convert a non-interval multi-valued relation to an interval one, by filling in the unused values between the smallest and the largest values in each vertex. For example, adding {1} to the value set {0,2} converts the non-interval relation from the above example into an interval one.

Lower and Upper Bound Intervals

Definition. For the given multi-valued relation R, a *lower (upper) bound interval* (denoted R_{LBI} and R_{UBI}) are multi-valued relations depending on the same input variables, having the same number of possible output values, and differing only in sets of the output values. In each vertex of the domain, the LBI takes all the values starting from 0 up to (and including) the first value used in

this vertex by the given relation, while the UBI takes all the values starting from (and including) the largest value used by the given relation up to the largest possible value of the output variable.

Fig. 6 illustrates this definition. An interval relation F is shown on the left. Its LBI and UBI are given in the center and on the right. For example, consider vertex (0,2). The LBI in this vertex has values starting from 0 up to the first value used by F, which is 1. The UBI in this vertex has all the values starting from the last value of F, which is 2, up to the largest possible value of the relation, which is also 2.

F

B\A	0	1
0	0,1	2
1	1	0,1,2
2	1,2	0,1

F_{LBI}

B\A	0	1
0	0	0,1,2
1	0,1	0
2	0,1	0

F_{UBI}

B\A	0	1
0	1,2	2
1	1,2	2
2	2	1,2

Fig. 6. Example illustrating the upper and lower bound intervals.

If the relation R(x,v) is represented by a BEMDD over the input-encoding variables x and the value-encoding variables v, computation of its lower (upper) bound interval can be performed using the formulas:

$$R_{LBI}(x, v) = \overline{\exists v\left[R(x, v) \,\&\, R_{v<w}(v, w)\right]}_{w \to v},$$

$$R_{UBI}(x, v) = \overline{\exists v\left[R(x, v) \,\&\, R_{v>w}(v, w)\right]}_{w \to v},$$

where $\exists$ stands for the existential quantification and $R_{v<w}$ and $R_{v>w}$ are characteristic functions of relations "more than" and "less than" which depend on two equal-sized interleaved sets v and w of value-encoding variables.

The first formula is interpreted as follows. For each vertex of the domain (given by one assignment of variables x), the lower bound interval contains all values w (v after variable replacement), for which there does not exists value v belonging to the relation such that v is less than w.

The interpretation of the second formula is similar.

It is interesting to note that if the initial relation is not an interval relation, its LBI and UBI are the same as the LBI and UBI of the interval relation created by filling in the intermediate unused values. Therefore, if the relation is represented by the LBI/UBI pair, there is no need for a specialized procedure to convert non-interval relations into interval ones!

In the procedures below, multi-valued relations are represented as LBI/UBI pairs. This simplifies processing because relations of blocks A and B resulting from the bi-decomposition are also computed as LBI/UBI pairs. In our experiments, we observed an additional advantage of this representation, namely that the BEMDDs for LBI/UBI are typically smaller than those for the pure lower/upper bounds as defined in [5].

Interval Increments and Decrements

Another operation for the manipulation of interval relations is the interval increment and decrement discussed in this subsection.

Definition. For the given multi-valued interval relation R, the *upper increment (decrement)* (denoted R^{U+} and R^{U-}) are multi-valued relations depending on the same input variables, having the same number of possible output values, and differing only in sets of the output values. In each vertex of the domain, the value set of R^{U+} (R^{U-}) is determined as the value set of the given relation plus (minus) one value at the upper end of the value set.

Similarly, it is possible to define the *lower increment (decrement)* of an interval relation by adding (subtracting) one value at the lower end of the value set.

Fig. 7 illustrates the definition for the upper increment and decrement. An interval relation R is shown on the left. R^{U+} and R^{U-} are shown in the center and on the right. For example, consider vertex (1,1) with value set {0,1}. R^{U+} in this vertex has values {0,1,2} created as the same value set plus the value immediately following the largest value, which is 2. R^{U-} in this vertex has value set {0} created as the same value set as R minus the largest possible value, which is 1. The dashes (-) in the tables mean that the value set for some vertices is empty.

R

B\A	0	1
0	0,1	2
1	0	0,1
2	0,1,2	0,1

R^{U+}

B\A	0	1
0	0,1,2	2
1	0,1	0,1,2
2	0,1,2	0,1,2

R^{U-}

B\A	0	1
0	0	-
1	-	0
2	0,1	0

Fig. 7. Example illustrating the lower increment and decrement of the interval relation R(A,B).

The upper (lower) increment (decrement) can be computed using the BEMDD representation as follows:

$$R^{U-}(x, v) = \exists v\left[R(x, v) \,\&\, R_{v>w}(v, w)\right]_{w \to v},$$

$$R^{U+}(x, v) = \overline{\exists v\left[\overline{R(x, v)} \,\&\, R_{v<w}(v, w)\right]}_{w \to v} =$$

$$= \forall v\left[R(x, v) + R_{v \geq w}(v, w)\right]_{w \to v}.$$

$$R^{L-}(x, v) = \exists v\left[R(x, v) \,\&\, R_{v<w}(v, w)\right]_{w \to v},$$

$$R^{L+}(x, v) = \overline{\exists v\left[\overline{R(x, v)} \,\&\, R_{v>w}(v, w)\right]}_{w \to v} =$$

$$= \forall v\left[R(x, v) + R_{v \leq w}(v, w)\right]_{w \to v}.$$

If the underlying BDD package does not support the complement edges, the formulas with the universal quantifier are likely to be more efficient because they do not require complementation of the intermediate result.

3 Checking and Deriving Bi-Decomposition

This section presents the main contribution of this paper: the formulas for checking the existence of strong (weak) MAX- and MIN-bi-decomposition and deriving the resulting relations for blocks A and B. It is only necessary to consider the case of MAX-bi-decomposition because the case of MIN-bi-decomposition can be checked (derived) using the same formulas after swapping R_{LBI} and R_{UBI}.

Definition. Expression $R_{crit}(x,v) = [R_{LBI}(x,v)]^{U-}$ is called *critical relation*. For each vertex of the domain, the critical relation contains all the values situated *below* the allowed set of values of the relation R(x,v).

Theorem 1. Multi-valued interval relation R(x,v) specified by UBI $R_{UBI}(x,v)$ and LBI $R_{LBI}(x,v)$ has strong MAX-bi-decomposition with variable sets (x_a, x_b) iff

$$R_{crit}(x,v) \,\&\, \exists x_a\, R_{UBI}(x,v) \,\&\, \exists x_b\, R_{UBI}(x,v) = 0.$$

Sketch of the proof. For each vertex of the domain, the lower bound interval $R_{LBI}(x,v)$ contains all the values below the value set *and* the smallest value of the value set. Computing $R_{crit}(x,v) = [R_{LBI}(x,v)]^{U-}$ removes the smallest value of the value set, so

remaining are only values below the value set that do not belong to the relation.

Each of the two components with the existential quantifier computes the projection of the upper bound interval in the direction of variables x_a and x_b. The product of projections denotes in each domain vertex the largest allowed value belonging to the value set of the relation and all the values above it.

The formula, therefore, reiterates the theorem as follows: the function is MAX-bi-decomposable with the given variable sets iff in all vertices the largest value below the value set is less than the largest allowed value in the row and in the column of the Karnaugh map created by variable sets x_a and x_b. Q.E.D.

R

B\A	0	1
0	1,2	0,1
1	0,1,2	0
2	2	1,2

R_{LBI}

B\A	0	1
0	0,1	0
1	0	0
2	0,1,2	0,1

R_{UBI}

B\A	0	1
0	2	1,2
1	2	0,1,2
2	2	2

$[R_{LBI}]^{U-}$

B\A	0	1
0	0	-
1	-	-
2	0,1	0

$\exists x_a\, R_{UBI}$

B\A	0	1
0	1,2	1,2
1	0,1,2	0,1,2
2	2	2

$\exists x_b\, R_{UBI}$

B\A	0	1
0	2	0,1,2
1	2	0,1,2
2	2	0,1,2

R^A

B\A	0	1
0	2	0
1	2	0
2	2	0

R^B

B\A	0	1
0	1	1
1	0	0
2	2	2

$MAX(R^A, R^B)$

B\A	0	1
0	2	1
1	2	0
2	2	2

Fig. 7. Example of checking the existence of strong MAX-bi-decomposition for relation R(A,B).

The example in Fig. 7 illustrates Theorem 1.

Theorem 2. Multi-valued relation R(x,v) specified by its UBI $R_{UBI}(x,v)$ and LBI $R_{LBI}(x,v)$ is MAX-bi-decomposable in the weak sense with the variable sets (x_a, x_b=∅) iff

$$\psi(x) = \exists v R_{crit}(x,v) - \exists v[R_{crit}(x,v) \,\&\, \exists x_a\, R_{UBI}(x,v)] \neq 0.$$

Proof. The weak MAX-bi-decomposition exists if, after excluding variables x_a from the support of block B, the UBI of this block ($\exists x_a R_{UBI}(x,v)$) has no overlap with the critical relation in at least one vertex of the domain where the value set of the critical relation is not empty. Q.E.D.

The quality of weak MAX-bi-decomposition with one variable in x_a can be measured by the number of minterms of the Boolean function $\exists x_a \psi(x)$.

Theorem 3. If the multi-valued relation R(x,v) specified by UBI $R_{UBI}(x,v)$ and LBI $R_{LBI}(x,v)$ is MAX-bi-decomposable with the variable sets (x_a, x_b), the relations of blocks A and B are:

$$R^A_{LBI} = \exists x_b[(v=0) + R_{LBI} \,\&\, \exists v\,(R_{crit} \,\&\, \exists x_a R_{UBI})\,].$$

$$R^A_{UBI} = \exists x_b\, R_{UBI}.$$

$$R^B_{LBI} = \exists x_a[(v=0) + R_{LBI} \,\&\, \exists v\,(R_{crit} \,\&\, R^A)\,].$$

$$R^B_{UBI} = \exists x_a\, R_{UBI}.$$

Proof. Deriving the UBIs for both blocks is straightforward. Computation of the LBI for block A is based on the observation that it should contain zero value (v=0) everywhere, except those vertices of the domain where the critical relation (R_{crit}) overlaps with the projection of the UBI ($\exists x_a R_{UBI}$). In the latter case, to prevent the violation of the initial relation, the LBI of block A should be set to the LBI of the initial relation.

In the formula for block B, the projection of the UBI is replaced by the completely-specified multi-valued function of block A, found after bi-decomposition is recursively applied to this block. Q.E.D.

The formulas for deriving the relations for blocks A and B are also true for the case of weak bi-decomposition if set x_b is assumed to be empty (no quantification w.r.t. x_b).

The example in Fig. 8 illustrates Theorem 3.

R

B\A	0	1
0	1,2	0,1
1	0,1,2	0
2	2	1,2

R_{LBI}

B\A	0	1
0	0,1	0
1	0	0
2	0,1,2	0,1

R_{UBI}

B\A	0	1
0	2	1,2
1	2	0,1,2
2	2	2

$R_{crit}=[R_{LBI}]^{U-}$

B\A	0	1
0	0	-
1	-	-
2	0,1	0

$\exists x_a\, R_{UBI}$

B\A	0	1
0	1,2	1,2
1	0,1,2	0,1,2
2	2	2

$\exists x_b\, R_{UBI}$

B\A	0	1
0	2	0,1,2
1	2	0,1,2
2	2	0,1,2

$R_{crit} \& \exists x_a R_{UBI}$

B\A	0	1
0	-	-
1	-	-
2	-	-

R^A_{LBI}

B\A	0	1
0	0	0
1	0	0
2	0	0

R^A_{UBI}

B\A	0	1
0	2	0,1,2
1	2	0,1,2
2	2	0,1,2

R^A

B\A	0	1
0	2	0
1	2	0
2	2	0

$R_{crit} \,\&\, R^A$

B\A	0	1
0	-	-
1	-	-
2	-	0

R^B_{LBI}

B\A	0	1
0	0	0
1	0	0
2	0,1	0,1

R^B_{UBI}

B\A	0	1
0	1,2	1,2
1	0,1,2	0,1,2
2	2	2

R^B

B\A	0	1
0	0,1	0,1
1	0	0
2	1,2	1,2

$MAX(R^A, R^B)$

B\A	0	1
0	2	0,1
1	2	0
2	2	1,2

Fig. 8. Example illustrating the computation of multi-valued relations for blocks A and B.

4 Bi-Decomposition Algorithm

This section presents the upper-level procedure that performs one step of recursive bi-decomposition (Fig. 9).

Procedure BiDecompose() is called with the LBI/UBI pair representing a MV relation. It returns a completely specified MV function implemented by the decomposed network of blocks.

If the relation has inessential variables, they are removed using a simple greedy algorithm.

Support size, |S|, is checked for being less than two. If the support size is 0 (or 1), a block representing a constant (or multi-valued literal) is added to the network. Otherwise, the procedure GroupVariables() is called to find sets X_A and X_B, leading to a bi-decomposition with MAX and MIN gates.

Procedure FindBestVariableGrouping() considers the variable sets and determines the best one. The cost function evaluating the

variable sets takes into account two factors: how many variables are included into X_A and X_B (the more, the better), and whether X_A and X_B are well-balanced (the closer their sizes are, the better). The procedure returns the best variable grouping and indicates what decomposition to perform (MAX or MIN).

```
procedure BiDecompose( bdd LBI, bdd UBI )
{  bdd LBI_A, UBI_A, LBI_B, UBI_B, S, F_A, F_B, F;
   ( LBI, UBI ) = RemoveInessentialVariables( LBI, UBI );
   if ( |S| < 2 ) {
      (F, gate) = CreateConstantBlockOrLiteral ( LBI, UBI );
      AddBlockToDecompositionTree( F, gate );
      return F; }
   bdd X_A^OR, X_B^OR, X_A^AND, X_B^AND, X_A^BEST, X_B^BEST;
   ( X_A^OR, X_B^OR ) = GroupVariablesOR( LBI, UBI );
   ( X_A^AND, X_B^AND ) = GroupVariablesAND( LBI, UBI );
    ( X_A^BEST, X_B^BEST, gate ) = FindBestVariableGrouping(
                    (X_A^OR, X_B^OR), (X_A^AND, X_B^AND) );
   if ( (X_A^BEST, X_B^BEST) == ( Ø, Ø ) )
       (X_A^BEST, X_B^BEST, gate) = GroupVariablesWeak( LBI, UBI );
   if ( (X_A^BEST, X_B^BEST) == ( Ø, Ø ) )
        return CompletelySpecifiedFunction( LBI, UBI );
   (LBI_A, UBI_A) = DeriveBlockA( LBI, UBI, X_A^BEST, X_B^BEST , gate);
   F_A = BiDecompose( LBI_A, UBI_A );
   (LBI_B, UBI_B)=DeriveBlockB( LBI, UBI,F_A,X_A^BEST,X_B^BEST, gate);
   F_B = BiDecompose( LBI_B, UBI_B );
   F = Gate( F_A, F_B );
   AddBlockToDecompositionTree( F, gate );
   return F;
}
```

Fig. 9. The pseudo-code of bi-decomposition algorithm.

If variable grouping with non-empty variable sets X_A and X_B is not found, procedure GroupVariablesWeak() finds the best variable grouping for the weak MAX- or MIN- bi-decomposition. If there is no weak decomposition, a completely specified function from the interval (LBI, UBI) is returned. In practice, it happens in less the 10% of cases, and the non-decomposable gate rarely has more than 2 inputs. An example of a weakly-non-decomposable gate is an Exclusive-OR gate for binary relations.

Given the variable sets and the type of decomposition, the LBI/UBI pair for block A are derived using formulas of Section 3. Next, procedure BiDecompose() is called recursively for block A, returning the completely specified function F_A representing one part of the netlist. F_A together with variable sets X_A and X_B is used to compute the LBI/UBI pair of block B. Procedure BiDecompose() is called again for block B.

Finally, multi-valued functions F_A and F_B derived in the process of decomposition and the information about the gate, are used to find the function F implementing the initial relation.

Similar to the case of bi-decomposition for Boolean functions [11], we implemented cache to store the functions represented by the decomposed parts of the network. However, savings due to the component reuse, which were substantial in the binary case, were negligible for MV relations (less than 1% of gates). One of the reasons for this is that there are many different MV literals.

5 Experimental Results

We implemented the bi-decomposition algorithm in a C++ program BI-DECOMP-MV with the BDD package BuDDy [12]. We tested our program on a 933Mhz Pentium III PC under Windows 2000 using multi-valued machine-leaning and data-mining benchmarks available from Portland Logic Optimization Group (POLO) [13].

Experimental results are listed in Table 1. Column "Bmark" gives the benchmark name. "In/Out" is the number of inputs and outputs. "Val" is the sum total of input values. "Cubes" is the number of lines (MV cubes) in the input file. "BDD nodes" is the number of nodes in the shared ROBDD without complement edges representing the BEMDD for LBI and UBI before bi-decomposition. "Reading time" is the time needed to read the input file and derive the BEMDD representation for the problem.

Column "Logic levels" gives the number of levels in the decomposed network. Multi-valued literals are counted as logic blocks. "DFC" gives the Decomposed Function Cardinality measured as the sum total of products of the input-variable cardinalities for all blocks including the multi-valued literals. The section "Gates" gives the number of MAX/MIN gates ("MM"), literals ("Lits"), non-decomposable blocks ("NonDec") and the total number of gates in the decomposed network. Finally, the column "BiDec time" gives the runtime of bi-decomposition.

The results of bi-decomposition for all benchmark functions have been verified by a built-in verifier, which computed a completely-specified MV function representing the decomposed network and checked that it is contained in the interval given by the original relation.

Table 2 compares the experimental results produced by BI-DECOMP-MV with those of a multi-valued bi-decomposer YADE [5] (runtime is measured on a 500MHz Pentium PC). The performance advantages of BI-DECOMP-MV speak for the efficiency of our algorithm and the robustness of BEMDDs.

Table 2. Comparison of decomposition results with YADE.

Bmark	YADE [5]			BI-DECOMP-MV		
	Gates	DFC	Time, c	Gates	DFC	Time, c
balance	268	2012	18	236	1889	0.09
breastc	95	634	24	96	987	0.24
flare1	154	932	11	109	1064	0.14
flare2	300	8049	37	215	6129	0.32
hayes	22	128	1	17	75	0.02

6 Conclusions

We presented a new approach to decompose MV relations into netlists of two-input MV MAX and MIN gates. The decomposition algorithm is based on BEMDDs and formulas with quantifiers evaluated using a standard BDD package.

Our algorithm can be characterized as follows:

- The generated netlists are *compact*, because the algorithm exploits external and internal don't-cares.
- The netlists are *well-balanced* (i.e., the subnetworks for both inputs of a MAX/MIN gate are close in size), which reduces the delay of the resulting circuit.
- It can be shown that the resulting multi-valued netlists are 100% testable for single MV stuck-at faults. A test pattern generation technique can be integrated into the decomposition algorithm with little if any increase in the complexity and runtime.

Future work may include application of the algorithm to datamining and adaptation for MV circuit synthesis. The latter may be achieved by extending the algorithm to work with arbitrary MV gates (not only MAXs and MINs) and integrating decomposition and ATPG.

7 Acknowledgements

The research was sponsored by the NSF grant for the U.S./German collaborative research in functional decomposition for datamining (grant #315/PPP/gű-ab).

Authors would like to thank Christian Lang for helpful comments on implementation and experimental results.

8 References

[1] C. M. Files, M. A Perkowski. Multi-valued functional decomposition as a machine learning method. *Proc. of ISMVL '98*, pp. 173 -178.

[2] MVSIS Group. *MVSIS Manual.* UC Berkeley, February 2001.

[3] M.Perkowski, M.Marek-Sadowska, L.Jozwiak, T.Luba, S.Grygiel, M.Nowicka, R.Malvi, Z.Wang, J.S.Zhang. Decomposition of multiple-valued relations. *Proc. of ISMVL'97,* pp. 13-18.

[4] B. Steinbach, M. A. Perkowski, Ch. Lang. Bi-Decomposition of Multi-Valued Functions for Circuit Design and Data Mining Applications. *Proc. of ISMVL'99*, pp. 50 – 58. http://www.informatik.tu-freiberg.de/prof2/publikationen/ismvl99_final.ps.

[5] Ch. Lang, B. Steinbach. Decomposition of Multi-Valued Functions into Min- and Max-Gates. Accepted to *ISMVL'01*.

[6] C.M. Files, M.A Perkowski. New multi-valued functional decomposition algorithms based on MDDs. *IEEE Trans. CAD*. Vol. 19, No. 9, Sept. 2000, pp. 1081-1086.

[7] C. Files. *A New Functional Decomposition Method as Applied to Machine Learning and VLSI Design*. Ph.D. thesis, Portland State University, Portland, Oregon, June 2000.

[8] A. Mishchenko, C. Files, M. Perkowski, B. Steinbach, C. Dorotska. Implicit algorithms for multi-valued input support minimization. *Proc. Intl. Workshop on Boolean Problems*, Freiberg, September 2000, pp. 9-20.

[9] R. E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation", *IEEE Trans. on Comp.*, Vol. C-35, No. 8 (August, 1986), pp. 677-691.

[10] D.Bochmann, F.Dresig, B.Steinbach, "A new decomposition method for multilevel circuit design". *Proc. of Euro-DAC*, Amsterdam, 1991, pp. 374 – 377.

[11] A. Mishchenko, B. Steinbach, M. Perkowski. "An algorithm for Bi-Decomposition of Logic Functions". Proc. of DAC'01.

[12] J. Lind-Nielsen. *BDD Package BuDDy, v.1.9*, August 2000, http://www.itu.dk/research/buddy/index.html

[13] POLO benchmark functions at http://www.ee.pdx.edu/~polo/ originally coming from the UCI Database http://www.ics.uci.edu/~mlearn/MLRepository.html

Table 1. Results of multi-valued bi-decomposition for POLO benchmarks [13]

Bmark	In/Out	Val	Cubes	Bdd nodes	Reading time,c	Logic levels	DFC	Gates				BiDec time,c
								MM	Lits	NonDec	Total	
audiology	69/1	154	200	12677	0.27	15	23345	94	74	14	182	0.91
balance	4/1	20	625	179	0.02	20	1889	121	115	0	236	0.07
baloon1	4/1	8	16	6	0.01	2	8	1	2	0	0	0.01
breastc	9/1	90	699	4027	0.07	10	987	51	38	7	96	0.17
bridges1	9/1	29	108	616	0.01	11	1469	47	38	5	90	0.06
bridges2	10/1	32	108	815	0.02	13	1889	58	46	7	111	0.10
car	6/1	21	1728	246	0.07	10	372	31	30	0	61	0.02
chess1	6/1	40	28056	15074	1.37	64	$1.7 \cdot 10^6$	5081	3493	716	9290	29.67
chess2	36/1	73	3196	9448	0.84	10	415	68	62	4	134	1.61
cloud	6/1	48	108	621	0.01	9	510	24	16	5	45	0.04
employ1	9/1	27	9600	154	0.51	12	309	26	23	0	49	0.01
employ2	7/1	29	18000	132	0.77	8	375	32	31	0	63	0.01
flag	28/1	133	194	10504	0.13	12	2816	96	75	13	184	0.51
flare1	10/3	33	969	305	0.06	8	1064	56	45	8	109	0.08
flare2	10/3	33	3198	342	0.17	12	6129	113	85	17	215	0.15
hayes	4/1	18	132	122	0.01	5	75	8	9	0	17	0.01
lung-c	56/1	224	32	4171	0.09	8	174	16	14	2	32	0.09
mushroom	22/1	117	8124	1230	1.18	6	32	5	6	0	11	0.03
programm	12/1	42	20000	47581	2.27	49	$3.6 \cdot 10^5$	11160	6621	1502	19283	64.46
sensory	11/1	36	576	3506	0.06	27	$4.5 \cdot 10^4$	633	442	109	1184	0.71
ships	4/1	16	34	105	0.01	8	201	12	8	2	22	0.01
sleep	9/1	83	62	1274	0.02	8	366	17	17	0	34	0.04
sponge	44/1	165	76	3146	0.09	5	41	5	6	0	11	0.06
tic-tac-toe	9/1	27	958	697	0.07	17	1942	274	208	42	524	0.24
train	32/1	105	10	336	0.01	2	8	1	2	0	3	0.01
zoo	16/1	39	101	448	0.02	6	195	8	9	0	17	0.01