ELectronics EXpress

IEICE Electronics Express, Vol.3, No.3, 51–57

Some practical remarks about Binary Decision Diagram size reduction

Piotr Porwik[a)], **Krzysztof Wrobel**[b)], **Piotr Zaczkowski**

Institute of Informatics, University of Silesia

41–200 Sosnowiec ul. Bedzinska 39, Poland

a) *porwik@us.edu.pl*

b) *kwrobel@zsk.tech.us.edu.pl*

Abstract: In this paper the new spectral method of the variable ordering in the Binary Decision Diagram (BDD) is introduced. The problem to find the best order is defined as NP-hard, and optimal solution can not be finding because optimizing algorithm is running in $O(n!2^n)$ time, and calculations can not be done for the large n. Such complexity was obviously improved in many algorithms but as will be demonstrated in this paper size of the BDD can be additionally reduced. Performed results with another representative methods have been compared.

Keywords: Boolean function, Binary Decision Diagram, Walsh transform

Classification: Science and engineering for electronics

References

[1] B. J. Falkowski, I. Schafer, and M. A. Perkowski, "Effective Computer Methods for the Calculation of Rademacher-Walsh Spectrum for Completely and Incompletely Specified Boolean Functions," *IEEE Trans. Computer-Aided Design*, vol. 11, no. 10, pp. 1207–1226, 1992.

[2] S. J. Friedman and K. J. Supowit, "Finding the Optimal Variable Ordering for Binary Decision Diagrams," *IEEE Trans. Comput.*, vol. 39, no. 5, pp. 710–713, 1990.

[3] J. Lind-Nielsen. BuDDy – Binary Decision Diagrams Library Package v2.3. University of Copenhagen, Denmark, Nov. 2002.

[4] C. Meinel, F. Somenzi, and T. Theobald, "Linear sifting of decision diagrams," *Proc. of the 34th Design Automation Conference*, DAC'96, pp. 202–207, 1997.

[5] P. Porwik, "Efficient spectral method of identification of linear Boolean function," *Control and Cybernetics*, vol. 33, no. 4, pp. 663–678, 2004.

[6] R. Rudell, "Dynamic variable ordering for ordered binary decision diagrams," *Proc. IEEE/ACM ICCAD'98 Conf.*, pp. 42–47, 1993.

[7] F. Somenzi, BDD Package: CUDD v.2.3.0. http://vlsi.colorado.edu/~fabio/CUDD/cuddIntro.html, 2003.

[8] R. S. Stanković and J. T. Astola, *Spectral Interpretation of Decision Diagrams*, New York: Springer-Verlag, 2003.

[9] I. Wegener, *Branching Programs and Binary Decision Diagrams. Theory*

© IEICE 2006

DOI: 10.1587/elex.3.51
Received January 04, 2006
Accepted January 16, 2006
Published February 10, 2006

and Applications, Philadelphia: SIAM Monographs, 2000.

1 Introduction and preliminaries

Decision Diagrams are the state-of-the-art representation for Boolean function in *Computer-Aided Design* applications (CAD) [7, 8, 9]. For functions of a few variables, such structures can be presented as trees or, after reduction, as diagrams. In reality, Decision Diagrams (DD) are data structures for discrete functions representation. Decision diagrams are stored in the memory of a computer as a hash tables (unique table) specifying nodes and interconnections among them. Representation of a Boolean function by means of BDD is very efficient, because building of diagram is carried out in memory, where only address and pointers of diagram nodes are located. Hence, BDDs algorithms are very efficient in terms of space and time, especially when BDDs describe large discrete functions. It can be noticed that DD can be constructed for completely and incompletely defined functions. If function is completely defined, efficiency of DDs is similar as FFT-base methods, but unlike FFT description, DD can be additionally optimized, what will be explain below. If an n variable Boolean function is weakly defined, especially when n is large, DD manipulation is very convenient [8, 9]. There are many cases where conventional algorithms can be significantly improved by BDDs [5, 7, 8, 9]. For example, computations even with the fast transforms can be difficult, because the truth-table of Boolean functions grows exponentially with n. Additionally, several variants of DDs are developed to represent other kinds of discrete functions or function properties such as: multi-valued functions, Reed-Muller forms, arithmetic formulas, calculation of spectral coefficients, cube sets, etc. [8]. From this reason, methods based on the BDDs are preferred.

The BDD is a canonical representation of an n-variable Boolean function for fixed order of input variables. However, the permutation of the variable order very often gives different BDD for the same a Boolean function. Hence, in many cases the size of the BDD can be significantly different. The efficiency of the BDD representation is determined by the size of the BDD defined as the number of nodes in the BDD [2, 9]. The size of BDD can be changed by means of variable ordering and it is continuously an important problem in the BDD techniques. The size of the BDD for a given Boolean function is sensitive to the choice of an ordering on the variables. For example the part of adder, which form carry-out signal, can be represented by n BDD nodes if ordering is optimal – but if order is another, the BDD has $2^{n/2}$ nodes [2]. Although the efficient algorithms of optimal variable ordering do not exist, some approaches give much better results than consideration of all $n!$ ordering. The algorithm with the best worst-case runtime was introduced in [2], where time complexity of variable ordering is $O(n^2 3^n)$ and space complexity is $O(3^n/\sqrt{n})$. Currently, the most popular dynamic reordering technique is sifting algorithm, which

© IEICE 2006
DOI: 10.1587/elex.3.51
Received January 04, 2006
Accepted January 16, 2006
Published February 10, 2006

was proposed by Rudell [6]. In this algorithm variables of function are ordered in natural order $(x_1, x_2, \ldots x_n)$. In next stage each variable is moved trough the variable ordering using swaps. The sifting algorithm tries to find a good variable ordering of a BDD by successive analyzing each variable, starts from initial order. The investigated variable is moved through the whole ordering. Finally, the variable is moved to its optimal position. Because each variable is moved only once, in sifting procedure only n^2 swaps is needed [6]. The sifting algorithm is stopped if so called increase factor c is achieves [6, 9]. If c is small algorithm works faster but if c-value is large, more possibilities of ordering can be explored. Nowadays, the modified sifting algorithm, called "linear sifting" is used in academic C++ language packages, described among other things in [3, 4]. If initial order of variables will be fixed differently, then size of BDD can be additionally decreased. The new initial pre-order of the variables can be found by means of the spectral analysis of the function f.

2 The first order Walsh coefficients calculation

The Boolean function $f(x_1, x_2, \ldots, x_n)$ given by the binary truth-vector $Y_f = [y_0, y_1, \ldots, y_{2^n-1}]^T$, can be transformed from the Boolean domain $\{0, 1\}$ into the spectral domain by a linear transformation $H \cdot Y_f = S$, where H is a $2^n \times 2^n$ transform matrix, and $S = [s_0, s_1, \ldots, s_{2^n-1}]^T$ is the vector of spectral coefficients called the spectrum of f. In particular, we get the Walsh-Hadamard transform when H is the Walsh matrix defined as:

$$W(n) = \bigotimes_{i=1}^{n} W(i), \quad W(1) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{1}$$

where: $W(i)$ is the basic Walsh matrix and $\otimes$ denotes the Kronecker product [8]. Hence, the forward and inverse Walsh transform can be described as follows:

$$S(n) = W(n) \cdot Y(n), \quad Y(n) = 2^{-n} W(n) \cdot S(n) \tag{2}$$

Taking into account properties of the Walsh transform (1), (2), an n-variable Boolean function f can be described by means of the spectral coefficients:

$$f(x_1, \ldots, x_n) = \frac{1}{2^{n+1}} \left[2^n - s_0 - \sum_{a=1}^{2^n-1} s_a (-1)^{x_1^{a_1} \oplus x_2^{a_2} \oplus \ldots \oplus x_n^{a_n}} \right] \tag{3}$$

where: $a_1, a_2, \ldots a_n \in \{0, 1\}$ – the binary representation of the decimal number a and $x^{a_i=0} = 0$, and $x^{a_i=1} = x$. From (3) it can be observed, that the spectral coefficient can be described by indexes which are correlated with variable of an function f. For example, $s_1 \leftrightarrow x_1$, $s_{12} \leftrightarrow x_1 \oplus x_2$, $s_{123} \leftrightarrow x_1 \oplus x_2 \oplus x_3$, $s_{12\ldots n} \leftrightarrow x_1 \oplus x_2 \oplus \ldots \oplus x_n$. The spectral coefficients, where index is describe by one number, are called coefficients of the first order coefficients, when index has two numbers – are the second order, and so on for the third, forth, ..., n^{th} order. The first order spectral coefficients of a given Boolean function f can be calculated immediately from some rows of the matrix H [8]. Unfortunately, such calculation can be used for fully defined functions. From this reason the values of the first order spectral coefficients

© IEICE 2006
DOI: 10.1587/elex.3.51
Received January 04, 2006
Accepted January 16, 2006
Published February 10, 2006

have been introduced on the basis of method described in [1]. Such method is very convenient for large Boolean functions because instead conventional method of spectra calculation, where large $2^n \times 2^n$ the Walsh matrix should be generated, cubes representation can be used. The mentioned method can be modified and suitably adopted, in such case the first order coefficients can be calculated with the aid of the algorithm:

The first order spectral coefficients generation algorithm.
Input: the $*$.pla file with ON/DC cubes of an n-variable Boolean function f
Output: The table S includes the first order Walsh coefficients $S[1], S[2], \ldots, S[n]$

```
for (i=1; i<=n; i++) S[i]=0;
for (j=1; j<=number of cubes; j++)
{
```

step I *Calculation of the number of places, where occurs don't care symbol (–)*

```
  p=0;
  for (i=1; i<=n; i++)
    if ( the ith bit in current cube = '-') p++;
```

step II *Calculation of the first order spectral coefficient*

```
  for (i=1; i<=n; i++)
  {
    if (the current cube is ON - type)
    { vi=2^(p+1);
      if (the ith bit in the current cube = 0) S[i]=S[i]-vi;
      if (the ith bit in the current cube = 1) S[i]=S[i]+ vi; }
    else
```

/∗ for the DC- type cube ∗/

```
    { vi=2^p;
      if (the ith bit in the current cube = 0) S[i]=S[i]-vi;
      if (the ith bit in the current cube = 1) S[i]=S[i]+ vi; }}
}
```

It can be checked that the presented algorithm has the time complexity $O(cn)$, where c is a number of disjoint cubes describe the Boolean function f.

3 A new method of the BDD size reduction

Optimization of BDDs can be performed by ordering of variables because different orderings produce different BDDs. They are many strategies of variable ordering: local computability, level heuristics, fan-in heuristics, etc. [9]. It

© IEICE 2006
DOI: 10.1587/elex.3.51
Received January 04, 2006
Accepted January 16, 2006
Published February 10, 2006

can be noticed that the freely available CUDD and BuDDy packages [3, 7] exploit sifting or linear sifting algorithms [4] as the method of the BDD size reduction. The ordering can be treated as suitable permutation of variables. Permutation of variables in f, corresponds to the elements permutation in the truth-vector Y_f. Similar permutation of Y_f can be obtained by use of special permutation matrices [8], but its construction seems to be difficult. Moreover such matrices are large for large Boolean functions. In this paper the new spectral initial variable ordering has been proposed. The modification can be treated as the method, where the influence of variables on control of function is determined, hence such variables should be at higher position in the BDD.

In the first stage, the set of the first order spectral coefficients $\{s_1, s_2, \ldots, s_n\}$ for a given function f is determined on the basis of above presented algorithm.

- In the next step, the obtained values in different manner can be ordered:

Table I. The first order spectral coefficients pre-ordering principles

Type of order	Symbol	Example
The descending order	$>$	$s_1 > \ldots > s_{n-1} > s_n$
The ascending order	$<$	$s_1 < \ldots < s_{n-1} < s_n$
The absolute descending order	$\|>\|$	$\|s_1\| > \ldots > \|s_{n-1}\| > \|s_n\|$
The absolute ascending order	$\|<\|$	$\|s_1\| < \ldots < \|s_{n-1}\| < \|s_n\|$

- In the another stage the values of the coefficients are ordered according to Table I. Because each the spectral coefficient s_i, $i = 1, 2, \ldots, n$ is connected with the variable x_i [1, 5, 8] – order of all variables is unambiguously also appointed.

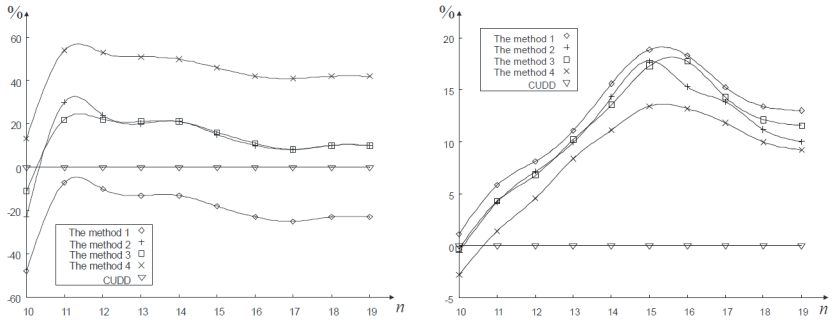
4 Proposed approach and investigation results

Because sifting procedure is commonly used as efficient method of the BDD size reduction, experiments with sifting and new technique implemented together with sifting algorithm have been conducted. In this paper the linear sifting procedure is used because it gives better results than sifting method [4, 9]. The new initial input variables order was implemented inside of the appropriate $*$.pla-type file. Each input file has the Espresso format. For example:

© IEICE 2006
DOI: 10.1587/elex.3.51
Received January 04, 2006
Accepted January 16, 2006
Published February 10, 2006

Input file:	Reordered file:	Comment:
`.i 2`	`.i 2`	For the truth-vector $Y_f = [0,0,1,1]$ the first spectral coeff. are $s_1 = 0$ and $s_2 = -2$ ($s_1 > s_2$). Hence, input variables there are in descending order. If assume another order, for example $s_1 < s_2$, then the new order of the variables in input file is stated. The new file in the CUDD package will be used and new order is stored in computer memory.
`.o 1`	`.o 1`	
`00 0`	`00 0`	
`01 0`	`10 0`	
`10 1`	`01 1`	
`11 1`	`11 1`	
`.e`	`.e`	

From this reason, suitable variable pre-ordering inside file $*$.pla can be treated as initial order of linear sifting algorithm. Hence, original ordering algorithm does not have to be changed. The calculations by means of the CUDD have been evaluated. The size reduction of the BDDs and the calculation time have been compared. For all type of orders, the appropriate BDD has been constructed and its size has been determined. The method by means of 300 randomly generated single output, full defined Boolean functions has been tested. For different orders the minimal BDD size has been chosen: the pre-ordering method 1: $min\{>,<,|>|,|<|\}$, the pre-ordering method 2: $min\{>,<\}$, the pre-ordering method 3: $min\{|>|,|<|\}$, the pre-ordering method 4: $random\{>,<,|>|,|<|\}$, where notation min ($random$) indicates the minimal number of BDD nodes from the appropriate set of orders. The results of investigations for each method present Fig. 1, where time as well as the BDDs size reduction has been performed. The efficiency of the method on a percentage basis has been expressed. From Fig. 1 follows, that the proposed method gives better size and time results than the CUDD method. The results in most cases are better even random variables ordering is chosen. Such result follows from assumption, that in sifting algorithm instead $n!$ brute-force exhaustive searches, only n^2 variable ordering is analyzed, hence only sub-optimal order will be found. Sifting algorithm starts from lexicographical order of variables $(x_1, x_2, \ldots, x_n)$, but on the basis of spectral coefficients analysis, better initial order is often proposed (for exam-



(a) (b)

Fig. 1. The CUDD linear sifting (without pre-ordering) with relation to CUDD with different pre-ordered input files: (a) reduction of the time calculation and (b) the BDD size reduction

© IEICE 2006
DOI: 10.1587/elex.3.51
Received January 04, 2006
Accepted January 16, 2006
Published February 10, 2006

ple $x_n, x_1, \ldots x_2$). It can be noticed that CUDD package has a conjectural internal limitation ($n < 20$), for full defined, single output functions. Therefore additionally, some representative weakly defined, multi-output benchmarks have also been tested, what presents Table II.

Table II. Effect of the variables ordering: The CUDD with linear sifting (without pre-ordering), and with additional pre-ordering: the number of the nodes in BDD and calculation time (nodes/time in seconds.)

Circuit	#i	#o	CUDD/s	>/s	|>|/s	</s	|<|/s	*min/tot.s*
9sym	9	1	25/.02	25/.02	25/.02	25/.02	25/.00	25/.06
co14	14	1	27/.02	27/.02	27/.02	27/.00	27/.00	27/.04
dist	8	5	121/.02	121/.02	121/.02	121/.02	121/.02	121/.08
ex1010	10	10	1060/.14	1058/.08	1054/.08	1056/.08	1050/.08	1050/.32
ex5	8	63	242/.05	186/.02	205/.03	204/.03	181/.02	181/.01
majority	5	1	8/.02	8/.02	8/.02	8/.02	9/.00	8/.06
mlp4	8	8	135/.05	145/.05	151/.05	151/.05	145/.05	145/.20
sqar5	5	8	35/.02	33/.02	34/.00	33/.02	33/.02	33/.06
sym10	10	1	31/.03	31/.02	31/.02	31/.02	31/.02	31/.08
Total			1684/.37	1634/.27	1656/.26	1656/.26	1622/.21	1621/.91

This type of investigations was performed similarly as previously, where first adequate variables order in benchmarks file was changed, and in next stage CUDD procedure is activated. Taking into account values from Table II can be observed that the new initial useful pre-ordering technique in the $*$.pla file, together with linear sifting algorithm gives better results with relation to singly used linear sifting algorithm. Should be noticed that only for benchmark mlp4.pla results of the BDD optimization were worse than the CUDD approach. In another cases even choice any pre-ordered method gives better results to compared with method where pre-ordering was not used. Can be observed that preliminary initial input file reorder before starts of CUDD, gives better results than CUDD without input file rebuilding. The symbols in columns 4–9 have the next meaning: number of BDD nodes/time of calculations in seconds. All experiments with Athlon XP 1700+ (1466 MHz) processor and 512 Mbytes main memory were performed.

5 Conclusion

In this paper the auxiliary, effective method of the BDDs ordering has been introduced. The proposed method, based on the first spectral coefficients analysis, can be used in the BDD optimization. The initial pre-ordering by input file conversion is achievement. The described approach gives often better results (time calculations, and the BDD size reduction) than method where only linear sifting algorithm is applied.

© IEICE 2006
DOI: 10.1587/elex.3.51
Received January 04, 2006
Accepted January 16, 2006
Published February 10, 2006