New multiplexer-based switching circuits synthesis methods

Peter Pištek*
Institute of Computer Systems and Networks
Faculty of Informatics and Information Technologies
Slovak University of Technology in Bratislava
Ilkovičova 2, 842 16 Bratislava, Slovakia
pistek@fiit.stuba.sk

Abstract

The work deals with the problem of methods of synthesis of switching circuits with multiplexers and it is mainly focused on optimization and reduction of these circuits. It analyses and describes the state of art in optimization and reduction methods of multiplexer trees and also contains description of the justification for the use of multiplexers in the switching circuits.

We design a novel method of decision diagrams based on binary decision diagrams (BDD) which uses residual variable. Proposed model can be easily transformed to the multiplexer tree. A new type of binary decision diagram allows anyone to work without using the lowest and most numerous level of nodes and through the preservation of the necessary properties of binary decision diagram allows its widespread use for a number of already existing, optimization methods as it is shown in performed experiments. New model was experimentally validated on the benchmark circuits with reference to the saving in the number of nodes in a model of what constitutes a major contribution to a new type of binary decision diagrams in the switching circuits design with multiplexers. Other benefits of our work is a method for reducing multiplexer-based circuit with dynamic propagation path control.

Categories and Subject Descriptors

B.6.3 [**Logic Design**]: Design Aids—*Automatic synthesis, Optimization*; B.6.1 [**Logic Design**]: Design Styles—*Combinational logic*

Keywords

BDD, multiplexer tree, residual variable, genetic algorithm, static algorithm, area reduction, multi–parameter optimization, synthesis acceleration

1. Introduction

In the design of the switching circuits there has always been observed a high priority on reducing the number of logic gates. Currently an equally important role is represented by effective design of switching circuits focusing on the reducing electrical consumption, size and delay of a circuit. Multiplexers are also an essential element of switching circuits which can represent any arbitrary function and they are used in the design of VLSI circuits (high degree of integration). It is also possible to combine multiplexers in a multiplexer tree and create the large-scale structure.

An extensive research in the area of multiplexer tree optimization has been done so far, resulting in a number of different optimization techniques. Since high fan-in multiplexers provide poor scalability and are inefficient to manufacture, it is most common to implement them as a multiplexer tree of several lower fan-in multiplexers, usually 2-to-1 multiplexers. A big contribution to multiplexer tree optimization was the idea of using BDD [3] as their structural description [8]. Most importantly, this means a BDD can be directly mapped onto a digital switching circuit. BDD and multiplexers are suitable for the representation of control and symmetric functions. They are not optimal for representation of arithmetic logic circuits or error correcting circuits[7].

Since the creation of an optimal BDD is an NP-complete problem[4], to further optimize a BDD, an optimal input variable ordering has to be found. Therefore efforts are being made to achieve closer to the optimal result by using several types of methods from a static variable ordering to evolutionary algorithms that also make it possible to optimize several parameters simultaneously.

BDD and multiplexers can also be used as an input for advanced tools, which combine and transform them to circuits containing logic gates AND, OR, MUX, XOR or XNOR [22]. This solution was later extended by [2] by MAJ members (carrying out the function of the majority) in the system BDS-MAJ. This extension covered functions for the control, as well as carrying out functional units (such as arithmetic logic unit, multipliers) and it can achieve 30% improvement compared to BDS.

When setting up appropriate synthesis' parameters the power consumption (besides circuit area) belongs to the main technical problems dealt with in the industry of integrated circuits at the present. Energy leakage is growing exponentially as the production process progresses to-

*Recommended by thesis supervisor: Prof. Milan Kolesár.

wards the smaller size in nano-scale[1]. Well-executed circuit synthesis of design technology can meet the market requirements and it can avoid the constraints caused by computational complexity. In the case of multiplexers it can also mean attempts to shorten the longest or an average path length (LPL, APL) of multiplexer tree, which may be important to create a circuit in a given technology (e.g. PTL). Additional endpoints often include energy consumption and average or the longest path length; these parameters can be optimized by a modification of binary decision diagrams. Nowadays, in ever smaller dimensions used in the manufacture of transistors dynamic power consumption decreases in total consumption, the contrary static consumption increases. Direct impact on the static power consumption is the number of gates in the circuit. It is appropriate to synthesize circuit with the lowest number of gates [17].

This work deals with the multiplexer and the multiplexer trees as such, their effective methods of synthesis, optimization and reduction and we use most widespread mathematical model in this area: decision diagrams. We propose a new extension of BDD which is more compact compared to the prior approach and we discuss its benefits.

2. Preliminaries

In this section, the properties of a multiplexer tree and BDD are defined. The input of algorithm based on Kolesar is a Boolean function (further B-function) f of n input variables $x_0, x_1, ..., x_{n-1}$ and a binary vector y. This is denoted as $f(x_0, x_1, ..., x_{n-1}) = y$, where the order of input variables corresponds to the decreasing weights assigned to the variables from left to right, starting from the weight of 2^{n-1} for variable x_0 down to the weight of 2^0 for x_1. Such ordering can be then denoted as $x_0, x_1, ..., x_{n-1}$. Any B-function f specified by its binary vector y and a fixed variable ordering can be easily expressed as a BDD [21].

The modification of given function to a function of residual variables can be done by representing binary vector of the function as a canonical matrix (1). Such matrix [18] contains 2^n columns of two rows. Each column represents a pair of values x_1 (bottom row) and $\bar{x_1}$(top row) of the function.

$$B = ((x_1, ..., x_n)) = \tag{1}$$

$$\begin{vmatrix} f(0) & f(1) & ... & f(2^n-2) & f(2^n-1) \\ f(2^n) & (f(2^n+1) & ... & f(2^{n+1}-2) & f(2^{n+1}-1) \end{vmatrix}$$

Afterwards, according to a pattern shown in *Table 1*, a value from set $\{0, 1, x, \bar{x}\}$ is assigned for every column $(f(i), f(i+2^n)$ where i resides in interval $< 0, 2^n >$.

The result is a modified binary vector of given function, called vector of residual functions. The length of this vector is equal to half of the length of the original binary vector. In the context of multiplexer trees, this modified function can be represented by a multiplexer tree one level shorter than original multiplexer tree. Values of residual variable in positive and complemented form are then connected to data inputs of multiplexer tree, along with values of logical I and 0.

BDD is a rooted, directed acyclic graph consisting of one

Table 1: Value assignment for vector of residual functions

Y		Value
f(i)	f(i)	
0	0	0
0	1	x_i
1	0	$\bar{x_i}$
1	1	1

or two terminal nodes of out-degree zero labelled by I and 0 and a set of variable nodes u of out–degree two with two outgoing edges labelled low and high. BDD has only one node with no parent edge, called the root node.

BDD optimization methods can be divided into two main categories [9]

1. BDD ordering – results in Ordered Binary Decision Diagram (OBDD), which respects a given order of input variables. Variable ordering can have great impact on effectiveness of BDD reduction.
2. BDD reduction – when applied on OBDD, results in Reduced OBDD (ROBDD) which has lower number of nodes than not reduced BDD. ROBDD respects these two rules:
   a) Uniqueness (Type I)– no two distinct nodes u and v represent the same variable and have the same left and right successor, i.e.: *var(u) = var(v), left(u) = left(v), right(u) = right(v)* which implies $u = v$.
   b) Non-redundancy (Type S) – no variable node u has identical left and right successor, i.e.: *left(u) = right(u)*

Since creation of an optimal BDD is an NP–complete problem [4], to further optimize a BDD, an optimal input variable ordering has to be found.

Especially for circuits consisting of larger number input variables, it is unusable to search the entire space of possibilities ($n!$ different orders of input variables). The methods are categorized into following groups[6]

- Basic methods.
- Heuristic methods.
- Alternative approaches (mostly evolutionary algorithms).

Basic methods

Goal of exchange variables at k and $k+1$ level for the $\pi - BDD\ G$ of function f is to transform G to $\pi' - BDD\ G'$ of function f. The only difference between variable ordering π and π' is only at levels k and $k+1$ and can be expressed as: $\pi(k) = \pi'(k+1)$ and $\pi(k+1) = \pi'(k)$. This operation is called the exchange of two adjacent variables (level $\pi(k)$ for the level of $\pi(k+1)$ - see *Fig. 1*). Exchange of adjacent variables does not affect the higher and lower levels in BDD.

Heuristic methods

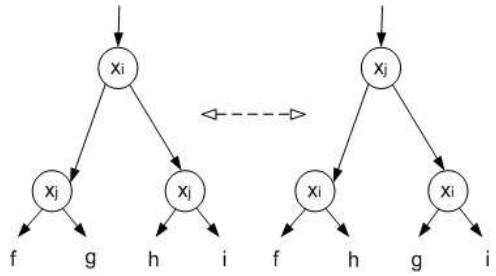

Figure 1: Exchange of adjacent levels

The ordering of variables will be determined according to the information available about the issue before the construction of BDD itself. The best known algorithms in this category is Force algorithm [20]. The idea behind the FORCE algorithm is simple: the algorithm computes the forces acting upon each variable and displaces the variables in the direction of the forces acting upon them. In FORCE, a CNF formula is viewed as a hypergraph, where the formula's variables correspond to vertices and clauses correspond to hyperedges. The FORCE algorithm determines two values during execution and iteratively uses them to order the variables.

Method [10] is based on the proven assumption that the number of nodes in a particular level of BDD depends only on the arrangement of variables at lower levels. The algorithm of this method sequentially places all the variables to the first level and determines to which of them it received the least number of nodes. This variable (or several variables) is saved for the level, then the remaining variables are tested at upper level. The process is repeated until the final BDD is obtained. Algorithm remains exponential, but with better parameters than testing all variables orderings. The method was used as the basis for a group of methods which improve it further.

Alternative methods

Most of alternative methods is based on evolutionary algorithms (EA), which belong to the state-of-art in the field. Approaches are characterized by learning rules for finding optimal variables ordering using heuristics for the dynamic variable ordering or direct the use of evolutionary algorithms for reducing the BDD. More information can be found in [6]. Every EA has three main components [19]:

- Population of individuals representing candidate solutions to the underlying optimization problem.
- Fitness function or objective function that determines the environment within which the solutions live and which determines the strength of an individual as an optimum solution to the given problem.
- Evolutionary operators leading to the evolution of population of individuals that are better suited to their environment than the individuals that they were created from.

Genetic algorithm (GA)[5]. Main parameters affecting the performance of GAs are population size, number of generations (iterations), crossover rate and mutation rate. Larger population size (number of chromosomes) and large number of generations increase the likelihood of obtaining a global best solution, but increase the computation time as well.

Particle Swarm Optimization (PSO) [15] is a population-based stochastic technique inspired by social behaviour of bird flocking or fish schooling. In PSO, the potential solutions, also called 'particles', fly through the problem space by following the current optimum particles. Particles learn from their past experiences, learn from others experience and finally converge near the solution, which may be the best or a suitably good solution after satisfying a definite termination criterion. The particles sense their proximity to a good solution using a parameter known as the fitness function [19].

A key feature of memetic algorithms (modified memtic algorithm - MMA) [19] is the use of various techniques of local search. Whereas the gene that passes on the offspring cannot be changed (except for mutations) in the genetic algorithms, memes transmit information among themselves so as to best suit the evaluation function (for example through local searches) in memetic algorithms which happens through knowledge of solution's local space.

In [16] low power architecture of MUX tree at the register transfer level was introduced. Within this architecture, each MUX has its own individual selection signal that is dynamically generated by a dedicated controller. The controller is designed to keep those selection signals remain unchanged as many as possible since only those selection signals lying on the actual output propagation path need to be properly set. As a result, redundant transitions of MUX cells in the MUX tree can be significantly eliminated. Adaptation of this method in BDD would be interesting in terms of dynamic power consumption. If we use this method in BDD we should design a reduction method of control part of circuit.

3. Residual variable in BDD

If we are going to implement a function with *n+1* variables using BDD as the complexity of the function with n variables, we need to have one of the input variables available in both direct and complemented form. The input has to identify variables ordering so the variable in both direct and complemented form had the highest weight in binary vector.

Definition 1 – If we identify the presences of variables in function of *n+1* variables (e.g., by ordering in the truth table), then we can assess these variables by weights. Excluding the variable that has the highest weight from the list of variables and it is replaced by the logic value in direct and complemented form, such variable is called a residual variable. An arbitrary variable may be a residual variable, if it is available in the both direct and complemented form, otherwise the transformation of RViBDD (Residual Variable in BDD) to a specific circuit will need to add a NOT logic gate.

Definition 2 – If we identify the residual variable in function of *n+1* variables, then the circuit of this function is transformed into a circuit of function with *n+1* variables, the residual variable is connected to the data input.

Definition 3 – Created BDD for the function f with *n+1* variables is called RViBDD (Residual Variable in BDD),

if one variable is the residual variable and it is also a terminal node of RViBDD. RViBDD can be created from the truth table by using the methodology used by multiplexer trees with a residual variable (see Preliminaries).

RViBDD creation can also be obtained by Shannon decomposition, which uses this notation [6]:$f = \bar{x_i} f_0^i + x_i f_i^1$,wherein f_i^0 and f_i^1 express these defined functions:

Decomposition of the function f by $x_i = c(c \in \{0,1\})$ and the variables ordering $x_1, ..., x_{i-1}, x_i, x_{i+1}, ..., x_n$ is given by function: $f|_{x_i=c}(x_1, ..., x_{i-1}, x_i, x_{i+1}, ..., x_n) = f(x_1, , x_{i-1}, c, x_{i+}), x_n))$. Instead of $f|_{x_i=0}$ respectively $f|_{x_i=1}$ denoted f_i^0 respectively $f_i^1, i \in \{1, , n\}$,where $n \in N$ is the number of variables.

To create BDD with n variables the existing procedure repeats Shannon decomposition until it reaches the level defined by the equation *2*.

$$f|_{x_n=c} = (x_1, ..., x_n) := f((vectorn - 1), c), \quad (2)$$

where $c \in \{0,1\}$ and *(vector n-1)* contains corresponding substitution of 0s and 1s according to the specified order of variables $x_1..., x_{n-1}$ in the upper levels of BDD and its particular propagation path. We obtain BDD which determines the value of the function for each combination of values for given variables $x_1..., x_n$.

In the case of decomposition of function is stopped by one iteration earlier, we gain equation *3*

$$f|_{x_n-1=c} = (x_1, ..., x_n) := f((vector\ n - 2), c, x_n), \quad (3)$$

where $c \in \{0,1\}$ and *(vector n-2)* contains corresponding substitution of 0s and 1s according to the specified order of variables $x_1..., x_{n-2}$ in the upper levels of BDD and its particular propagation path. It is possible by using this method to achieve up to four final conditions which depend on the value c a x_n:

- 0, represented by inputs "00",
- 1, represented by inputs "11",
- $\bar{x_n}$, represented by inputs "10",
- x_n, represented by inputs "01".

The total number of nodes (without terminal nodes) in G BDD for the function with n variables is possible to obtain by modification of existing equation [11]:

$$|G| = 1 + \sum_{i=2}^{L} [2^{n - \sum_{j=i}^{L} k_j}], \quad (4)$$

where $L \in N$ is number of levels in a multiplexer tree or in a BDD and it is defined by equation:

$$L = \lceil \frac{n}{k} \rceil, \quad (5)$$

where:

- n is the number of control variables (also the level in BDD),
- k (Equation *5*) or k_j (Equation *4*) is number of control variables of each multiplexer. In BDD model k_j, where $j = i, i+1, ...L$.

Table 2: Rules for usage of residual variable

Rule	v_1	v_2	u_i
0	0	0	0
1	0	1	x_n
2	1	0	$\bar{x_n}$
3	1	1	1

After substituting values 1 for k the equation *5* is adjusted to $L{=}n$. It is sufficient substitute value n for L for other equations which use the number of levels of BDD. Equation (*4*) can be modified for BDD (which represents function with n variables) to:

$$|G| = 1 + \sum_{i=2}^{n} [2^{n - \sum_{j=i}^{n} 1}] = 1 + \sum_{i=2}^{n} [2^{i-1}], \quad (6)$$

where $|G|$ denotes number of internal nodes in BDD.

In the case of function with n variables is implemented by RViBDD; the residual variable is used and it expresses the value of the terminal nodes in RViBDD. The number of nodes in RViBDD is determined by the equation (modification of *4*):

$$|G'| = 1 + \sum_{i=2}^{n} [2^{(n-1)-(n-i-2)}] = 1 + \sum_{i=2}^{n-1} [2^{i-1}], \quad (7)$$

It is clear from the comparison of equations *6* and *7* that RViBDD achieves decrement of the number of nodes in the most numerous (first) level compared to BDD. The achieved reduction is 2^{n-1} nodes where n represents number of input variables.

Let u_i for $i = 0, 1, ..., 2^{n-1}$, where n denotes number of input variables of given function, denotes i–th terminal node in RViBDD, then the value of the terminal node u_i (*Fig. 3*) can take four possible values, which depend on the function values $v_1 = f((vector\ n-1), x_n)\ and\ v_2 = f((vector\ n-1), \bar{x_n})$, wherein:

- *(vector n-1)* represents the corresponding substitution 0 and 1 according to a specified order of variables $x_1, x_2, ..., x_{n-1}$ in upper levels of BDD and its specific propagation path,
- $v_1, v_2 \in \{0,1\}$

This makes possible to convert BDD to RViBDD without necessity of inclusion of residual variable at the first place. (See *Table 2*).

Transformation of BDD (which expresses the function of *n+1* variables) to RViBDD will be carried according to *Fig. 2* based on the rules in *Table 2* if x_{n+1} is chosen to be a residual variable. Change occurs only at levels $x_n\ and\ x_{n+1}$. Upper levels of BDD remain unchanged and are therefore identical to RViBDD.

Since it is a new type of DD, which is derived from the BDD, it is necessary, to maintain the basic rules for the reduction and to work with BDD. Maintenance of the rules is important in terms of reusing existing reduction and optimization algorithms developed for BDD. Consequence of conservation of the features is to increase the usability of RViBDD. It is necessary to compare the two

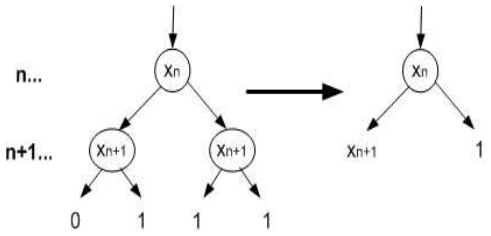

Figure 2: BDD transformation to RViBDD

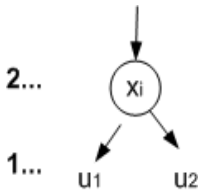

Figure 3: RViBDD in bottom levels

diagrams and to determine their compatibility rate if the same rules of reduction have to be applied to both BDD and RViBDD.

In the case of the Shannon decomposition there is I–type and S–type of reduction. Proof of usability is trivial, we have to itemize the first level of RViBDD to original BDD (*Fig. 2*) and it can be seen that nodes in the first level can be used as nodes in the BDD. In the case of the same value (0 or 1) at node in RViBDD, the situation is a very similar to the reduction of S–type, which leads to the same result in both BDD and RViBDD. The upper levels of the RViBDD and BDD remain identical. The same rules of decomposition remain preserved which is necessary for the reduction of type S and I.

Important feature in the reduction of BDD is the exchange of adjacent levels. This is the basic approach for getting more efficient BDD in terms nodes. Although the approach itself is not very efficient, it serves as the basis for most state-of-art complex methods. If adjacent variables are exchanged (levels i and j) then variables ordering of function is modified from $f(x_1, x_2, ..., x_i, x_j, ..., x_n)$ to $f(x_1, x_2, ..., x_j, x_i, ..., x_n)$.

Current approach remains unchanged if it is the exchange between the other than the first and terminal level of BDD (level of terminal nodes; the values from the set $\{0, 1, \bar{x}_i, x_i\}$) because of higher levels of BDD and RViBDD are identical and exchange variables between adjacent level are independent from nodes in the other levels.

Based on the rules for the exchange of nodes between adjacent levels (*Fig. 4*), we can modify variables ordering in upper levels for RViBDD the same way as for BDD. Creation of a new RViBDD would be necessary when a new residual variable is selected. It is appropriate to add rules for the exchange of the residual variable in the generated diagram for better usability of RViBDD. This new approach can be used to exchange adjacent variables at higher levels in modification of the method in [18] to optimize multiplexer trees.

BDD in the first level can look like in *Fig. 3*.

Terminal nodes u_1, u_2 can take values $\{0, 1, \bar{x}_i, x_i\}$. If level 1 would be adjusted retrospectively then the rules for exchange of adjacent levels (variables) is recreated (*Fig 4*), and BDD from RViBDD can be obtained (example in *Fig. 5*).

Terminal nodes v_1, v_2, v_3, v_4 take the values $\{0, 1\}$, and on the basis of their values we obtained values in the terminal level where new residual variable in RViBDD is used (*Fig. 3*, *Table 2*).

The methods of exchange variables between adjacent levels allow RViBDD to be applied to other, more complex methods (heuristic methods, evolutionary algorithms, etc.), which are mostly derived right from this method, and it is the major contribution of exchange of residual variable. Replacing the residual variable allows us the modification of existing RViBDD, which is more efficient than creating a new RViBDD with another residual variable. Since the exchange of residual variable needs to perform more operations than for common exchange of variables at adjacent levels it can be used a static variable orderings (e.g. methods analysed by [20]) to find the variable that appears to be most suitable for the position of the residual variable. Otherwise, this is probably the only part of the synthesis, which may occur to increase required computational time.

3.1 Propagation path control

If we adapt [16] to be used in BDD, it is necessary for the correct behaviour of the control circuit to implement adjustment for the following reason:

If we remove a node from RViBDD based on of the rules for the reduction it is not enough to simply delete the corresponding part of the control circuit. We need to connect together the logic gates that determine the control of the propagation paths.

Modification occurs by adding an OR gate before the multiplexer, which represents two or more multiplexers from the original non–optimized circuit. To this OR gate the outputs of logical AND gate of preserved multiplexer is connected, but also the outputs of other logic gates AND that were used in the multiplexer removed from the circuit. Connecting of AND gates into a single OR gate is necessary because of joining multiple propagation paths of non-optimized UMS—n into a single propagation path in the reduced tree multiplexer. Thus, in reducing such circuit, it is necessary to add a new gate to the logic circuit – OR. Whereas, an OR is a fundamental logic gates, there are no significant changes in the area of the circuit from an already used logic gates.

4. Experimental results

Experiments were made on benchmark circuits LGSynth'93 [14]. When removing the redundant terminal nodes RViBDD can be improved compared to BDD only by two nodes at maximum in fact, whereas terminal nodes may acquire the value of residual variable in direct or complemented form. Thus formed BDD and RViBDD can be described as a ROBDD and reduced and ordered RViBDD (RORViBDD). The improvements were anticipated due to the preservation of attributes of ROBDD in RORViBDD and thus the same procedures for the reduction can be used in both models. Substantial advantage of RViBDD reduction is to be based on a decision diagram which

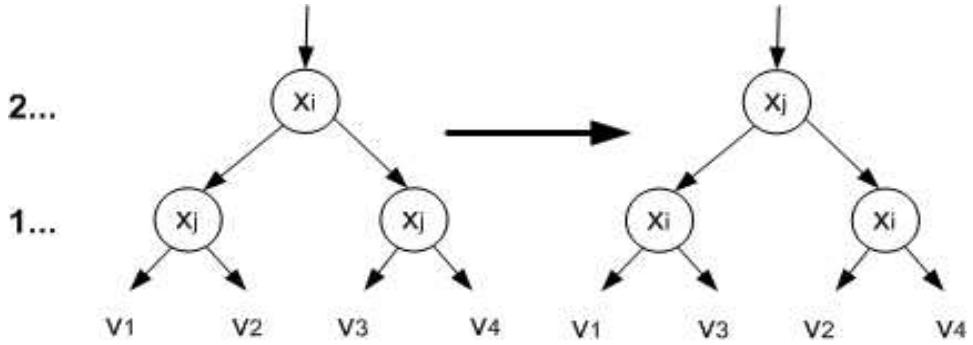

Figure 4: Exchange of residual variable in RViBDD

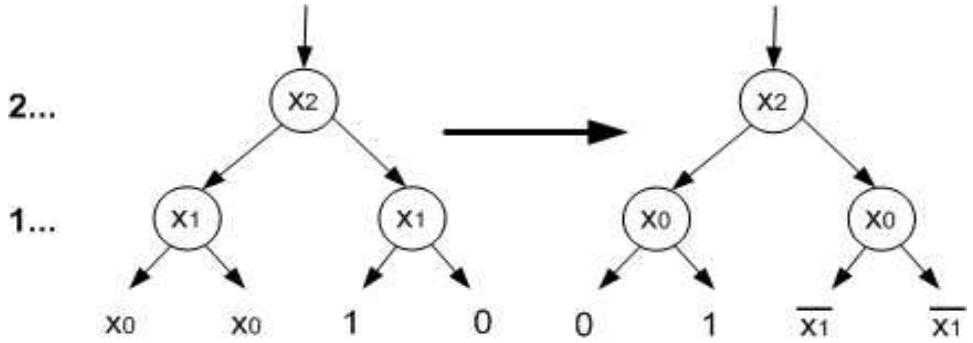

Figure 5: Exchange of residual variable in RViBDD - example

contents smaller number of nodes, which is advantageous both from a memory, as well as over time needed to apply reduction rules to the model.

Advantage of using RViBDD for finding optimal RORViBDD compared with BDD, designed for finding ROBDD, in terms of time is shown in the last column in *Table 3* (column Imp. mean Number of nodes improvement and column Acc. means Synthesis' acceleration), which shows the percentage improvement of time needed for the synthesis of circuits that were obtained by testing *n!* of variable orderings. By creating data inputs and a residual variable right at the beginning, it is necessary to make a smaller number of controls throughout the diagram. The acceleration caused by removing most numerous level is given by the number of occurrences of residual variable in direct or complemented form (0 and 1 in the data input can also be obtained by using existing reduction rule S–type).It can be assumed that in the special cases where the removal of most numerous level applies none residual variable is used (in the direct or complemented form), so the method will have no contribution. An exception could be only if the rules for removing most numerous level repeat again for the next variable in variable ordering. The average improvement in time needed for the synthesis RViBDD is 40.7% (20.41% was the smallest observed improvement). It is not a problem, if the acceleration is below 50%, although the most numerous level was removed which contains approximately half of the nodes. When using the rules for the reduction of BDD is complicated in terms of time to apply I–type reduction, especially at upper levels of decision–diagram when you need to compare longer parts of the binary vector of function(s). I–type reduction is fast enough at removed first level of BDD compared to upper levels so the contribution to the overall reduction in the time required may be less than it might seem at first glance. Shrinking factor of the time needed to optimize RViBDD can also be a greater probability for faster finding of the difference between data vectors (in the vector, there are four distinct elements instead of two) belonging to the nodes to be tested on the reduction rule I–type. The comparison is of course already accelerated by creating hash for these vectors and we compare only those vectors that have the same hash. The further acceleration occurs by shortening the comparison vectors to half the size in each stage (compared to the BDD for the same order of input variables). With this faster comparison of vectors we can ultimately exceed the 50% threshold of acceleration, for example 81.38% acceleration synthesis in case of con1 cir-

Table 3: BDD and RViBDD comparison

Benchmark	BDD	RViBDD	Imp. (%)	Acc. (%)
5XP1	78	63	19,23	34,94
9sym	33	31	6,06	41,28
cm151a*	44	42	4,55	-
cm152a*	21	19	9,52	-
con1	17	14	17,65	81,38
inc	100	87	13,00	43,15
majority	7	6	14,29	29,23
misex1	64	50	21,88	20,41
parity*	31	29	6,45	-
rd53	29	24	17,24	39,41
rd84	71	64	9,86	43,13
sao2	163	155	4,91	45,13
sqrt8	42	37	11,90	40,45
squart5	47	35	25,53	31,64
xor5	9	7	22,22	30,64

cuit.

We shown benefits of using residual variable in RViBDD where it reduced from 4.55% to 25.53% more nodes compared to ROBDD. The following RViBDD were compiled during the transformation without nodes that may only be excluded due to use of residual variable in direct or complemented form (Rule 2 and 3 in *Table 2*). For all test circuits for all individual experiments the reduced RViBDD came out more effectively than the reduced BDD. There was not achieved a single case with identical or a larger number of nodes in RViBDD which is attesting to its advantage compared to BDD.

In cooperation with [13] the existing genetic algorithm [5] was modified without using the hybrid method (called "Branch and Bound") and with using of elitism [12]. Testing parameters were set as follows:

- Parameters of fitness function: 1 – 0 – 0 (Area – Power consumption – APL).
- Population: 500 (variable count <12), 200 (variable count from 12 to 21), 100 (cordic).
- Crossover rate: 80%.
- Mutation rate: 20%.
- Elitist: 1%.
- Iterations: 100.

After basic comparison of evolutionary algorithm with a residual variable, the advanced EAs were compared with: PSO [15] and MMA [19]. Test parameters for the genetic algorithm BDD and RViBDD were preserved as in the previous case. Results of compared algorithms have been incorporated from the relevant articles in which the authors set the optimal testing parameters:

- PSO: population: 100, w=0,9, $c_1 = 1,75$, $c_2 = 2,25$, 100 iterations.
- MMA: population: 20, 80% crossover rate, 4% mutation rate, 20 iterations.

Weight of parameters of evaluation function has been set so that the algorithms focus on minimizing area. In addition to evolutionary methods there has been added an sifting method [21] which is often used as a reference method for testing. As it is clear from the results of *Table 4* the genetic algorithm (column BDD) is the least effective among the compared evolution algorithms. Under normal circumstances, MMA would be considered as the most effective which uses local optimization techniques. If the residual variable is used and thus the RViBDD model, we managed to reach that the genetic algorithm using RViBDD has become the most effective in 8 out 14 cases. On the other hand, it should be noted that the genetic algorithm using BDD was worst in 9 out 14 cases compared to the other evolutionary algorithms, and in 5 cases it has reached the same result as another genetic algorithm, and not even once reached the best result. This demonstrates the importance of RViBDD for practical use in other already existing methods which can achieve improvement, if they would be based on this model. Of course, the more

Table 4: EAs comparison (numeber of nodes)

Benchmark	BDD	RViBDD	PSO	MMA	Sifting
con1	15	12	16	15	18
rd53	29	24	-	23	23
cm151a	32	30	32	-	34
cm150a	32	31	32	-	33
mux	32	31	32	-	33
sqrt8	35	32	33	33	42
squar5	47	34	37	37	38
misex1	62	49	36	36	41
rd84	71	64	-	59	59
inc	96	81	79	61	68
5xp1	76	59	68	68	82
sao2	103	96	91	85	92
cordic	209	144	105	-	93
9sym	33	31	-	33	33

Table 5: BDD and RViBDD comparison (%)

	UMS-n	MUX	D-FF	Logic
5XP1	81,88	84,28	81,88	67,96
9sym	90,98	91,34	90,98	21,57
cm152a	98,63	98,73	98,63	98,63
con1	73,91	77,27	73,91	67,39
majority	60,00	64,29	60,00	46,67
misex1	79,83	82,30	79,83	77,68
rd53	60,00	64,29	60,00	22,22
rd84	90,94	91,67	90,94	25,59
sao2	95,35	95,54	95,35	78,96
sqrt8	80,72	82,72	80,72	18,07
xor5	73,33	78,57	73,33	26,67

efficient the method is and the smaller the difference to the results achieved by *n!* of variable orderings is obtained, the smaller benefit RViBDD will have in terms of the absolute number nodes (maximum improvement of 2 nodes to a single output circuit).

The achieved results (in *Table 5)* show a significant improvement in percentages, particularly for multiplexer tree itself which provides benchmark function. The column UMs-n means improvement in the number of multiplexers in benchmark function; column MUX means improvement in the number of multiplexers in control part of a circuit; column D–FF means improvement in the number of D-FFs in control part of a circuit and column Logic means improvement in the number of OR and AND gates. Also very good results (over 60%) were subsequently achieved in the number of multiplexers and D–FF in the control circuit. Percentages of improvements caused by removing multiplexers (whether in the control part of circuit, or in the circuit itself) and by removing D–FF preserved the ratio between them. The reason is the structure of the control circuit, which assigns on D–FF to each multiplexer in UMS–n and the number of multiplexers in the control circuit is one less than the number of multiplexers in UMS-n, because the multiplexer at the top level of UMS–n does not need to have own multiplexer in the structure of control circuit. Improvement of control part of the circuit is thus proportional to the reduction rate obtained in the main part of the circuit. Greater variance in terms of improving (18.07% – 98.63%) occurs at the elementary logic gates OR and AND. At smaller improvements in elementary logic gates we can say that the

reduction of multiplexer tree took place mostly at the lower levels of the circuit. This is caused by the occurrence of greater number of AND gates in lower levels, which have to be connected to the multiplexer, which replaces other multiplexers eliminated based on the rules for the reduction.

If the reduction of UMS–n is able to remove multiplexers from the upper levels, it is also possible to remove part of the control circuit's structure, which controls the propagation path in lower levels. From this perspective, the size of the reduced circuit is more preferably an order of input variables, which allows the removal of multiplexers making up a subtree of the UMS–n instead of removing an equal number of multiplexers in the lower levels. The selected method improves the reduction of the number of all types of gates and thus the area of the resulting circuit is decreased. We managed to prove presumption and modify the method of dynamic power consumption in multiplexer circuits (referred in [16]) in such way it can be useful for decision diagrams and at the same time it provides the possibility of reducing of the control part of the circuit, which can be significantly smaller in reduced circuit. Through the use of decision diagrams it is possible to create a benchmark function by other logic gates (e.g. XOR, AND, OR, NOT, etc.). In that case, the importance of dynamic control of propagation paths is probably greatly reduced because the phenomenon where logical gates in one level are using a specific variable on its input may disappear.

5. Conclusions

The main purpose of this work was the proposal of a new method of optimization and reduction of circuits based on multiplexer structures. Reduction and optimization, based on work with the variable orderings, belongs to NP–complete problems, and therefore research in this area is still in progress and effectiveness of the proposed methods is tested under various parameters. The most common parameter includes the number of multiplexers in the circuit. In addition, there are more important parameters such as the static and dynamic power consumption circuit, switching intensity of a circuit or the average path length, which affects the delay obtained by a circuit. The specific objectives of the work were:

- Proposal of a new type of BDD (RViBDD), which uses the residual variable and reduction model of circuit's structure which consists of multiplexers.
- Proposal of reduction of type BDD and ensure the compatibility with the previous model: BDD.
- Design of method of reducing number of gates in the multiplexer tree while control of propagation path is used.

Our main contribution is the definition of a new model (RViBDD), which is based on BDD and it can be the basis of new methods of optimization and reduction of selected switching circuits. RViBDD uses residual variable and removes the most numerous level of nodes. Thanks to this it contains 2^{n-1} nodes less (n is the number of input variables) compared with the BDD obtained by Shannon decomposition. Definition of a new type of decision diagram without compatibility with BDD could be relevant only for a limited number of cases. A new decision tree - RViBDD has the same properties (to ensure mutual compatibility), which are used in BDD, in particular the rules for the reduction and replacement of adjacent levels. This allows the application of more complex methods, which are derived mainly from the rule for the exchange of adjacent levels and rules for the reduction. Existing methods can use RViBDD as its initial state, and they immediately reach approximately half the number of nodes in comparison with BDD, which reduces the memory and computational demands.

The average improvement of the RViBDD was in [10] (20,88%), [20] (15,56%). Implemented genetic algorithm using the RViBDD was compared with the basic genetic algorithm (uses a BDD), but also to more advanced algorithms (PSO [15] and MMA [19]), which generally have better results than genetic algorithms. The comparison found out that through the use of RViBDD it was possible to improve the performance of less efficient genetic algorithm. It was the worst performing algorithm in 9 out of 14 cases (and not a single best result) with BDD and the best with RViBDD in 8 out of 14 cases, all of the remaining cases there was an improvement in comparison with BDD. It can be expected that the use of RViBDD as input model should improve the results of the most recent multiparameter algorithms.

It is advisable to select a methods from state-of-art and fully test and exchange of adjacent variables between levels in RViBDD and compare the rate of synthesis of BDD while searching for optimal circuit using RViBDD as a RORViBDD As shown in definition of RViBDD the exchange of residual variable is only part of the calculation which is more complex in comparison with BDD because it is necessary to control 16 different cases (actually there may be only 12 of them during working with ROViBDD; we skip the option where a node has the same children). The complexity of calculating in use of the S-type reduction rule is maintained, on the other hand the complexity of the calculation for I-type reduction rule decreases because of the length of the compared vectors (even when using hash functions) is reduced to half the size for nodes at each level. I-type reduction rule for internal nodes is significantly more often in use than exchange of residual variable.

It would be appropriate to propose extension of RViBDD by complemented edges for better work with the decision diagram and to transform RViBDD into multiplexers with direct and complemented output. The complemented edge modification would provided the possibility to compare with the approaches BDS [22] and BDS-MAJ [2], which allow the transfer of BDD into hardware implementation, which uses also other gates than multiplexers. We assume that the exchange of group of nodes in RViBDD by XOR and MAJ gates should also remain compatible for this model. It would be more complex to modify the existing rules for OR and AND gates, which is mainly due to the fact that in RViBDD, it is not guaranteed that the internal node is on the propagation path that begins an value 0 (or 1). After designing of these rules it will be necessary to make tests to verify the suitability of RViBDD (or RORViBDD) for these tools, not only in terms of the number of nodes in the circuit, but also in terms of the synthesis time.

Usage of residual variable would be appropriate to test for other types of decision diagrams, especially ZBDD and FDD which are using other reduction rules applied to the nodes, or nodes obtained by using a different type of decomposition. Especially for ZBDD, it will be a simple adjustment, because it would be necessary to prove only compatibility with another rule for reduction (type D).

Acknowledgements. This work has been partially supported by the Grants No. 1/0649/09 and 1/1008/12 of the Slovak VEGA Grant Agency.

References

[1] International technology roadmap for samiconductors: Design. `http://www.itrs.net/Links/2009ITRS/`. [Online; accessed September, 2009].

[2] L. Amarú and G. P E Gaillardon, G De-Micheli. Bds-maj: A bdd-based logic synthesis tool exploiting majority logic decomposition. In *50th ACM / EDAC / IEEE Design Automation Conference (DAC)*, pages 1–16. IEEE, June 2013.

[3] B. Becker, R. Drechsler, and R. Werchner. On the relation between bdds and fdds. *LCNS - Journal Information and Computation*, 123(2):185–197, 1995.

[4] B. Bolling and I. Wegener. Improving the variable ordering of obdds is np–complete. *IEEE Transactions on Computers*, 45(9):993–1002, 1996.

[5] S. Chaudhury and A. Dutta. Algorithmic optimization of bdds and performance evaluation for multi-level logic circuits with area and power trade–offs. *Circuits and Systems*, 2(3):217–224, 2011.

[6] R. Drechsler and B. Becker. *Binary Decision Diagrams*. Kluwer Academic Publisher, Netherlands, 1998.

[7] R. Drechsler, M. Theobald, and B. Becker. Fast ofdd-based minimization of fixed polarity reed-muller expressions. *IEEE Transactions on Computers*, 45(11):1294–1299, 1996.

[8] R. Ebendt and R. Drechsler. Exact minimisation of path-related objective functions for binary decision diagrams. *IEEE Proceedings - Computers and Digital Techniques*, 153(4):231–242, 2006.

[9] R. Ebendt, G. Fey, and R. Drechsler. *Advanced BDD Optimization*. Springer, Netherlands, 2005.

[10] S. J. Friedman and K. J. Supowit. Finding the optimal variable ordering for binary decision diagrams. *IEEE Transactions on Computers*, 39(5):710–713, 1990.

[11] N. Frištacký, M. Kolesár, and J. Koleniča. *Logické systémy - Kombinačné obvody(In Slovak)*. SVŠT, Slovakia, 1986.

[12] A. Kumar, S. Choudhary, and P. V. Varde. Optimization of binary decision diagram using genetic algorithm. In *2nd International Conference on Reliability, Safety and Hazard (ICRESH)*, pages 168–175. IEEE, 2010.

[13] M. Maruniak. Metódy optimalizácie multiplexorových stromov(In Slovak). Master's thesis, 2014.

[14] K. McElvain. LGSynth93 Benchmark Set: Version 4.0. `http://www.cbl.ncsu.edu:16080/benchmarks/LGSynth93/`. [Online; accessed August, 2013].

[15] A. Mitra and S. Chattopadhyay. Variable ordering for shared binary decision diagrams targeting node count and path length optimisation using particle swarm technique. *IET Computers & Digital Techniques*, 6(6):353–361, 2012.

[16] L. Nan-Shing, H. Juinn-Dar, and H. Han-Jung. Low power multiplexer tree design using dynamic propagation path control. In *IEEE Asia Pacific Conference on Circuits and Systems - APCCAS 2008*, pages 838–841. IEEE, 2008.

[17] R. S. Panda and et al. *Power-efficient System Design*. Springer Science+Business Media, New York, 2010.

[18] P. Pištek, M. Kolesár, and K. Jelemenská. Optimization of multiplexer trees using modified truth table. In *International Conference on Applied Electronics*, pages 265–268. IEEE, 2010.

[19] S. Rehan and M. Bansal. Performance comparison among different evolutionary algorithms in terms of node count reduction in bdds. *International Journal of VLSI and Embedded Systems – IJVES*, 4(1):491–496, 2013.

[20] M. Rice and S. Kulhari. *A Survey of Static Variable Ordering Heuristics for Efficient BDD/MDD Construction*. University of California, Technical Report, 2008.

[21] R. Rudell. Dynamic variable ordering for ordered binary decision diagrams. In *Int'l Conf. on CAD*, pages 42–47. IEEE, 1993.

[22] C. Yang and M. Ciesielski. Bds: A bdd-based logic optimization system. *Computer-Aided Design of Integrated Circuits and Systems*, 21(7):866–876, 2002.

Selected Papers by the Author

P. Pištek, M. Kolesár, K. Jelemenská. Multiplexer Trees using Modified Truth Table. In Jiří Pinker, ed. *2010 International Conference of Applied Electronics Proceedings*, pages 265–268, Pilsen Czech Republic: University of West Bohemia, 2010, IEEE.

P. Pištek, M. Maruniak. Binary Decision Diagram Optimization Method Based on Multiplexer Reduction Methods. In *IEEE International Conference on System Science and Engineering –ICSSE 2013 Proceedings*, pages 395–399, Budapest, Hungary, 2013, IEEE.

P. Pištek, M. Maruniak. A New Multiplexer–based Multi–objective Digital Circuit Synthesis Method. In *EUROMICRO DSD/SEAA 2014*, Verona, Italy, IEEE. Accepted for publishing.

P. Pištek , R. Marcinčín, T. Palaj, J. Štrba. Logical Circuits Design Education Based on Virtual Verification Panel. In *Lecture Notes in Electrical Engineering*, volume 151 of *Emerging Trends in Computing, Informatics, Systems Sciences, and Engineering*, pages 903–914, New York: Springer Science–Business Media, 2013.

P. Pištek ,K. Šutý. Analysis of multiplexer trees by path analysis. In *AWERProcedia. Information Technology and Computer Science*, volume 1 of *2nd World Conference on Information Technology I – WCIT 2011 Proceedings*, pages 366–370, Antalya, Turkey, 2013, Academic World Education & Research Center.

P. Pištek, M. Kolesár, K. Jelemenská. Reduction of multiplexer trees using modified lookup table. In *AWERProcedia. Information Technology and Computer Science*, volume 1 of *2nd World Conference on Information Technology I – WCIT 2011 Proceedings*, pages 533–538, Antalya, Turkey, 2013, Academic World Education & Research Center.

P. Pištek. Automatic Design of Combinational Logical Circuits with Multiplexers and SSI Circuits. In *Proceedings of Informatics and Information Technologies Student Research Conference 2010*, pages 365–372, 2010, Nakladateľstvo STU.

P. Pištek. Optimalizácia multiplexorových stromov. In *Proceedings of Počítačové architektury a diagnostika – PAD 2010*, pages 69–74, Češkovice, Czech Republic, 2010, Brno: VUT in Brno – FIT.

P. Pištek. Redukcie a optimalizácie multiplexorových stromov. In *Proceedings of Počítačové architektúry a diagnostika – PAD 2011*, pages 127–132, Stará Lesná, Slovak republic, 2011, Nakladateľstvo STU.