The Use of Autocorrelation Coefficients for Variable Ordering for ROBDDs

Jackie Rice, Micaela Serra and Jon Muzio

Department of Computer Science
University of Victoria
Victoria, B.C., Canada

Abstract

We present a method of ordering variables for logic functions based on the function's autocorrelation coefficients. This ordering is used in the building of a Reduced Ordered Binary Decision Diagram (ROBDD) representing the function. The ordering allows optimal, or close to, sharing of subtrees in the ROBDD to be possible, thus creating a close to minimal ROBDD. This is tested in the context of mapping the function to lookup table-based Field Programmable Gate Arrays using a mapper that makes use of the ROBDD representation.

1 Introduction

There are many areas of research that require that a logic function be represented in some intermediate form before being mapped to a final use. For example, Field Programmable Gate Array (FPGA) synthesis requires that a function be optimized in a number of ways before being mapped to the elements of the FPGA. One format that has been used for many years is the ROBDD.

Reduced Ordered Binary Decision Diagrams, or *ROBDDs* are a canonical form of Binary Decision Diagrams, or *BDDs*. BDDs represent a function's truth table by having a leaf node for every possible combination of values that the input variables can hold. ROBDDs attempt to reduce this exponential size by sharing similar parts of the tree. This sharing, however, can depend heavily on the order in which variables are encountered. Therefore it is desirable to overcome this one disadvantage by determining a method to optimize the selection of the variable ordering.

In this paper, we examine the information present in a function's autocorrelation coefficients, and use this information to generate an ordering that results in a small, or optimal ROBDD. This work is investigated in the context of mapping a logic function to a lookup table (LUT)-based Field Programmable Gate Array (FPGA). Since an ROBDD can be segmented into a combination of LUTs, this means that the smaller the ROBDD, the smaller the area (and fewer LUTs required) of the FPGA.

2 Background

2.1 ROBDDs

One of the most commonly used representations of a Boolean function is the Binary Decision Diagram, or *BDD*. A BDD is a graph representation of a Boolean function that is defined as [3]:

> "a binary directed acyclic graph with two leaves *TRUE* and *FALSE*, in which each non-leaf node is labeled with a variable and has two out-edges, one pointing to the subgraph that is evaluated if the node label evaluates

to *TRUE* and the other pointing to the subgraph that is evaluated if the node label evaluates to *FALSE*."

This is shown as a network of nodes where each node represents a variable in the logic function being described. If the variable has the value 0, the edge marked "0" leading from that node in the graph is followed. If the variable has the value 1 the edge marked "1" is followed. Edges in BDDs do not have explicitly marked direction; the direction is implied.

BDDs are often used in their canonical form. This is a simplified form called a Reduced Ordered Binary Decision Diagram, or *ROBDD*. A ROBDD is a reduced BDD with a specified ordering of variables. It must meet two main specifications:

- it contains no vertex whose left subdiagram is equal to its right subdiagram, nor does it contain distinct vertices v and v' such that the subdiagrams rooted by v and v' are isomorphic;
- every path on the root node to an output must encounter the variables of the function in the specified order.

An example BDD and ROBDD are shown in Figure 1 for the variable ordering A, B, C. In this figure, and all future figures of BDDs, the path to follow if the node evaluates to a "0" is marked by a dotted line, while the path to follow if the node evaluates to a "1" is marked with a solid line. Edges to the final outputs may be shared in all ROBDDs; however, in order to simplify the diagrams, this is not always shown.

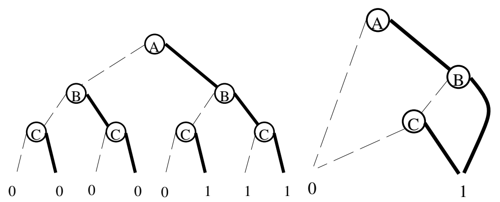

Figure 1: the BDD and ROBDD for the function $f = AB + AC$

ROBDDs are a good choice for representing functions for a number of reasons [2]:

- most commonly encountered functions have reasonable ROBDD representations;
- for many practical functions the time complexity of any single operation is bounded by the product of the graph sizes for the functions being operated on; and
- ROBDDs are a canonical representation.

The major drawback of this representation is that for some functions, the size of the ROBDD representing the function is highly sensitive to the ordering. An example of this is shown in Figures 2 and 3 for the function $f_1 = x_1x_2 + x_3x_4 + x_5x_6$.

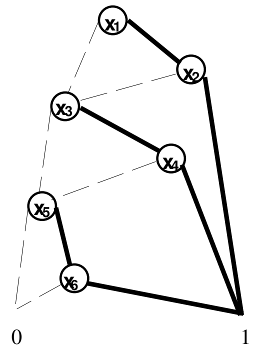

Figure 2: The ROBDD for $f_1 = x_1x_2 + x_3x_4 + x_5x_6$ using the variable ordering $x_1, x_2, x_3, x_4, x_5, x_6$.

Finding the optimal variable ordering starting from a given BDD representation is known to be NP-hard [1]. It is therefore desirable to find a relatively fast method to determine an ordering that results in either the smallest possible ROBDD or one that is close to this.

2.2 Autocorrelation Coefficients

Functions can also be translated to other domains, such as the *spectral domain*. In this paper we make use of the function's *autocorrelation coefficients* which are derived from the function's spectral coefficients.

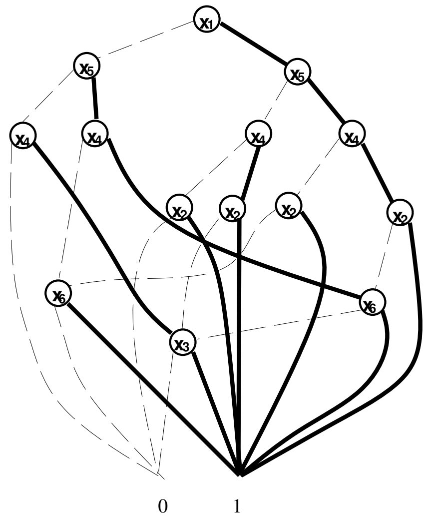

Figure 3: The ROBDD for $f_1 = x_1x_2 + x_3x_4 + x_5x_6$ using the variable ordering $x_1, x_5, x_4, x_2, x_6, x_3$.

The general cross-correlation (convolution) function between two given functions f and g at a distance u is defined as

$$\sum_{v=0}^{2^n-1} f(v)g(v \oplus u) \qquad (1)$$

for two functions $f(X)$ and $g(X)$ where $X = x_n x_{n-1} ... x_1$. The symbol $\oplus$ represents the bitwise exclusive-or function. When $f = g$ the resulting equation gives the cross-correlation of a function with itself, translated by u. The resulting coefficients are referred to as the autocorrelation coefficients of the function.

The autocorrelation function is defined as follows:

$$B(u) = \sum_{v=0}^{2^n-1} f(v)f(v \oplus u) \qquad (2)$$

where the function is $f(X)$, $X = x_n x_{n-1} ... x_1$, $u = \sum_{i=1}^{n} u_i 2^{i-1}$, $v = \sum_{i=1}^{n} v_i 2^{i-1}$, and n is the number of inputs [6].

For multiple output functions a second step must be performed, to combine the autocorrelation function for each of the individual functions into the *total autocorrelation function.* The total autocorrelation function is defined as

$$B(u) = \sum_{i=0}^{m-1} B_i(u) = \sum_{i=0}^{m-1} \sum_{v=0}^{2^n-1} f_i(v) f_i(v \oplus u) \qquad (3)$$

where m is the number of outputs and the multiple-output function F consists of $f_0 f_1 ... f_{m-1}$.

Using Equation 3 to calculate the autocorrelation coefficients for a function requires $O(m2^{2n})$ operations. However, there is a significantly faster method for calculating the autocorrelation coefficients. Using the spectrum $\mathbf{R}_i$ of the function $f_i \in +1, -1$, as calculated using the Rademacher-Walsh or another appropriate transform $\mathbf{T}^A$ where $\mathbf{B}_i = \mathbf{T}^n \mathbf{f}_i$, then the autocorrelation function can be computed as

$$\mathbf{B}_i = \frac{1}{2^n} \mathbf{T}^n \mathbf{R}_i^2 \qquad (4)$$

${\mathbf{R}_i}^2$ is defined as the spectral vector $\mathbf{R}$ for the ith function output with each term squared. Each corresponding element of $\mathbf{B}_i$ is then summed over the number of outputs to calculate the total autocorrelation coefficients, $\mathbf{B}$. Using this method requires $O(mn2^n)$ operations, assuming a fast method is used to calculate $\mathbf{R}_i$[6]. However, we use only a small subset of the total set of coefficients ($\frac{n(n+1)}{2}$ out of 2^n), so it may be possible to evaluate our required subset even more efficiently.

3 Conjecture: can a good ordering of variables be found using a function's autocorrelation coefficients?

The autocorrelation coefficients of a function f give a measure of the similarity between f and the same function translated by a distance u. When this n-tuple u contains very few "1"s, the distance (u) is small. We conjecture that the corresponding first and second-order autocorrelation coefficients potentially provide information useful in determining a good variable ordering for the ROBDD representation.

We illustrate the nature of this conjecture with some examples.

Example 1
Initially we examine a short example that briefly outlines the idea. Given the function $f = \overline{x}_2\,\overline{x}_4 + x_2\overline{x}_3x_4 + x_1x_2x_4 + \overline{x}_1x_3\overline{x}_4$ the autocorrelation coefficients are shown in Figure 4. In each column of coefficients the value in brackets on the left is the variable(s) corresponding to the coefficient value, which is given on the right. The coefficient labeled as (0) determines the number of minterms for the function, and is not a first-order coefficient, although it is listed under this column.

first–order coefficients	second–order coefficients	third–order coefficients	fourth–order coefficients
[(0)=8]			
(1)=6	(12)=2	(123)=2	(1234)=6
		(124)=6	
	(13)=6	(134)=2	
	(14)=2		
(2)=2	(23)=2	(234)=6	
	(24)=6		
(3)=6	(34)=2		
(4)=0			

Figure 4: The autocorrelation coefficients for the function $f = \overline{x}_2\,\overline{x}_4 + x_2\overline{x}_3x_4 + x_1x_2x_4 + \overline{x}_1x_3\overline{x}_4$.

In this example the first-order coefficients give an indication that pairing the variables x_1 and x_3 should result in more sharing of subdiagrams in the resulting ROBDD. We reach this conclusion by examining the first-order coefficients of the function and identifying the variables with large first-order autocorrelation coefficients.

In the second-order autocorrelation coefficients we see that the value of the coefficient for the pairings of x_1x_3 is also of a high magnitude, thus giving another reason to group together these variables. However, another second-order coefficient also has the value 6, indicating that possibly the variables x_2 and x_4 should be paired together.

In fact the table of all possible orderings (Table 1) shows that all those resulting in a minimal ROBDD (5 nodes)[1] have the variables x_1 and x_3 adjacent to each other. There is no ordering with these variables grouped together that has more than 6 nodes. However, the other pairing suggested by the autocorrelation coefficients, x_2x_4 seems to have little or no effect on the size of the resulting ROBDD.

ordering	number of nodes	ordering	number of nodes
1 2 3 4	6	1 2 4 3	7
1 3 2 4	5	1 3 4 2	6
1 4 2 3	7	1 4 3 2	6
4 2 3 1	5	4 2 1 3	5
4 3 2 1	6	4 3 1 2	5
4 1 2 3	6	4 1 3 2	5
3 2 1 4	6	3 2 4 1	7
3 1 2 4	5	3 1 4 2	6
3 4 2 1	7	3 4 1 2	6
2 1 3 4	5	2 1 4 3	6
2 3 1 4	5	2 3 4 1	6
2 4 1 3	6	2 4 3 1	6

Table 1: All possible orderings for four inputs, with the corresponding number of nodes for the ROBDD for the function $f = \overline{x}_2\,\overline{x}_4 + x_2\overline{x}_3x_4 + x_1x_2x_4 + \overline{x}_1x_3\overline{x}_4$.

It appears from this example that choosing the ordering of variables based on the magnitude of the autocorrelation coefficients does have some relevance to the size of the resulting ROBDD. The following example explores this in more detail.

Example 2
In this example we examine the function $f = x_1x_2x_3 + x_2x_3x_4$. The autocorrelation coefficients for this function are shown in Figure 5.

The autocorrelation coefficients have high magnitude values for the variables x_4 ($u = 1000$), x_1 ($u = 0001$), and the pair of variables x_1x_4 ($u = 1001$).

By using all possible orderings to construct the ROBDD it is shown that orderings in which x_1 and x_4 are adjacent result in ROBDDs with only 5 nodes (see Table 2). One such ordering is shown in Figure 6.

[1]Note that to calculate this number we assume one node for each decision point in the ROBDD plus one node that is shared between all outputs; that is, terminal nodes labeled 1 and/or 0.

first–order coefficients	second–order coefficients	third–order coefficients	fourth–order coefficients
[(0)=3]			
(1)=2	(12)=0	(123)=0	(1234)=0
		(124)=0	
	(13)=0	(134)=0	
	(14)=2		
(2)=0	(23)=0	(234)=0	
	(24)=0		
(3)=0	(34)=0		
(4)=2			

Figure 5: The autocorrelation coefficients for the function $f = x_1x_2x_3 + x_2x_3x_4$.

ordering	number of nodes	ordering	number of nodes
1 2 3 4	7	1 2 4 3	6
1 3 2 4	7	1 3 4 2	6
1 4 2 3	5	1 4 3 2	5
4 2 3 1	7	4 2 1 3	6
4 3 2 1	7	4 3 1 2	6
4 1 2 3	5	4 1 3 2	5
3 2 1 4	5	3 2 4 1	5
3 1 2 4	6	3 1 4 2	5
3 4 2 1	6	3 4 1 2	5
2 1 3 4	6	2 1 4 3	5
2 3 1 4	5	2 3 4 1	5
2 4 1 3	5	2 4 3 1	6

Table 2: All possible orderings for four inputs, with the corresponding number of nodes for the ROBDD for the function $f = x_1x_2x_3 + x_2x_3x_4$.

Analysis and Discussion

The first-order autocorrelation coefficients, those for which $|u| = 1$, measure the similarity of a function with the same function, but with the relevant variable inverted. That is, in a 4 variable situation with $u = 0001$, the corresponding autocorrelation coefficient measures the similarity between $f(x_4, x_3, x_2, x_1)$ and $f(x_4, x_3, x_2, \overline{x}_1)$. Similarly, the second-order coefficient, with, say, $u = 0011$ measures the correlation between $f(x_4, x_3, x_2, x_1)$ and $f(x_4, x_3, \overline{x}_2, \overline{x}_1)$.

Consequently higher magnitude coefficients give us some useful information concerning potential identical subfunctions which might be used in a ROBDD. These indicators will, however, never be foolproof since they depend on the precise distribution of the 1's in the function. This detailed information is lost by just

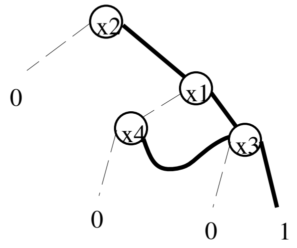

Figure 6: ROBDD for the function $f = x_1x_2x_3 + x_2x_3x_4$ with ordering determined by examining the autocorrelation coefficients.

considering the first and second-order coefficients.

Figure 7 shows how the identical subfunctions make themselves apparent when ordering according to the magnitudes of the first and second-order coefficients as shown in Figure 5.

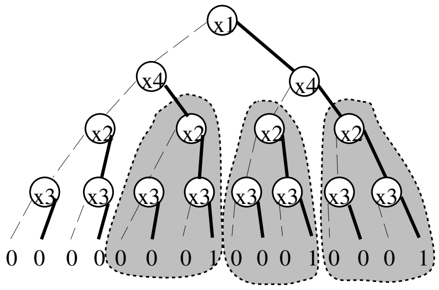

Figure 7: The reordered BDD for $f = x_1x_2x_3 + x_2x_3x_4$.

In this BDD the variables have been ordered so that the similarities present in the function are grouped together. The three shaded subdiagrams have identical terminals, and can therefore be shared when reducing the BDD.

Example 3

The third example is $f = x_1x_2\overline{x}_3 + \overline{x}_1x_2x_3 + \overline{x}_1x_3x_4$, where the autocorrelation coefficients for this function are shown in Figure 8.

Here, the largest first-order autocorrelation coefficients are for the variables x_4 and x_2. The largest second-order coefficient is for the variable pairings x_1x_3, with the pairing x_2x_4 having the second largest second-order coefficient. This suggests grouping x_2 and x_4 and grouping

first–order coefficients	second–order coefficients	third–order coefficients	fourth–order coefficients
[(0)=5]			
(1)=0	(12)=0	(123)=2	(1234)=2
		(124)=0	
	(13)=4	(134)=4	
	(14)=0		
(2)=2	(23)=0	(234)=0	
	(24)=2		
(3)=0	(34)=0		
(4)=4			

Figure 8: The autocorrelation coefficients for the function $f = x_1x_2\overline{x}_3 + \overline{x}_1x_2x_3 + \overline{x}_1x_3x_4$.

x_1 and x_3, but it is not clear as to which pairing should have the highest priority.

Table 3 shows all possible orderings and the corresponding ROBDD size.

ordering	number of nodes	ordering	number of nodes
1 2 3 4	7	1 2 4 3	6
1 3 2 4	7	1 3 4 2	6
1 4 2 3	6	1 4 3 2	7
4 2 3 1	7	4 2 1 3	7
4 3 2 1	7	4 3 1 2	8
4 1 2 3	7	4 1 3 2	8
3 2 1 4	7	3 2 4 1	6
3 1 2 4	7	3 1 4 2	6
3 4 2 1	6	3 4 1 2	7
2 1 3 4	7	2 1 4 3	6
2 3 1 4	7	2 3 4 1	6
2 4 1 3	6	2 4 3 1	6

Table 3: All possible orderings for four inputs, with the corresponding number of nodes for the ROBDD for the function $f = x_1x_2\overline{x}_3+\overline{x}_1x_2x_3+\overline{x}_1x_3x_4$.

In this table, the smallest ROBDD has 6 nodes. For most of the orderings resulting in ROBDDs with 6 nodes, the variables x_2 and x_4 are paired together. Out of 24 orderings, there are 10 that result in ROBDDs with 6 nodes. Of these 10 orderings, 8 contain the pairing x_2x_4 or x_4x_2, and 4 contain the pairing x_1x_3 or x_3x_1. Two of the 10 have neither of these pairings.

Example 4

The final example is $f = x_1x_3 + x_2x_4$. with autocorrelation coefficients for as shown in Figure 9.

This function differs from the previous functions in that the first-order autocorrelation coefficients are all of the same magnitude, thus

first–order coefficients	second–order coefficients	third–order coefficients	fourth–order coefficients
[(0)=7]			
(1)=4	(12)=2	(123)=2	(1234)=2
		(124)=2	
	(13)=4	(134)=2	
	(14)=2		
(2)=4	(23)=2	(234)=2	
	(24)=4		
(3)=4	(34)=2		
(4)=4			

Figure 9: The autocorrelation coefficients for the function $f = x_1x_3 + x_2x_4$.

ordering	number of nodes	ordering	number of nodes
1 2 3 4	7	1 2 4 3	7
1 3 2 4	5	1 3 4 2	5
1 4 2 3	7	1 4 3 2	7
4 2 3 1	5	4 2 1 3	5
4 3 2 1	7	4 3 1 2	7
4 1 2 3	7	4 1 3 2	7
3 2 1 4	7	3 2 4 1	7
3 1 2 4	5	3 1 4 2	5
3 4 2 1	7	3 4 1 2	7
2 1 3 4	7	2 1 4 3	7
2 3 1 4	7	2 3 4 1	7
2 4 1 3	5	2 4 3 1	5

Table 4: All possible orderings for four inputs, with the corresponding number of nodes for the ROBDD for the function $f = x_1x_3 + x_2x_4$.

providing no useful information as to a good ordering. However, in the second-order coefficients there is more information. Here we see that the pairings x_1x_3 and x_2x_4 have larger magnitude coefficients than the rest of the pairings. This would suggest that to result in a smaller ROBDD, a variable ordering should consist of some combination of these two pairs. Once again, Table 4 shows all possible orderings and the corresponding ROBDD size.

In Table 4 it is clear that the smallest ROBDD has 5 nodes. We note that every ordering consisting of some combination of the variables pairs x_2x_4 and x_1x_3 results in 5 nodes. Moreover, every ordering with only 5 nodes consists of some ordering in which x_2 and x_4 are paired and x_1 and x_3 are paired.

While it is clearly possible in some specific cases to see the potential value of using the third- (or higher) order coefficients, it is not clear as to how it might be possible to extract such useful information without the process becoming computationally unreasonable (there are, after all $m!$ possible rearrangements of the variables within the grouping given by an mth-order coefficient).

4 Experimental Results

The experimental procedure for examining the validity of our conjectures used a number of different ordering procedures, each based on the autocorrelation coefficients. The autocorrelation coefficients were calculated by a program called *acspec* [7]. The ordering procedures were then applied to Item [4], a system for logic synthesis for FPGAs that makes use of ROBDDs.

Item was chosen for this work because the internal representation was ROBDDs, and the mapping algorithm Xmap *does not modify the underlying structure of the ROBDD* [5]. Thus finding a smaller ROBDD representation should result in fewer combinational logic blocks (CLBs) being required in the FPGA mapping of the function. The CLBs in this case were lookup tables, as the FPGAs being mapped to were lookup table (LUT)-based FPGAs. Both the number of ROBDD nodes and number of CLBs in the FPGA mapping were used as measures in this work.

The ordering methods used are described below.

- Ordering Method 1 - use only the first-order coefficients to determine an ordering.
- Ordering Methods 2a) and 2b) - use only the second-order coefficients to determine an ordering (two versions).
- Ordering Method 3 - use a weighted combination of the first-order and second-order coefficients to determine the ordering.

The reverse of each of these orderings was also generated and tested, in order to determine if the overall position of grouped variables in the ordering would affect the ROBDD size, or if only the adjacency of certain variables would affect it.

These were each compared to a randomly generated ordering, and to the best ordering that Item could produce.

The functions used for this work are limited to those with fewer than 20 inputs. In addition, only single-output functions were considered. Table 5 lists the functions used for the experiments, from the standard Berkeley benchmark set. Multiple output files were broken into one file for each output. The output in question is indicated by a number appended to the end of the file name, *e.g.*, the first output of add6 becomes add61.pla.

file name	inputs	products
9sym.pla	9	87
add6	12	1092
adr4	8	256
b9	16	123
cm152a.blif	11	8
co14.pla	14	47
ex1.pla	5	16
ex2.pla	5	7
ex3.pla	5	4
ex4.pla	5	8
life	9	512
log8mod	8	47
max46	9	46
parity.pla	8	128
ryy6	16	112
sym10	10	837

Table 5: List of the files used in the experiments.

The truncated average for each result is also compared, defined as the average percent improvement (as given for the other comparisons) with the best and worst values discarded.

4.1 Ordering Method 1 – First-order Coefficients

Ordering Method 1 is based only on the first-order autocorrelation coefficients of the function, and the variables are sorted from smallest to largest according to their first-order autocorrelation coefficients.

Tables 6 and 7 give the percentage improvements that the first ordering method achieves over Item's ordering method and a randomly generated ordering. It is clear that the reverse ordering does far worse than the original ordering, indicating that the variables with the largest first-order coefficients should appear at the *end* of the ordering.

ordering method	% improvement over Item	% improvement over Random
first-order	11.15	23.54
truncated avg.	10.20	22.82
reverse	-58.11	6.75
truncated avg.	3.23	15.83

Table 6: Summary of results for Method 1 orderings, as percentage improvement in the number of ROBDD nodes.

ordering method	% improvement over Item	% improvement over Random
first-order	9.09	24.32
truncated avg.	9.22	26.12
reverse	-92.68	6.00
truncated avg.	-4.82	11.42

Table 7: Summary of results for Method 1 orderings, as percentage improvement in the number of CLBs.

4.2 Ordering Methods 2a and 2b – Second-order Coefficients

Ordering methods 2a and 2b are slightly more complex than method 1. Ordering method 2a is described below.

1. Choose the first variable according to the variables' first-order autocorrelation coefficients – the variable with the largest first-order coefficient is chosen.
2. Choose another variable to be placed adjacent to this variable by finding a pair containing this variable that has a large second-order autocorrelation coefficient.
3. Repeat the previous step until all the variables have been added to the ordering. No variables can be repeated.

Ordering method 2b is a variation on the above algorithm. The steps are listed below.

1. Find the pair of variables with the largest second-order autocorrelation coefficient.
2. Check to see if either of these variables have already been added to the ordering; if one of them has been then:
   - if the variable is at the beginning of the list, add its pair to the beginning (put it adjacent to the one already in the list);
   - if the variable is at the end of the list, add its pair to the end;
   - if the variable is in the middle of the list, add its pair to the beginning of the ordering.
3. If neither of the pair is in the ordering, add them to the head of the pairing.
4. If both of the pair are already in the ordering, discard them.
5. Go back to step 1 and continue until either all variables are in the ordering, or the end of the list of coefficients is reached. Each time a pair is used, it is removed from the list.

Tables 8 and 9 give the average improvement achieved over Item's ordering and the random ordering by these two ordering methods. Method 2b) performs better overall, with a truncated average of over 11% improvement over Item's best ordering. Note that since method 2a) orders the variables in such a way that those with *largest* coefficients appear at the start of the ordering instead or at the end, the reversed method gives a better result than the original 2a) ordering.

ordering method	% improvement over Item	% improvement over Random
Method 2a)	-58.09	5.51
truncated avg.	3.24	14.51
2a) reversed	9.38	20.78
truncated avg.	8.93	20.61
Method 2b)	11.89	24.24
truncated avg.	10.98	23.56
2b) reversed	-60.88	5.20
truncated avg.	0.29	14.81

Table 8: Summary of results for Method 2a) and 2b) orderings, as percentage improvement in the number of ROBDD nodes.

ordering method	% improvement over Item	% improvement over Random
Method 2a)	-89.13	8.15
truncated avg.	-2.61	13.53
2a) reversed	8.72	23.52
truncated avg.	8.04	24.24
Method 2b)	5.16	20.53
truncated avg.	4.81	20.07
2b) reversed	-84.82	9.99
truncated avg.	-0.47	15.17

Table 9: Summary of results for Method 2a) and 2b) orderings, as percentage improvement in the number of CLBs.

4.3 Ordering Method 3 – First and Second-order Coefficients

Ordering method 3 makes use of both the first and second-order autocorrelation coefficients in determining an ordering. A weighted sum consisting of $r_i + r_j + r_{ij}$. r_i is the first-order coefficient for variable x_i, r_j is the first-order coefficient for variable x_j, and r_{ij} is the second-order coefficient for the pair of variables $x_i x_j$. This metric is then used to determine an ordering by following the algorithm described for method 2b above.

Tables 10 and 11 give the percentage improvement of the third ordering method over the Item's ordering method and a randomly generated ordering.

ordering method	% improvement over Item	% improvement over Random
ri + rj + rij	10.71	23.00
truncated avg.	9.79	22.25
reverse	-58.63	6.25
truncated avg.	2.72	15.33

Table 10: Summary of results for Method 3 orderings, as percentage improvement in the number of ROBDD nodes.

ordering method	% improvement over Item	% improvement over Random
ri + rj + rij	8.85	25.01
truncated avg.	9.00	25.81
reverse	-90.50	5.02
truncated avg.	-6.41	9.96

Table 11: Summary of results for Method 3 orderings, as percentage improvement in the number of CLBs.

4.4 Summary of Results

The results displayed above demonstrate that each of the ordering methods 1, 2b) and 3 performs relatively well. By this we mean that the orderings always achieved both smaller ROBDDs and smaller number of CLBs than did the randomly generated ordering, and there was also some improvement in these measures over the ordering that the synthesis software Item could produce. It should be noted here that our ordering methods performed on average 90% faster than did the ordering heuristics employed by Item.

5 Conclusions

This work has proved the feasibility of such an approach to determining variable ordering for an ROBDD representation. The results achieved by making use of the autocorrelation coefficients to order the variables in building the ROBDD are consistently better than the random ordering, even when such a simple set of heuristics were used.

There are many areas in which additional work can be done, in particular on improving the heuristics. For instance, whether to put the variables with the most similarity at the top of the ordering or at the end has not been addressed in detail, and yet this obviously makes a significant difference in the size of the ROBDD.

Extending this work to multiple-output functions is clearly an area where work can continue. For instance, some functions simply do not have reasonable (*i.e.* non-exponential-sized) ROBDDs, in the case of multiple output functions, for all outputs. In this case it would be logical to be able to identify, using the autocorrelation coefficients possibly, these troublesome functions before attempting to find an optimal ordering. If it is determined that there may be a "good" ordering, then the heuristics in this work may be extended to take into account multiple outputs.

Finally, it is clear that for certain functions, different ordering methods may work better than others. It would be very useful to be able to determine ahead of time which ordering method will produce the best results.

References

[1] B. Bollig and I. Wegener. Improving the Variable Ordering of OBDDs is NP-complete, 1996.

[2] R. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, vol. C-35(no. 8):677–691, August 1986.

[3] K. Karplus. Using if-the-else DAGs for Multi-Level Logic Minimization. Technical Report UCSC-CRL-88-29, University of California Santa Cruz, November 1988. ftp'd from ftp.cse.ucsc.edu, 20 pages.

[4] K. Karplus. ITEM: an If-Then-Else Minimizer for Logic Synthesis. In *Proceedings of Euroasic 1989*, 1989. ftp'd from ftp.cse.ucsc.edu.

[5] K. Karplus. Using if-then-else DAGs to do Technology Mapping for Field-Programmable Gate Arrays. Technical report, Universtiy of California Santa Cruz, 1990. ftp'ed from ftp.cse.ucsc.edu.

[6] M. Karpovsky. *Finite Orthogonal Series in the Design of Digital Devices.* John Wiley and Son, 1976.

[7] Randal Tomczuk. Autocorrelation and Decomposition Methods in Combinational Logic Design, 1996.