Science Publications

Open Access
Author Manuscript

American Journal of Applied Sciences 5 (2): 158-164, 2008
ISSN 1546-9239

Minimization of Multiple-Valued Decision Diagrams Based on Matrix Computation

[1] Omid Kavehie, [1] Keivan Navi, [1] Ebrahim Afjei and [1] Hamid Khorsand
[1] Department of Electrical and Computer Engineering, Shahid Beheshti University, Tehran, IRAN

Abstract: This paper proposes a new algorithm to simplify the multiple valued logic (MVL) decision diagrams. This algorithm is based on a new level coupling rule. By changing the designer's perspective towards the design, this rule can make further simplification possible. In most of state of the art designs, the decision diagram plays a serious role in the implementation of the logical functions. The proposed algorithm uses the new level coupling rule and combines it with the existing ones, presenting a new method in simplifying and implementing the basic decision diagram.

Keywords: Multiple-Valued Logic (MVL), Matrix Computation, Directed Acyclic Graph, Free Decision Diagram (FDD), Ordered Decision Diagram (ODD) Design Automation.

INTERODUCTION

Using the decision diagram is a very efficient and common method to illustrate the switching functions. Additionally using this method compared to the other methods has many advantages such as overall circuit delay reduction, higher layout density, less power dissipation and added logical flexibility [10]. This fact is confirmed by widespread application of this trend in different logic families. For example we can cite the pull-down tree in DCVSL (Differential Cascade Voltage Switch Logic), SSDL (Sample-Set Differential Logic), ECDL (Enable/disable CMOS Differential Logic), DCSL (Differential current switch logic), etc. Another example is the realization of multiple valued logic circuits, especially in current mode (BDDs, and MDDs). The arrangement of the layout variables or their ordering will affect the dimensions of the decision diagram. Many different methods have been presented to find the best ordering however the level coupling method is a very efficient method and it can be used in BDDs and MDDs [2,15,21].

In this method, the coupling method, achieving the best ordering in the input variables (controls) is simply accomplished by analyzing those specific diagrams obtained by the coupling operations. This fact will help to reduce time and operational complexity of the proposed algorithm. This rule dose not influence the simplification of the reduction tree directly, but by changing the viewpoint of the designer it would simplify further possibilities, in comparison to the existing rules.

As mentioned above, the decision diagram defines a graph based structure for representing radix2 and higher radix. Numerous researches in simplification and implementation of MVL decision diagram have been conducted [1,3,9]. In[3] some of the well known implementations of MDD have been considered as well several methods to reduce dimensions of the decision diagram (ROMDD). The realization of algorithms is achieved by using the C programming language. In[5, 6 and 9], some of methods for realization of MVL decision diagram are introduced, which are completely encapsulated in the CAD package. In [1] a method has been proposed in which the dimension reduction of diagrams is achieved by using multiplexers and combining the output terminals. In [7] one can observe the different kinds of simplification and implementation of radix 3 decision diagram. There is also a discussion about the MVL decision diagram, which illustrates that the number of nodes for function realization in a defined radix has an exponential relation to the number of variables, and the upper bound of this number is analyzed.

In [4], a method called CDD (copy DD) is introduced and initially its influence on size reduction of the multi-terminal BDD is investigated and then its simplifying domain in MDDs is expanded by introducing CMDDs. Other method such as "sifting" [15] is also proposed.

In this paper the required preliminaries to begin the discussion is described and then the state of the art rules are explained. Third section discusses the coupling rule and its related algorithm for decision diagram and realization of their current mode in radix2 (BDD) and higher radixes (MDD). In the following section results of some examples in different radixes are represented. The penultimate section investigates the time complexity of the mentioned algorithm and the final section an overall conclusion with regard to the obtained results is provided.

Corresponding Author: Keivan Navi, Department of Electrical and Computer Engineering, Shahid Beheshti University, P.O. Box 19839-63113, Velenjak, Evin, Tehran, Iran, Tel: +98-2129902286, Fax: +98-2122431804

PRELIMINARIES

We assume that there is an r-valued function $f(x_0, x_1, \ldots, x_{n-1})$ in which x_i can have any arbitrary value from the set *{0, 1, …, r-1}*. In other words function f is some form of Multiple-dimensional (multiple-valued) to two-dimensional (two-valued) domain assignment, $f: R^n \to \{0,1\}$ where $R = \{0,1,2,\ldots,r-1\}$. These functions are presented based on multiple value decision diagrams and using DAG (Directed Acyclic Graph). In this graph every node has r input terminals which are labeled 0, 1… r-1. Every node has one output. The nodes including "0" and "1" values are called "leaf". An MDD is called ordered if going from root to leaves, traveling through every path and only one specific ordering of input variables is seen. The ordered MDD is abbreviated as OMDD. A reduced MDD is one in which there are no nodes with inputs which are all connected to a single node or leaf. Meanwhile the graph is not permitted to have an isomorphic sub-graph (the second simplifying rule) [7].

In this type of representation the minterms or a product term (p) has two general forms. The first one is:

$$p = x_0^{v_0} \cdot x_1^{v_1} \cdot x_2^{v_2} \cdot \cdots \cdot x_{n-1}^{v_{n-1}} \qquad (1)$$

x_i s are the function variables and Vi s are variable values. We have $v_i = \{0,1,\ldots,n-1\}$ where $i = \{0,1,\ldots,n-1\}$.

In other words, if the product term p is a component of the product terms of f, then if $x_0 = v_0, x_1 = v_1, x_2 = v_2, \ldots, x_{n-1} = v_{n-1}$ f will equal "1". Generally, a set of these product terms constitute the function $f = \sum(p_1, p_2, \ldots, p_k)$.

Another representation of p is:

$$p = (x_0, x_1, x_2, \cdots, x_{n-1}) = (v_0 v_1 v_2 \cdots v_{n-1})_r \qquad (2)$$

This representation is also very descriptive. As an example an MDD graph is illustrated in Fig. 1 which represents function f, which is equal to $f = \sum(101_3, 110_3, 111_3, 112_3)$ where n= 3 and r= 3. The function F can be demonstrated as $f = A^1B^0C^1 + A^1B^1$.

Before continuing the discussion on MDD we track BDDs and their related implementation functions. The structured BDD design method was expanded by Pulfrey and Chu [21]. In this method the function table is used to design a tree capable of realization of f and $\overline{f}$. This method allows the designer to construct some part of the transistor making f and $\overline{f}$ in common. This fact greatly helps the simplifying procedure, chip area reduction and circuit speedup [2]. In order to implement this tree we also use source-coupled transistors (as illustrated in Fig. 2). From now on, we consider Fig. 2(b) as the symbolic representation of the source-coupled transistors. One of the transistors is controlled by the input and the other by the inverse of the output. Every transistor has a separate drain which is connected to a and b. "a" and "b" can take the logic values "0" or "1". Either can be the output of higher layers. This structure has a unique output called "u". The way "u" is calculated which is based on the controls is illustrated in Fig. 2.

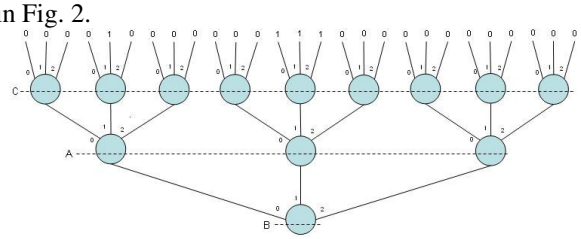
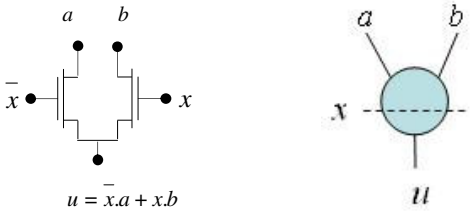


Fig. 1: Illustration of a 3-varible graph.



(a) Basic circuit (b) Symbol

Fig. 2: Basic BDD cell structure.

In reality Fig. 2 structure is a multiplexer. In Fig. 2(b) the side with negative sign (-) is controlled by $\overline{x}$ and the opposite side (+) is controlled by x. The first DD simplification rule for r=2 is described with equation (3) and illustrated in Fig. 3. According to this rule, a node with equal inputs has no influence on circuit functionality and can be omitted [11], [12].

$$u = a \cdot x^0 + a \cdot x^1 = a \cdot (x^0 + x^1) = a \qquad (3)$$

In this situation the transistors do not perform any logical operations apart from directing the input to the output node, so one can directly connect one of the inputs to the related output. This is also called short circuit.

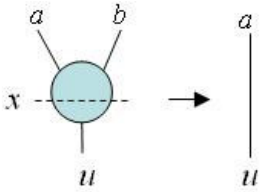


Fig. 3: Omitting one node with equal inputs (a= b).

If $r \geq 2$ then, in the related node entry, there will be r inputs. As in here we have r= 2 and the number of inputs is 2. Now if there is a case in which two nodes have the exact same inputs and controls, then we can consider these two nodes as equivalents and omit one (Fig. 4). This is called the second simplifying rule [11, 12].

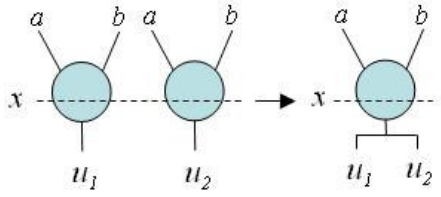


Fig. 4: Elimination of a node.

If $r \succ 2$ then, in the related node entry, there will be r inputs. We want to accentuate that it is straightforward to extend these rules to MDDs. Using the two noted simplifying rules; one can simplify the decision diagram and reduce it size. In the next section we will describe the level coupling rule.

NEW D.D. SIMPLIFICATION ALGORITHM

A. The coupling rule: In the following algorithm, we use a new rule. We have called it the "coupling rule" or the "third simplifying rule". A point to note in applying the third rule into simplifying is that in spite of the first and second rules, this rule dose not directly contribute to the simplifying process, however by changing the designer's viewpoint it provides more simplifying capability than the first and second rule (which directly influence the simplifying procedure) [2]. Fig. 5 illustrates a branch of decision diagram for r radix with "n" variables.

$\alpha_k, \beta_k, \gamma_k, \ldots, \omega_k$ are the symbols reserved for branch inputs (where $k = \{0,1,\ldots,r-1\}$ and x_i, x_j are control variables in these two levels, where $i, j = \{0,1,\ldots,n-1\}$, $i \neq j$, and $\alpha, \beta, \gamma, \ldots, \omega$ are also the weights or the labels of node input terminals. Fig. 6 represents the coupling of these two levels of decision diagram.

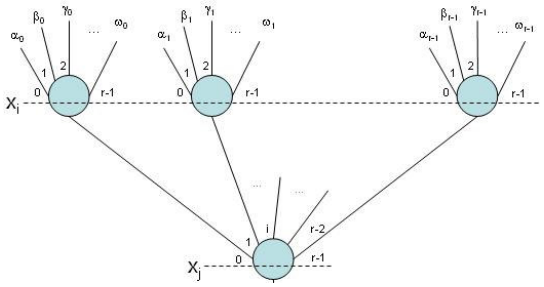


Fig. 5: Two levels of decision diagram.

The proof of the fact that the outputs of the two branches are equal is as follows (in this equation the node output is presented by *u*) (equ. 4);

$$\begin{aligned} u = &\left(\alpha_0 \cdot x_i^0 + \beta_0 \cdot x_i^1 + \gamma_0 \cdot x_i^2 + \cdots + \omega_0 \cdot x_i^{r-1}\right) \cdot x_j^0 + \\ &\left(\alpha_1 \cdot x_i^0 + \beta_1 \cdot x_i^1 + \gamma_1 \cdot x_i^2 + \cdots + \omega_1 \cdot x_i^{r-1}\right) \cdot x_j^1 + \\ &\left(\alpha_2 \cdot x_i^0 + \beta_2 \cdot x_i^1 + \gamma_2 \cdot x_i^2 + \cdots + \omega_2 \cdot x_i^{r-1}\right) \cdot x_j^2 + \\ &\quad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ &\left(\alpha_{r-1} \cdot x_i^0 + \beta_{r-1} \cdot x_i^1 + \gamma_{r-1} \cdot x_i^2 + \cdots + \omega_{r-1} \cdot x_i^{r-1}\right) \cdot x_j^{r-1} \end{aligned} \quad (4)$$

The matrix representation of those relations is as shown in equ. 5:

$$u = \begin{pmatrix} x_j^0 & x_j^1 & x_j^2 & \ldots & x_j^{r-2} \end{pmatrix}_{1\times r} \times \begin{pmatrix} \alpha_0 & \beta_0 & \gamma_0 & \ldots & \omega_0 \\ \alpha_1 & \beta_1 & \gamma_1 & \ldots & \omega_1 \\ \alpha_2 & \beta_2 & \gamma_2 & \ldots & \omega_2 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \alpha_{r-1} & \beta_{r-1} & \gamma_{r-1} & \cdots & \omega_{r-1} \end{pmatrix}_{r\times r} \times \begin{pmatrix} x_i^0 \\ x_i^1 \\ x_i^2 \\ \vdots \\ x_i^{r-1} \end{pmatrix}_{r\times 1} \quad (5)$$

Using multiplication and simplifying operations in the above matrix one can find out that there is another representation for *u*. This is exact representation of what we have called coupling.

$$\begin{aligned} u = &\left(\alpha_0 \cdot x_j^0 + \alpha_1 \cdot x_j^1 + \alpha_2 \cdot x_j^2 + \cdots + \alpha_{r-1} \cdot x_j^{r-1}\right) \cdot x_i^0 + \\ &\left(\beta_0 \cdot x_j^0 + \beta_1 \cdot x_j^1 + \beta_2 \cdot x_j^2 + \cdots + \beta_{r-1} \cdot x_j^{r-1}\right) \cdot x_i^1 + \\ &\left(\gamma_0 \cdot x_j^0 + \gamma_1 \cdot x_j^1 + \gamma_2 \cdot x_j^2 + \cdots + \gamma_{r-1} \cdot x_j^{r-1}\right) \cdot x_i^2 + \\ &\quad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ &\left(\omega_0 \cdot x_j^0 + \omega_1 \cdot x_j^1 + \omega_2 \cdot x_j^2 + \cdots + \omega_{r-1} \cdot x_j^{r-1}\right) \cdot x_i^{r-1} \end{aligned} \quad (6)$$

Or:

$$u=\begin{pmatrix} x_i^0 & x_i^1 & x_i^2 & \cdots & x_i^{r-2}\end{pmatrix}_{1\times r} \times \begin{pmatrix} \alpha_0 & \alpha_1 & \alpha_2 & \cdots & \alpha_{r-1} \\ \beta_0 & \beta_1 & \beta_2 & \cdots & \beta_{r-1} \\ \gamma_0 & \gamma_1 & \gamma_2 & \cdots & \gamma_{r-1} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \omega_0 & \omega_1 & \omega_2 & \cdots & \omega_{r-1}\end{pmatrix}_{r\times r} \times \begin{pmatrix} x_j^0 \\ x_j^1 \\ x_j^2 \\ \vdots \\ x_j^{r-1}\end{pmatrix}_{r\times 1} \quad (7)$$

Where we label the following matrix, the input matrix:

$$I_{r\times r}=\begin{pmatrix} \alpha_0 & \beta_0 & \gamma_0 & \cdots & \omega_0 \\ \alpha_1 & \beta_1 & \gamma_1 & \cdots & \omega_1 \\ \alpha_2 & \beta_2 & \gamma_2 & \cdots & \omega_2 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \alpha_{r-1} & \beta_{r-1} & \gamma_{r-1} & \cdots & \omega_{r-1}\end{pmatrix}_{r\times r} \quad (8)$$

And the following vector, the control vector of the j-th level:

$$Cj_{1\times r}=\begin{pmatrix} x_j^0 & x_j^1 & x_j^2 & \cdots & x_j^{r-2}\end{pmatrix}_{1\times r} \quad (9)$$

Additionally we labeled the following matrix the control matrix of the i-th level input:

$$Ci_{r\times 1}=\begin{pmatrix} x_i^0 \\ x_i^1 \\ x_i^2 \\ \vdots \\ x_i^{r-1}\end{pmatrix}_{r\times 1} \quad (10)$$

Transposing all elements of the matrix and then switching the two matrices $(Ci_{r\times 1})^T_{1\times r}$ and $(Cj_{1\times r})^T_{r\times 1}$ according to matrix multiplication rules, will result in the same "*u*" as before. The matrix:

$$\begin{pmatrix} \alpha_0 & \alpha_1 & \alpha_2 & \cdots & \alpha_{r-1} \\ \beta_0 & \beta_1 & \beta_2 & \cdots & \beta_{r-1} \\ \gamma_0 & \gamma_1 & \gamma_2 & \cdots & \gamma_{r-1} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \omega_0 & \omega_1 & \omega_2 & \cdots & \omega_{r-1}\end{pmatrix}_{r\times r}$$

Is the transpose of matrix $I_{r\times r}$ or $(I_{r\times r})^T$.

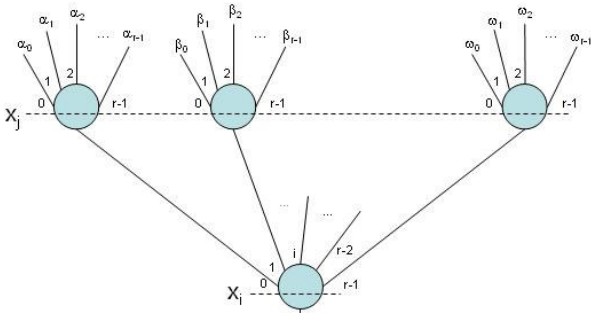


Fig. 6: Coupling of two levels of decision diagram.

There are some remarkable points about level coupling (level exchange) operation that make it important:

1. Exchanging and interchanging of the location of input variables (control variables)
2. The possibility of applying the rule locally and only to those special branches of the graph which can be simplified
3. The possibility of simplifying prediction, based on the inputs ordering before applying the coupling operation.

This property helps us to make more and more suitable implementation of the decision diagram. As an example the level coupling rule for a binary decision diagram can be represented as in equ. 11:

$$\begin{aligned} u_{BDD} &= \left(\alpha_0\cdot x_1^0+\beta_0\cdot x_1^1\right)\cdot x_0^0+\left(\alpha_1\cdot x_1^0+\beta_1\cdot x_1^1\right)\cdot x_0^1= \\ &\alpha_0\cdot x_1^0\cdot x_0^0+\beta_0\cdot x_1^1\cdot x_0^0+\alpha_1\cdot x_1^0\cdot x_0^1+\beta_1\cdot x_1^1\cdot x_0^1= \\ &\left(\alpha_0\cdot x_0^0+\alpha_1\cdot x_0^1\right)\cdot x_1^0+\left(\beta_0\cdot x_0^0+\beta_1\cdot x_0^1\right)\cdot x_1^1=u \end{aligned} \quad (11)$$

The coupling operation can be defined for level L (L is indicated in next pseudo codes) according to the following codes:

```
array branch_inputs[0···r−1,0···r−1];
array temp[0···r−1,0···r−1];
  for col = 0 to r−1 {
    for raw = 0 to r−1 {
      temp[raw,col] = branch_inputs[col,raw];
    };
  };
branch_inputs = temp;
```

In [17], it is proven that for achieving the simplest graph with minimum size, it is sufficient to consider and analyze the specific trees with independent control variable locations.

$$\begin{pmatrix} x_0 & x_1 & x_2 & \cdots & x_{n-1} \\ x_1 & x_2 & x_3 & \cdots & x_0 \\ x_2 & x_3 & x_4 & \cdots & x_1 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ x_{n-1} & x_0 & x_1 & \cdots & x_{n-2} \end{pmatrix}_{n\times n}$$

In other words this matrix demonstrates that for achieving the minimum size of the decision diagram it is sufficient to analyze only n graphs. It means that in the above example for a branch of a two level binary graph for n= 3, it is sufficient to analyze only 3 graphs. Although the proof of it was questioned in 2005 [16], as stated in that paper, only in some rare cases this procedure cannot achieve the minimum graph size. Nevertheless, the proposed rule can be applied in both cases. In other words, by using the new coupling rule as new optimize tools; any possible graph can be covered. Using the following algorithm an ordered DD could lead to an ordered or non-ordered (free) DD. In the following algorithm, this rule is applied to graphs that have already as far as possible been simplified with first and second rules.

As another example, if we analyze all the different kind of permutations in a binary graph (BDD) with n= 3, we will obtain six different graphs (with different variable ordering) which represent the same function as shown below:

$$u_{BDD}^{n=3} = \left((\alpha_0 \cdot x_2^0 + \beta_0 \cdot x_2^1) \cdot x_1^0 + (\alpha_1 \cdot x_2^0 + \beta_1 \cdot x_2^1) \cdot x_1^1\right) \cdot x_0^0 + \left((\gamma_0 \cdot x_2^0 + \omega_0 \cdot x_2^1) \cdot x_1^0 + (\gamma_1 \cdot x_2^0 + \omega_1 \cdot x_2^1) \cdot x_1^1\right) \cdot x_0^1 =$$

$$u_{BDD}^{n=3} = \left((\alpha_0 \cdot x_1^0 + \alpha_1 \cdot x_1^1) \cdot x_2^0 + (\beta_0 \cdot x_1^0 + \beta_1 \cdot x_1^1) \cdot x_2^1\right) \cdot x_0^0 + \left((\gamma_0 \cdot x_1^0 + \gamma_1 \cdot x_1^1) \cdot x_2^0 + (\omega_0 \cdot x_1^0 + \omega_1 \cdot x_1^1) \cdot x_2^1\right) \cdot x_0^1 =$$

$$u_{BDD}^{n=3} = \left((\alpha_0 \cdot x_1^0 + \alpha_1 \cdot x_1^1) \cdot x_0^0 + (\gamma_0 \cdot x_1^0 + \gamma_1 \cdot x_1^1) \cdot x_0^1\right) \cdot x_2^0 + \left((\beta_0 \cdot x_1^0 + \beta_1 \cdot x_1^1) \cdot x_0^0 + (\omega_0 \cdot x_1^0 + \omega_1 \cdot x_1^1) \cdot x_0^1\right) \cdot x_2^1 =$$

$$u_{BDD}^{n=3} = \left((\alpha_0 \cdot x_0^0 + \gamma_0 \cdot x_0^1) \cdot x_1^0 + (\alpha_1 \cdot x_0^0 + \gamma_1 \cdot x_0^1) \cdot x_1^1\right) \cdot x_2^0 + \left((\beta_0 \cdot x_0^0 + \omega_0 \cdot x_0^1) \cdot x_1^0 + (\beta_1 \cdot x_0^0 + \omega_1 \cdot x_0^1) \cdot x_1^1\right) \cdot x_2^1 =$$

$$u_{BDD}^{n=3} = \left((\alpha_0 \cdot x_0^0 + \gamma_0 \cdot x_0^1) \cdot x_2^0 + (\beta_0 \cdot x_0^0 + \omega_0 \cdot x_0^1) \cdot x_2^1\right) \cdot x_1^0 + \left((\alpha_1 \cdot x_0^0 + \gamma_1 \cdot x_0^1) \cdot x_2^0 + (\beta_1 \cdot x_0^0 + \omega_1 \cdot x_0^1) \cdot x_2^1\right) \cdot x_1^1 =$$

$$u_{BDD}^{n=3} = \left((\alpha_0 \cdot x_2^0 + \beta_0 \cdot x_2^1) \cdot x_0^0 + (\gamma_0 \cdot x_2^0 + \omega_0 \cdot x_2^1) \cdot x_0^1\right) \cdot x_1^0 + \left((\alpha_1 \cdot x_2^0 + \beta_1 \cdot x_2^1) \cdot x_0^0 + (\gamma_1 \cdot x_2^0 + \omega_1 \cdot x_2^1) \cdot x_0^1\right) \cdot x_1^1$$

Algorithm: As stated in the previous part of this section, we present an algorithm which adds many different capabilities to the decision diagram design tool.

The algorithm illustrated in Fig. 7 accepts an OMDD graph as input. The next step in this approach is detecting branches that can be coupled and lead to further simplification.

We begin this operation by making some matrix called "coupling matrix", which must be fabricated for every two level branch of the graph.

```
array CM_Z[0···r−1, 0···r−1];
for Z = 0 to r^(m−2) − 1 {
  for col = 0 to r − 1 {
    for raw = 0 to r − 1 {
      CM_Z[raw, col] = value(x_j^col · x_i^raw);
    };
  };
};
```

This process is done step by step and if we are to number graph levels from root to bottom(0 to m) then level m would be the terminal nodes, or a node containing '0' or "1" values. We perform the procedure of making CM matrices from the (m-2) [th] level nodes to the top and consequently in every step we consider only one level. In each step, we consider m-1 and m-2 level nodes.

For the whole graph:

```
array CM_Z^L[0···r−1, 0···r−1];
int bound_level;
if m mod 2 = 0 { bound_level = 0;
  else bound_level = 1; };
for L = m − 2 to bound_level step = 2 {
  for Z = 0 to r^(m−2) − 1 {
    for col = 0 to r − 1 {
      for raw = 0 to r − 1 {
        CM_Z^L[raw, col] = value(x_j^col · x_i^raw);
      };
    };
  };
/* STOP for reduction */
};
```

In this step, simplifying must be applied. We must notify that every time the term 'L level coupling" is used, it means that the L-1 and L+1 levels are considered and manipulated. The overall function of the algorithm is as follows:

1. If in CM_Z^L matrices (with constant L, Z) all elements of the C-th column are equal, then we can eliminate the c-th node from L+1 level. In the case where all the elements of CM_Z^L matrixes

(considering constant L, Z) are equal, then the Z-th branch can be omitted in L and L+1 levels

2. If in the same CM_Z^L matrixes (considering constant L) there are some columns with equal elements then we can keep one node and eliminate the others. Choosing the node we keep is an important consideration
3. The information about the number of nodes in the above two steps will be stored and then we transpose the coupling matrix and finally steps 1 and 2 will be applied to them again. The transposition is done from the lowest levels up to the root and in this procedure, the upper-level matrices are corrected as well.

Transposition of coupling matrices means applying the coupling operation on levels m-2, m-4, m-6, and so on. If a larger number of simplifications are obtained in this level, simplification information is rewritten; otherwise, the old information is kept. In both cases, matrices are transposed once again after this step and all the proceeding steps are repeated.

RESULTS

In [20] a method named 123dd has been presented in which the improvement of the number of transistors in DCVS tree for:

F = (1011 0000 1011 0011 1011 1001 1011 0001)

is more than 15% (from 26tT to 22T). If we implement F with the new proposed algorithm we will only need 16T. The improvement is about 34.5% in comparison to the 26T implementation and 27.27% in comparison to the 12dd method.

As an example of MDD, Fig. 1 demonstrates the function $G = \sum(101_3,110_3,111_3,112_3)$ and as another example, we can mention the sample presented in [7]. Its implementation with the new proposed method illustrates a reduction of 20% in the number of transistor used.

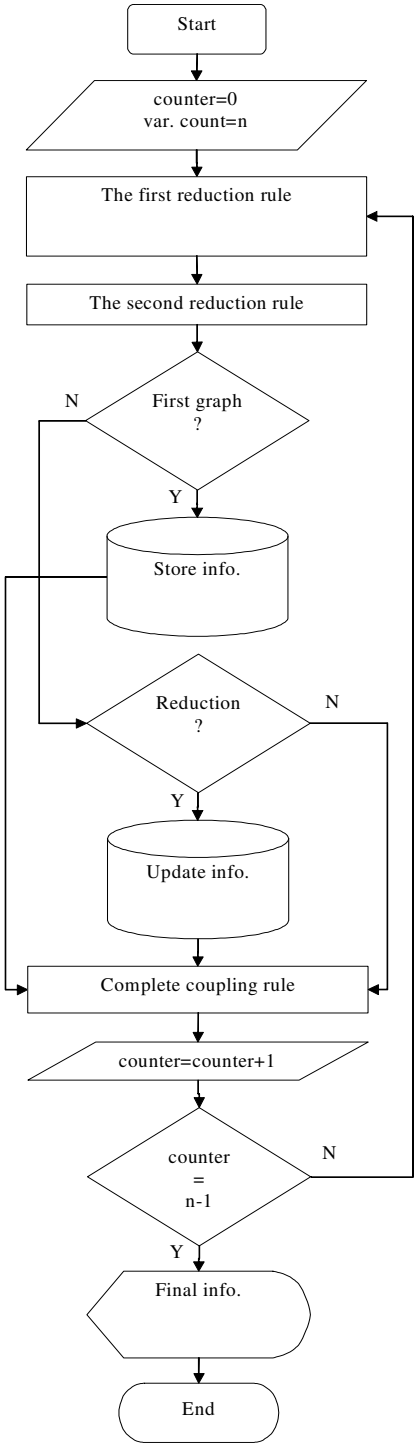


Fig. 7: Proposed Algorithm.

After applying the primary algorithm to the above mentioned graph, the number of nodes will be reduced from 13 to 4. Now if we analyze the coupling matrix for the resulted graph further simplification using the first rule can be obtained by applying the coupling algorithm to the root of the D.D. above. So one of the four resulting nodes is omitted and the design is completed using three nodes achieving, 25% more simplification in comparison to the common ROMDD. In other words, we have made an enhanced ROMDD that needs the least number of nodes to implement a function. This simplification will affect size and power dissipation.

CONCLUSION

In this paper we presented a new algorithm for free MDD graph design. By applying the coupling rule, as described in the paper, we can obtain better simplification in circuit implementation. This new rule referred to as the third simplifying rule, changes the designer's perspective leading to better simplification using the first and second rules. Simplifying is improved by 20% in the three implemented variable example function compare to the old algorithm. In multiple-value implementation 27% improvement and in other comparisons an improvement of more than 27% was achieved.

REFERENCES

1. D. M. Miller and G.W. Duec, 2003. On the size of multiple valued decision diagrams. 33rd IEEE International Symposium on Multiple-Valued Logic (ISMVL'03)
2. O. Kavehie, K. Navi, T. Nikoubin, M. Rouholamini, 2006. A Novel DCVS Tree Reduction Algorithm. IEEE International Conference on Integrated Circuit Design &Technology (ICICDT'06), pp. 1-7, 24-26
3. D. M. Miller and R. Drechsler, 2002. On the construction of multiple-valued decision diagrams. 32rd IEEE International Symposium on Multiple-Valued Logic (ISMVL'02), pp. 245-253
4. Dragan Jankovic, Radomir S. Stankovic, Rolf Drechsler, 2004. Reduction of Sizes of Multi-Valued Decision Diagrams by Copy Properties. 34rd IEEE International Symposium on Multiple-Valued Logic (ISMVL'04)
5. F. Somenzi, 2001. Efficient manipulation of decision diagrams. International Journal on Software Tools for Technology Transfer, 3, pp. 171-181
6. D. M. Miller and R. Drechsler, 1998. Implementing a multiple valued decision diagram package. 28rd IEEE International Symposium on Multiple-Valued Logic (ISMVL'98), pp. 52-57
7. M. Abd-El-Barr, and H. Fernandez, 1999. Synthesis of Multiple-Valued Decision Diagrams using Current-Mode CMOS Circuits. 29rd IEEE International Symposium on Multiple-Valued Logic (ISMVL'99), pp. 160-165
8. R. Drechsler, 2002. Evaluation of Static Variable Ordering Heuristics for MDD Construction. 32rd IEEE International Symposium on Multiple-Valued Logic (ISMVL'02)
9. D. M. Miller, 1993. Multiple-Valued Logic Design Tools. 23rd International Symposium on Multiple-Valued Logic, pp. 2-11
10. R. E. Bryant, 1986. Graph-based Algorithms for Boolean Function Manipulation. IEEE Transactions on Computers, Vol. 35, No. 8, pp. 677-691
11. S. Minato, N. Ishiura and S. Yajima, 1990. Shared Binary Decision Diagrams with Attributed Edges for Efficient Boolean Function Manipulation. ACM/IEEE Design Automation Conference (DAC'90), pp. 52-57
12. R. Rudell, 1993. Dynamic variable ordering for ordered binary decision diagrams. IEEE/ACM International Conference on Computer-Aided Design (ICCAD'93), pp. 43-47
13. F. Somenzi, CUDD: CU Decision Diagram Package, http://bessie.colorado.edu/~fabio
14. S. J. Friedman and K. J. Supowit, 1990. Finding the Optimal Variable Ordering for Binary Decision Diagrams. IEEE Transactions on Computers, Vol. 39, Issue. 5, pp: 710-713
15. Wolfgang Guenther, and Rolf Drechsler, 2003. Efficient Minimization and Manipulation of Linearly Transformed Binary Decision Diagrams. IEEE Transactions on Computers, Vol. 52, No. 9
16. M. Teslenko, A. Martinelli, E. Dubrova, 2005. Bound-Set Preserving ROBDD Variable Orderings May Not Be Optimum. IEEE Transactions on Computers, Vol. 54, No. 2, pp. 236-237
17. R. Ashenhurst, 1959. The Decomposition of Switching Functions. International Symposium Theory of Switching, Vol. 29, pp. 74-116
18. O. Kavehie, K. Navi, 2005. A Novel 54x54-bit Scalable Multiplier Architecture. 13th Iranian Conference on Electrical Engineering (ICEE'05), pp. 367-371
19. Yu-L. Wu, H. Fan, M.M. Sadowska, C.K. Wong, 2000. OBDD Minimization Based on Two-Level Representation of Boolean Functions. IEEE Transactions on Computers, Vol. 49, No. 12, pp. 1371-1379
20. Arunita Jaekel, Subir Bandyopadhyay, and Graham A. Jullien, 1998. Design of Dynamic Pass-Transistor Logic Circuits Using 123 Decision Diagrams. IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications, Vol. 45, No. 11
21. K.M. Chu and D.L. Pulfrey, 1987. A Comparison of CMOS Circuit Techniques: Differential Cascode Voltage Switch Logic versus Conventional Logic. IEEE Journal of Solid-State Circuits, Vol. SC-22, No. 4, pp. 528-532.