IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received August 2, 2019, accepted August 26, 2019, date of publication August 29, 2019, date of current version September 20, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2938257

A Novel Logic Detection Algorithm for Logic Circuits

ZHENXUE HE[1], JIA LIU[1], FAN ZHANG[1], TAO WANG[2], AND ZHISHENG HUO[2]
[1]College of Information Science and Technology, Hebei Agricultural University, Baoding 071001, China
[2]School of Computer Science and Engineering, Beihang University, Beijing 100191, China

Corresponding author: Zhenxue He (hezhenxue@buaa.edu.cn)

This work was supported in part by the Introducing Talent Research Project of Hebei Agricultural University under Grant YJ201829, in part by the Hebei Youth Talents Support Project under Grant BJ2019008, in part by the National Natural Science Foundation of China under Grant 61772053, Grant 60973106, Grant 61232009, Grant 81571142, and Grant 31801782, in part by the China Postdoctoral Science Foundation under Grant 2018M641154, in part by the Scientific Science and Technology Research Projects of Universities in Hebei under Grant BJ2018012, and in part by the Project of Hebei Natural Science Foundation under Grant G2018204093.

ABSTRACT Synthesizing logic circuits with different logic will result in different circuit structure, and then obtain different circuit performance. However, few studies have focused on logic detection technology that detects which logic is appropriate for the implementation of logic circuits. In this paper, we use Reduced Ordered Binary Decision Diagram (ROBDD) processed by dynamic variable sorting approach to describe logic circuits. Moreover, based on the proposed XOR logic judging condition and logic detection judging condition, we propose a Novel Logic Detection Algorithm (NLDA), which can quickly and effectively detect which logic is appropriate for the implementation of logic circuits. The experimental results on MCNC benchmark circuits show that compared to the logic detection algorithm based on minterms, the logic detection algorithm based on product terms, and the logic detection algorithm based on disjointed cubes, NLDA has the highest logic detection accuracy rate and highest logic detection efficiency.

INDEX TERMS Logic circuits, logic detection, ROBDD, dynamic variable sorting, product terms.

I. INTRODUCTION

Lots of research has shown that compared to circuits implemented by Boolean logic, circuits implemented by Reed-Muller (RM) logic have significant advantage in terms of area, power, speed, and testability [1]–[3]. Now, the open source Electronics Design Automation (EDA) software, represented by ABC and YoSys, and business EDA software, represented by Cadence and Synopsys, are mostly based on Boolean logic to optimize circuits. This makes the circuits, which are well-suited to RM logic implementation, can not obtain the best circuit performance at logic-level. On the contrary, the circuits those are well-suited to Boolean logic implementation are synthesized using RM logic, which will increase the complexity of circuit structure and reduce the circuit performance. Moreover, research shows that for the most circuits, using the dual logic optimization scheme combining Boolean logic and RM logic can remarkably improve circuit performance [4]–[7]. Therefore, it will optimize circuit structure and improve circuit performance to detect which logic is appropriate for the implementation of logic circuits before the circuits are realized.

The associate editor coordinating the review of this manuscript and approving it for publication was Yiyu Shi.

A dual logic expression of n-variables logic functions can be expressed as follows:

$$f(X) = f_1(X_1) \circ f_2(X_2) \ldots \circ f_i(X_i) \quad (1)$$

where $X = \{x_1, x_2, \ldots, x_n\}$ denotes the sets of input variables, '$\circ$' denotes a kind of logic operation such as AND, OR and XOR, $f_1, f_2, \ldots, f_i$ denote the sub-functions that implemented by Boolean logic or RM logic, $X_1, X_2, \ldots, X_i$ are the subset of X and satisfy $X_1 \cup X_2 \cup \ldots \cup X_i = X$, $i \geq 1$. Equation (1) will become a Boolean logic expression when $f_1, f_2, \ldots, f_i$ implemented by Boolean logic and '$\circ$' denotes AND or OR. Similarly, equation (1) will become a RM logic expression when $f_1, f_2, \ldots, f_i$ implemented by RM logic and '$\circ$' denotes AND or XOR. Therefore, Boolean logic and RM logic can be regarded as a special logic of dual logic, and the judgment and recognition of XOR logic relationship between variables is an important part for implementing logic detection of logic circuits.

Recently, there is little research on the logic detection approaches. The existing logic detection algorithms can be mainly divided into three classes through a survey of literature as follows: the logic detection algorithm based on minterms, the logic detection algorithm based on product terms, and the logic detection algorithm based on disjointed cubes. The logic detection algorithm based on minterms converts the Boolean function into the form of sum of minterms, and analyzes the logic mode of logic circuits by computing the Hamming distances between minterms. It is simple and easy to understand. In [4], the authors proposed a logic detection algorithm for logic circuits to benefit from dual logic implementation, which analyzes the logic mode of logic circuits according to the logic characteristic of XOR logic and Hamming distances between minterms. The logic detection algorithm based on product terms is an improvement to the first algorithm. It converts the Boolean function into the form of sum of the simplest product terms, and analyzes the logic mode of logic circuits by computing the Hamming distances between the simplest product terms. It has low memory overhead and is easy to understand. In [5], the authors proposed an algorithm for detecting dual logic based on product term, which analyzes the logic mode of logic circuits according to the logic characteristic of XOR logic and Hamming distances between product terms. The logic detection algorithm based on disjointed cubes is an improvement to the above two algorithms. It converts the Boolean function into the form of sum of disjointed cubes, and splits the Boolean function into different logic parts. Since the disjointed cubes have the characteristics that the minterms do not intersect each other and they have some characteristics of the simplest cubes, the logic detection algorithm based on disjointed cubes has a high logic detection accuracy rate. In [6] and [8], the authors proposed a logic detection algorithm, which converts the product terms of logic circuits into disjointed cubes and uses the Hamming distances between disjointed cubes to split the logic circuits into two parts.

However, the logic detection algorithm based on minterms is inefficiency when it deals with large-scale circuits, because the number of minterms increases exponentially with the increase in the number of input variables. Moreover, the logic detection algorithm based on product terms is difficult to implement, because the logic circuits simplification is always one of the difficulties in the field of logic synthesis and this algorithm can not guarantee that logic circuits simplification is effective and thorough. Furthermore, the logic detection algorithm based on disjointed cubes is also inefficient in detecting large-scale circuits, because this algorithm needs to perform sharp operation on product terms to produce disjointed cubes and the number of disjointed cubes will increase linearly with the increase in the number of input variables.

Compared to truth table, karnaugh map, and paradigm based on the sum of products, ROBDD has the advantages of small memory overhead, easy programming and high computational efficiency in describing logic circuits. Therefore, we use the ROBDD to describe logic circuits and introduce dynamic variable sorting approach [9] to remedy the defect that ROBDD is sensitive to variable sorting. Since the actual

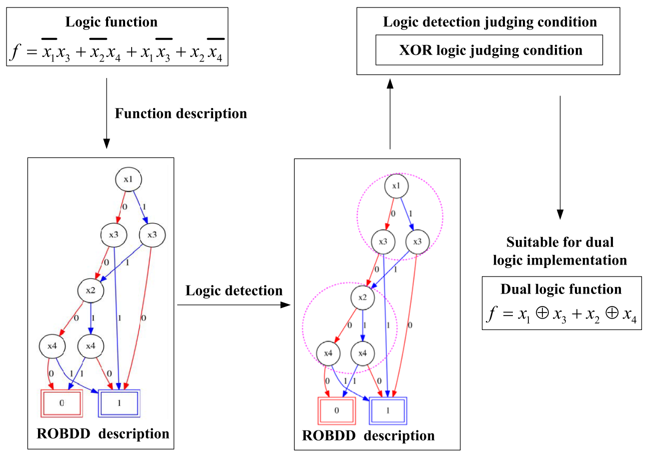
Logic function
$f = \overline{x_1}x_3 + \overline{x_2}x_4 + x_1\overline{x_3} + x_2\overline{x_4}$
Function description
ROBDD description
Logic detection
ROBDD description
Logic detection judging condition
XOR logic judging condition
Suitable for dual logic implementation
Dual logic function
$f = x_1 \oplus x_3 + x_2 \oplus x_4$

FIGURE 1. The flow of NLDA.

circuit performance can not be obtained during the logic detection phase, we start with the logic functions structure and propose a XOR logic judging condition by analyzing the XOR logic characteristic. Based on ROBDD description of logic circuits and XOR logic judging condition, we propose a logic detection judging condition. Moreover, based on the logic detection judging condition, we propose a novel logic detection algorithm, called NLDA, which can judge whether a logic circuit is suitable for implementation with RM logic, Boolean logic, or dual logic. The flow of NLDA is shown in Figure 1. Firstly, a logic function is described by a ROBDD that processed by the dynamic variable sorting approach. Secondly, the logic function is detected by the logic detection judging condition according to its ROBDD description. Lastly, an appropriate logic implementation for the logic function is obtained by NLDA.

The remainder of this paper is structured as follows. In Section II, we introduce RM expression and ROBDD. The XOR logic judging condition is described in Section III. A novel logic detection algorithm is described in Section IV. The experimental results are described in Section V. Our conclusions are presented in Section VI.

II. PRELIMINARIES

A. RM EXPRESSION

The well-known properties of RM logic make it an attractive candidate for logic circuit implementations. Firstly, RM expression (exclusive-OR sum of product form) generally requires fewer logic gates than the more traditional Boolean expression (sum-of-products form). Examples of many useful circuits are adders, parity checkers and arithmetic units, because they are heavily XOR oriented. Moreover, Programmable Logic Arrays (PLA's) implementing RM expressions randomly generated functions require, on the average, fewer product terms than standard PLA's implementing conventional Boolean expressions. Secondly, compared to Boolean expressions, circuit networks realized with RM expressions are easily testable, because the change in any input to an exclusive-OR/NOR function will always propagate to the output. Lastly, XOR gates are now used as basic cell components for Field-Programmable Gate Array (FPGA). Moreover, the layout technology has reduced the layout area required for XOR gates so that it is comparable with that of the other basic logic gates.

A n-variables Boolean function may be represented canonically in a sum-of-products form as [10], [11]

$$f(x_{n-1}, x_{n-2}, \ldots, x_0) = \sum_{i=0}^{2^n-1} a_i m_i \tag{2}$$

where Σ is an OR operator and m_i are the minterms. a_i are the coefficient of minterms, and $a_i = 1$ or 0 represents the presence or absence of minterms, respectively.

By applying shannon theorem [12], the Boolean function can be expressed as follows:

$$\begin{aligned} f &= \overline{x_{n-1}} f(0, x_{n-2}, \ldots, x_0) + x_{n-1} f(1, x_{n-2}, \ldots, x_0) \\ &= (1 \oplus x_{n-1}) f(0, x_{n-2}, \ldots, x_0) \oplus x_{n-1} f(1, x_{n-2}, \ldots, x_0) \\ &= f(0, x_{n-2}, \ldots, x_0) \oplus x_{n-1}[f(0, x_{n-2}, \ldots, x_0) \\ &\quad \oplus f(1, x_{n-2}, \ldots, x_0)] \end{aligned} \tag{3}$$

Therefore, the XOR/AND expansion corresponding to variant x_{n-1} is as follows:

$$f = f_0(x_{n-2}, \ldots, x_0) \oplus x_{n-1} f_1(x_{n-2}, \ldots, x_0) \tag{4}$$

where $f_1(x_{n-2}, \ldots, x_0) = f(0, x_{n-2}, \ldots, x_0) \oplus f(1, x_{n-2}, \ldots, x_0)$, and $f_0(x_{n-2}, \ldots, x_0) = f(0, x_{n-2}, \ldots, x_0)$.

By applying shannon theorem to each variant in turn, the Boolean function can be expressed by a RM expression as follows:

$$f^p(x_{n-1}, x_{n-2}, \ldots, x_0) = \oplus \sum_{i=0}^{2^n-1} b_i \pi_i \tag{5}$$

where $\oplus\Sigma$ denotes the modulo-2 addition, and $\pi_i = x_{n-1}x_{n-2}\ldots x_0$ represents the product term of a RM expression. $b_i \in \{0, 1\}$ represents whether or not π_i appears in the function, $p = (p_{n-1}p_{n-2}\ldots p_0)$ is the polarity, and $i = (i_{n-1}i_{n-2}\ldots i_0)$ is the subscript.

B. ROBDD

ROBDD can be obtained by removing redundancy nodes and isomorphism sub-graphs of Ordered Binary Decision Diagram (OBDD) [13], [14]. Although OBDD can be used as the regular expression of logic circuits, its structure is not necessarily the simplest because OBDD may have redundancy nodes or isomorphism sub-graphs. Compared with OBDD, ROBDD can represent and manipulate logic circuits more efficiently because of more simplified structure.

However, the number of nodes and tree depth of ROBDD are susceptible to variable sorting [15], [16]. In other words, different variables sorting will get different ROBDD. A good variable sorting can greatly simplify ROBDD structure, and thus improve the efficiency of variable operational algorithms.

Example 1: Given a logic function $f = x_1x_2 + x_3x_4 + x_5x_6$, if the variable sorting is $x_1, x_2, x_3, x_4, x_5, x_6$, then the ROBDD description is shown in Figure 2(a); if the variable sorting is $x_1, x_3, x_5, x_2, x_4, x_6$, then the ROBDD description is shown in Figure 2(b).

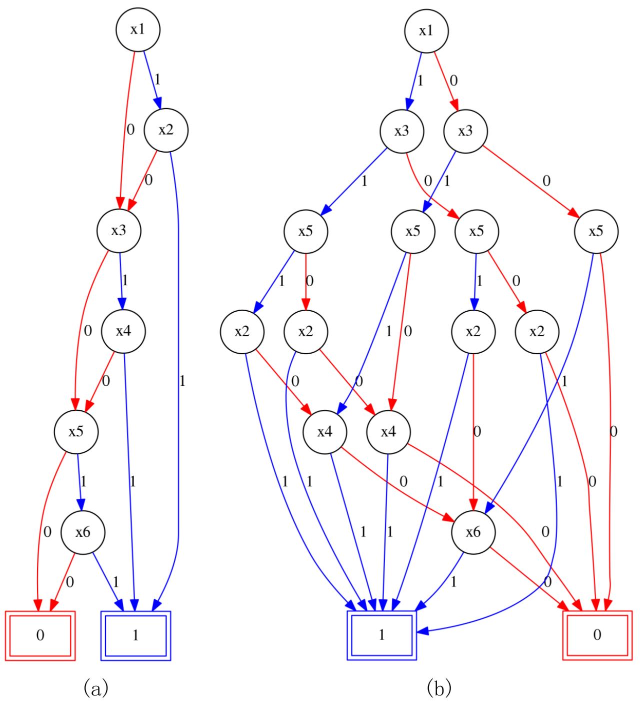

FIGURE 2. ROBDD description.

III. XOR LOGIC JUDGING CONDITION

As described in Section II, the number of nodes and tree depth of ROBDD are susceptible to variable sorting. We introduce the dynamic variable sorting approach to remedy the defect that ROBDD is sensitive to variable sorting, to further reduce the number of nodes and further simplify ROBDD structure. For the convenience of description, we abbreviate the ROBDD optimized by dynamic variable sorting approach as Minimal ROBDD (MROBDD). In this paper, we use MROBDD to describe logic circuits, which can improve the operational efficiency of logic function and thus improve logic detection efficiency.

Example 2: Figure 3(a) and Figure 3(b) are the ROBDD description and MROBDD description of the logic function $f = \overline{x_1}x_3 + \overline{x_2}x_4 + x_1\overline{x_3} + x_2\overline{x_4}$, respectively. As shown in Figure 3, compared to the ROBDD description of the logic function, the MROBDD description of that has a more simplified structure and has fewer nodes.

Since the actual circuit performance can not be obtained at logic synthesis stage, we start with the logic function structure, and propose a XOR logic judging condition by analyzing the XOR logic characteristic. Figure 4(a) and Figure 4(b) are the MROBDD descriptions of the logic functions $f_1 = \overline{x_1}x_2 + x_1\overline{x_2}$ and $f_2 = x_1x_2 + \overline{x_1}\,\overline{x_2}$, respectively. If logic functions f_1 and f_1 are implemented by RM logic, then $f_1 = x_1 \oplus x_2$, $f_2 = \overline{x_1} \oplus x_2$. Obviously, compared to the Boolean logic implementation form of two functions, the RM logic

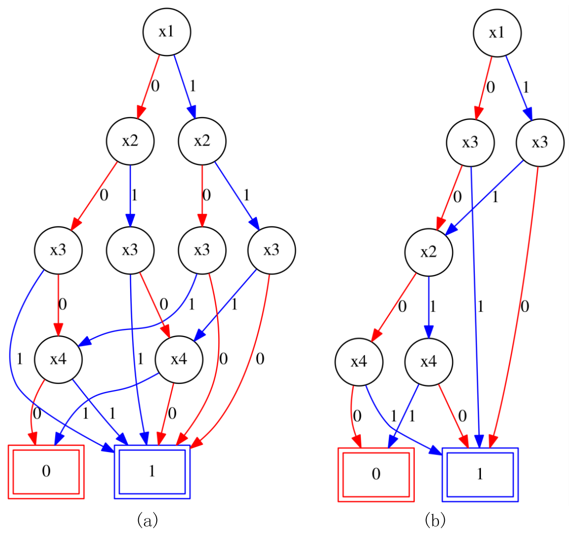

FIGURE 3. ROBDD description and MROBDD description.

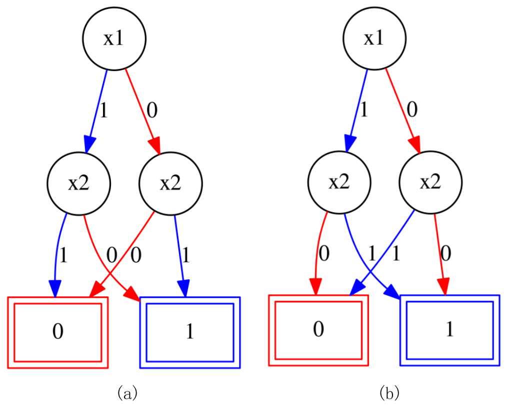

FIGURE 4. MROBDD description.

implementation form of that can reduce two variables. Therefore, it can offer foundation for detecting which logic is appropriate for the implementation of logic circuits to analyze whether the binary tree part shown in Figure 4 exists in MROBDD description of logic circuits.

A. XOR LOGIC JUDGING CONDITION

For the MROBDD description of a n-variables logic function $f(x_1, x_2, \ldots, x_n)$, if node v_{father}, non-terminal children nodes v_{left} and v_{right} exist, and the following two conditions are satisfied, then x_{father} and x_{left} are suitable for implementation in XOR logic.

$$f|_{x_{father}=0,x_{left}=0} = f|_{x_{father}=1,x_{right}=1} \tag{6}$$

$$f|_{x_{father}=0,x_{left}=1} = f|_{x_{father}=1,x_{right}=0} \tag{7}$$

where $v_{left} = low(v_{father})$, $v_{right} = high(v_{father})$, x_{father} is the variable of node v_{father}, x_{left} is the variable of node v_{left}, and x_{right} is the variable of node v_{right}, $x_{left} = x_{right}$.

Prove: The binary tree part that satisfies XOR logic judging condition is shown in Figure 5, F_1 and F_2 are functions or

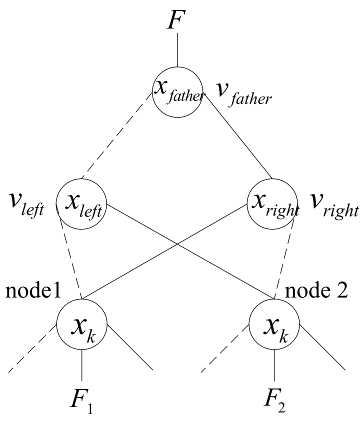

FIGURE 5. MROBDD description that satisfies XOR logic judging condition.

values that correspond to node 1 and node 2. The following formula can be obtained according to the nature of ROBDD describes logic functions.

$$\begin{aligned} F &= \overline{x_{father}x_{left}}F_1 + x_{father}x_{right}F_1 + \overline{x_{father}}x_{left}F_2 \\ &\quad + x_{father}\overline{x_{right}}F_2 \\ &= (\overline{x_{father}x_{left}} + x_{father}x_{right})F_1 \\ &\quad + (\overline{x_{father}}x_{left} + x_{father}\overline{x_{right}})F_2 \end{aligned} \tag{8}$$

Since the variables of each layer nodes in MROBDD are the same, namely, $x_{left} = x_{right}$, the following formula can be obtained.

$$\begin{aligned} F &= (\overline{x_{father}x_{left}} + x_{father}x_{left})F_1 + (\overline{x_{father}}x_{left} + x_{father}\overline{x_{left}})F_2 \\ &= (\overline{x_{father}} \oplus x_{left})F_1 + (x_{father} \oplus x_{left})F_2 \end{aligned} \tag{9}$$

Therefore, the x_{father} and x_{left} that satisfy the XOR logic judging condition are suitable for implementation in XOR logic.

Example 3: Figure 6 is MROBDD description of $f = \sum(m_1, m_2, m_3, m_4, m_6, m_7, m_8, m_9, m_{11}, m_{12}, m_{13}, m_{14})$. If this function is implemented in Boolean logic, it can be simplified to $f = \overline{x_1}x_3 + \overline{x_2}x_4 + x_1\overline{x_3} + x_2\overline{x_4}$, and this simplified function has 8 variables. Since the variables x_1,x_3,x_2 and x_4 satisfy the XOR logic judging condition, this function can be expressed as $f = x_1 \oplus x_3 + x_2 \oplus x_4$. Compared to Boolean logic implementation form of this function, XOR logic implementation form of that saves 4 variables.

IV. NOVEL LOGIC DETECTION ALGORITHM

In this section, based on ROBDD description of logic circuits and XOR logic judging condition, we propose a logic detection judging condition. Moreover, based on the XOR logic judging condition and logic detection judging condition, we propose a novel logic detection algorithm, called NLDA, which detects which logic is appropriate for the implementation of logic circuits.

For the convenience of description, we abbreviate the binary tree part that is suitable for expression in XOR logic as BT_XOR. Moreover, we abbreviate the layer that only

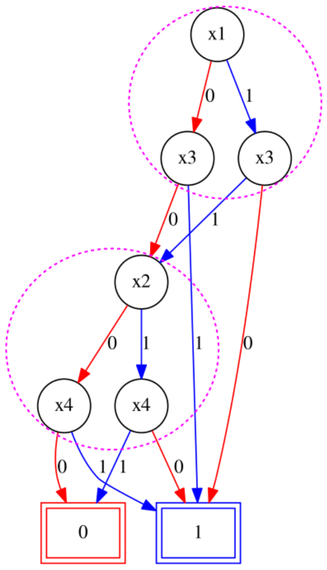

FIGURE 6. MROBDD description.

contains one node with a low edge pointing to the '0' terminal node as single layer. A BT_XOR is represented by XOR logic, which can save 2 variables. Therefore, the more BT_XORs in MROBDD description, the more variables are saved and the greater the improvement of circuit performance.

Example 4: If the logic function corresponding to Figure 7 is implemented with Boolean logic, it can be simplified to $f = \overline{x_1}x_2 + x_1\overline{x_2} + x_2\overline{x_3}x_4 + x_2x_3\overline{x_4} + x_4\overline{x_5}x_6 + x_4x_5\overline{x_6} + x_6\overline{x_7}x_8 + x_6x_7\overline{x_8}$, and it has 22 variables. Since there are 4 BT_XORs in the MROBDD description, this function can also be expressed as $f = x_1 \oplus x_2 + x_2(x_3 \oplus x_4) + x_4(x_5 \oplus x_6) + x_6(x_7 \oplus x_8)$. Compared to Boolean logic implementation form of this function, XOR logic implementation form of that saves 11 variables.

A. LOGIC DETECTION JUDGING CONDITION

1) LOGIC DETECTION JUDGING CONDITION

The logic judging result of a logic circuit can be obtained by verifying layer by layer whether there is a BT_XOR between two adjacent layers from the zeroth layer to the end of the second layer of MROBDD description.

(1) If there is a BT_XOR between the adjacent two layers except the single layer, then the logic circuit is suitable for implementation with RM logic.

(2) If BT_XOR does not exist, namely no pair of product terms that are suitable for implementation with RM logic are found, then the logic circuit is suitable for implementation with Boolean logic.

(3) If BT_XOR exists and the condition that a BT_XOR exists between two adjacent layers is not satisfied, namely not all product terms are suitable for implementation with RM logic, then the logic circuit is suitable for implementation with dual logic.

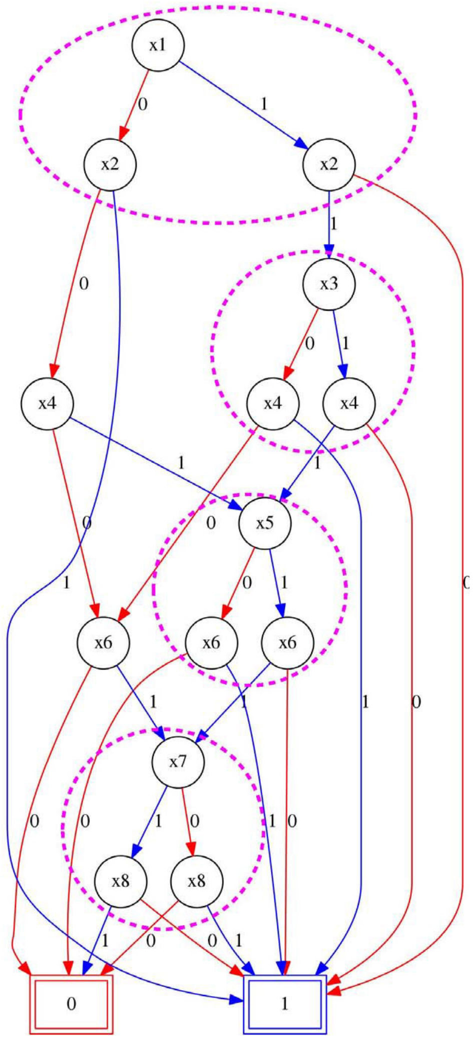

FIGURE 7. MROBDD description.

Example 5: As shown in Figure 8, since a BT_XOR exists between zeroth layer and first layer in Figure 8(a), the logic circuit is suitable for implementation with RM logic, namely, $f_a = x_1 \oplus x_2$; since a BT_XOR exists between second layer and third layer except the zeroth single layer and first single layer in Figure 8(b), the logic circuit is suitable for implementation with RM logic, namely, $f_b = x_1x_2(x_3 \oplus x_4)$; since a BT_XOR exists between zeroth layer and first layer except the second single layer and third single layer in Figure 8(c), the logic circuit is suitable for implementation with RM logic, namely, $f_c = x_2x_3(x_1 \oplus x_4)$; since BT_XORs exists between zeroth layer and first layer, first layer and second layer, second layer and third layer in Figure 8(d), the logic circuit is suitable for implementation with RM logic, namely, $f_d = x_1 \oplus x_2 \oplus x_3 \oplus x_4$.

Example 6: As shown in Figure 9, since there is no BT_XOR in Figure 9(e) and Figure 9(f), the logic circuits that correspond to Figure 9(e) and Figure 9(f) are suitable for implementation with Boolean logic, namely, $f_e = x_1x_2x_3 + \overline{x_1}x_2$, $f_f = x_1x_2 + x_3x_4 + x_5x_6$; since a BT_XOR only exists

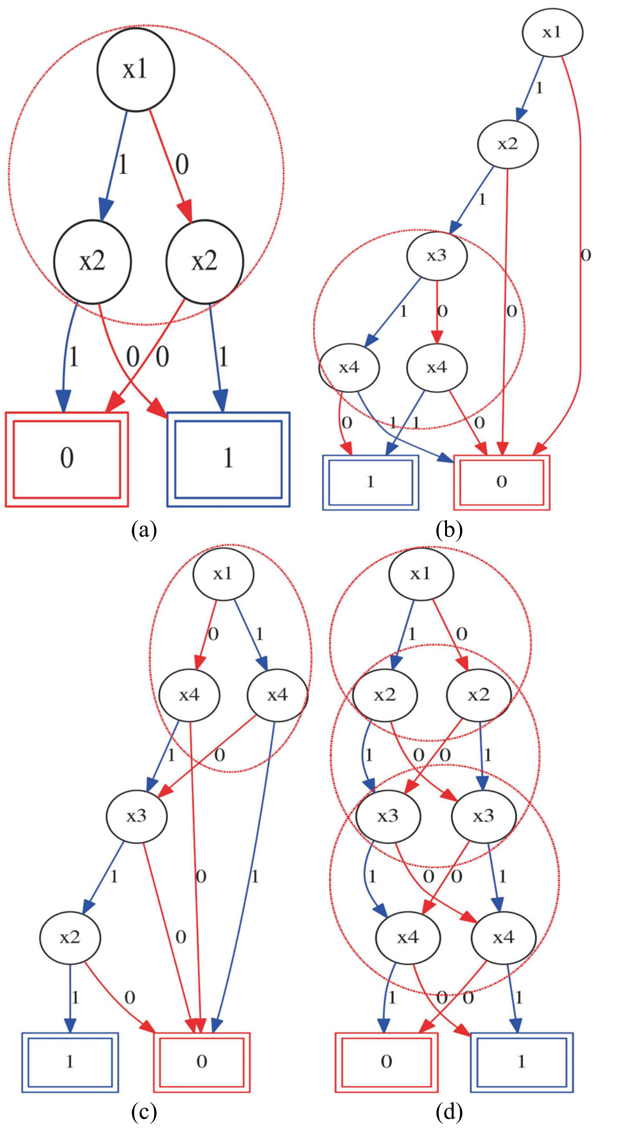

FIGURE 8. MROBDD description.

between zeroth layer and first layer in Figure 9(g), the logic circuit is suitable for implementation with dual logic, namely, $f_g = x_1 \oplus x_3 + x_2(\overline{x_3} \oplus x_4)$; since there is no BT_XOR between first layer and second layer in Figure 9(h), the logic circuit is suitable for implementation with dual logic, namely, $f_h = x_1 \oplus x_3 + x_2 \oplus x_4$.

B. ALGORITHM DESCRIPTION

Based on the above description, the pseudo code of NLDA is described in Algorithm 1, where 'L' denotes the number of layers of MROBDD. As shown in Algorithm 1, NLDA consists of two parts: the generation of dynamically sorted ROBDD and logic detection. Since the time complexity for reducing BDD to ROBDD is O(mlogm) and the time complexity of dynamic variable sorting is O(n^2), the time complexity of the generation of dynamically sorted ROBDD is O(mlogm)+ O(n^2), where m denotes the number of nodes and n denotes the number of input variables. Since the logic detection is implemented by verifying layer by layer whether there is a BT_XOR between two adjacent layers from the zeroth layer to the end of the second layer, the time complexity of logic detection is O(k), where k denotes the number of layers.

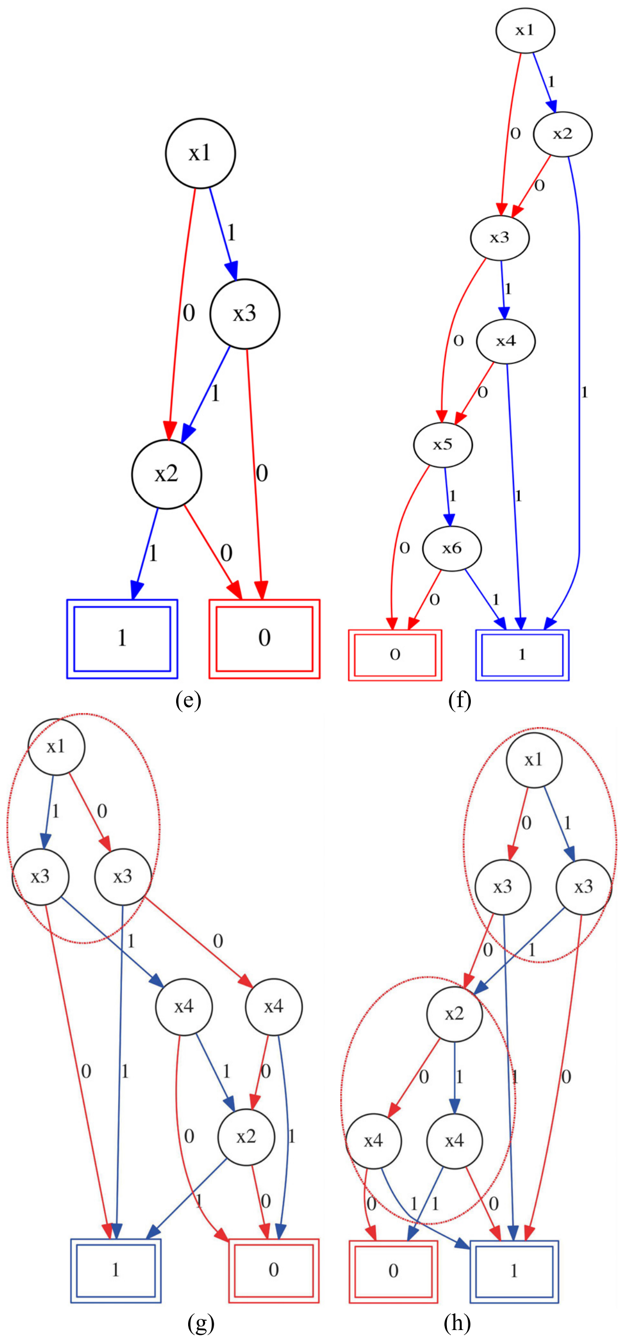

FIGURE 9. MROBDD description.

V. EXPERIMENTAL RESULTS

CMU BDD software package is an effective tool for generating and manipulating ROBDD. Therefore, the proposed NLDA was realized by programming based on CMU BDD

TABLE 1. The pseudo code of NLDA.

```
Algorithm 1: Pseudo code of NLDA
Begin
  Read a logic circuit;
  Obtain MROBDD description of the logic circuit;
  for  $i = 0 : L-1$ ( $L$ layers) do
    if the  $i$ th layer belong to single layer then
      continue;
    end  if
    Verify whether there is a BT_XOR between two adjacent layers;
  end for
  if BT_XOR exists between two adjacent layers except single layer then
    The logic circuit is suitable for implementation with RM logic;
  end if
  else if there is no BT_XOR between two adjacent layers then
    The logic circuit is suitable for implementation with Boolean logic;
  end else if
  else
    The logic circuit is suitable for implementation with dual logic;
End
```

software package. Moreover, the dynamic variable sorting was embedded in the CMU BDD software package as a background process. Therefore, the dynamically sorted ROBDD can take advantage of all the techniques in the CMU BDD software package to speed up ROBDD generation. Furthermore, in addition to using the memory management strategies of CMU BDD software package, we also deleted the node name field (vertex_name: string) of node data structure in CMU BDD software package, because each variable x in ROBDD uniquely corresponds to an index(x) and a variable can be recognized based on the index of the variable. For complex circuits with more nodes, this can save a lot of memory. Moreover, identifying nodes based on indexes is much faster than that based on variable names.

The experimental results were obtained by using a PC with Intel Core i7 3.40 GHz with 4G RAM under Linux. Moreover, the Microelectronics Center of North Carolina (MCNC) benchmark circuits were used as experimental circuits. Specifically, the format of experimental circuits is combinational two-level logic in Berkeley PLA. The PLA description mainly includes the following keywords: 1 .i[d] (specifies the number of input variables) 2 .o[d] (specifies the number of output functions) 3 .type[s] (sets the logic interpretation of the character matrix) 4 .p[d] (specifies the number of product terms) 5 .e (marks the end of PLA description).

We used the existing logic detection algorithms, namely, Minterms based Logic Detection Algorithm (MLDA) [4], Product Terms based Logic Detection Algorithm (PTLDA) [5], and Disjointed Cubes based Logic Detection Algorithm (DCLDA) [6], as comparison algorithms to verify the logic detection accuracy rate and efficiency of NLDA. Furthermore, in order to ensure the fairness of experimental results, one part of experimental circuits were selected from literatures [4]–[6], and another part of that were MCNC benchmark circuits containing small-scale circuits and large-scales circuits.

Logic detection efficiency and logic detection accuracy rate are two important performance indicators for logic detection algorithms. Therefore, we used the logic detection time and logic detection result of each experimental algorithm as experimental data to evaluate the comprehensive performance of experimental algorithms. The comparison of NLDA, MLDA, PTLDA, and DCLDA is shown in Table 2, where 'Circuits' denotes the circuit name, 'i/o/p' denotes the number of input variables, the number of output variables, and the number of product terms. 'time(s)' denotes the run time (in second) of experimental algorithms. 'Result' denotes the logic detection results, where 'B' stands for suitable for Boolean logic implementation, 'R' stands for suitable for RM logic implementation, and 'BR' stands for suitable for dual logic implementation. 'Save1' shows the percentage of time saved by NLDA compared to MLDA, which is defined as

$$Save1 = ((MLDA_{time} - NLDA_{time})/MLDA_{time})^{*}100\% \tag{10}$$

'Save2' shows the percentage of time saved by NLDA compared to PTLDA, which is defined as

$$Save2 = ((PTLDA_{time} - NLDA_{time})/PTLDA_{time})^{*}100\% \tag{11}$$

'Save3' shows the percentage of time saved by NLDA compared to DCLDA, which is defined as

$$Save3 = ((DCLDA_{time} - NLDA_{time})/DCLDA_{time})^{*}100\% \tag{12}$$

where $NLDA_{time}$, $MLDA_{time}$, $PTLDA_{time}$, and $DCLDA_{time}$ denotes the run time of NLDA, MLDA, PTLDA, and DCLDA, respectively. 'Save1', 'Save2' and 'Save3' can intuitively demonstrate the performance of NLDA in logic detection efficiency compared to the existing logic detection algorithms.

From the Table 2, we can see that the logic detection results of NLDA and DCLDA were the same and consistent with the results of literatures [17], [18], which verified the logic detection accuracy rate of NLDA. However, MLDA and PTLDA can only give results that are suitable for dual logic implementation or for Boolean logic implementation, and can not judge whether a circuit is suitable for RM logic implementation. Moreover, the detection results of rd73, rd84, Apex4, and other circuits that obtained by MLDA and PTLDA were wrong. The mainly reason for the above results is that the logic detection conditions of MLDA and PTLDA have certain limitations, these logic detection conditions are only sufficiency and not necessity, namely, MLDA and PTLDA are prone to misjudgment for those circuits that contain XOR logic and do not meet the judgment condition. Furthermore, 10 of the 16 test circuits (namely, 62.5% circuits) were suitable for dual logic implementation, which is consistent with the statement that most circuits are suitable for dual logic implementation.

Moreover, we can also conclude from the Table 2 that NLDA had the highest logic detection efficiency, PTLDA and DCLDA is the second, and MLDA had the lowest logic

TABLE 2. Comparison of MLDA, PTLDA, DCLDA, and NLDA.

Circuits	i/o/p	MLDA		PTLDA		DCLDA		NLDA		time(%)		
		time(s)	Result	time(s)	Result	time(s)	Result	time(s)	Result	Save1	Save2	Save3
xor5	5/1/16	0.66	B	0.49	R	0.54	R	0.41	R	37.88	16.33	24.07
rd73	7/3/141	0.83	BR	0.67	BR	0.70	R	0.52	R	37.35	22.39	25.71
Inc	7/9/35	0.79	B	0.56	B	0.65	B	0.48	B	39.24	14.29	26.15
rd84	8/4/256	1.17	B	0.90	B	0.98	R	0.73	R	37.61	18.89	25.51
Apex4	9/19/438	2.58	B	2.23	B	2.04	BR	1.35	BR	47.67	39.46	33.82
Clip	9/5/167	2.73	B	1.92	BR	1.61	BR	0.94	BR	65.57	51.04	41.61
Sao2	10/4/58	3.89	B	2.76	B	0.83	BR	0.66	BR	83.03	76.09	20.48
Misex3	14/14/1848	67.10	B	40.37	B	23.45	BR	8.18	BR	87.81	79.74	65.12
Alu4	14/8/1028	56.04	BR	33.59	BR	14.92	BR	5.85	BR	89.56	82.58	60.79
Table3	14/14/175	74.84	B	45.62	BR	7.33	BR	3.82	BR	94.90	91.63	47.89
b12	15/9/431	164.65	B	94.81	BR	9.26	BR	2.54	BR	98.46	97.32	72.57
t481	16/1/481	248.29	BR	136.50	BR	10.84	BR	4.57	BR	98.16	96.65	57.84
Parity	16/1/32768	263.84	BR	158.92	BR	115.17	R	19.50	R	92.61	87.83	83.07
Cmb	16/4/54	251.40	BR	172.09	BR	5.63	R	2.41	R	99.04	98.60	57.19
Table5	17/15/158	573.26	B	430.68	B	9.54	BR	3.49	BR	99.39	99.19	63.42
Tcon	17/16/32	612.35	B	484.90	B	3.27	BR	1.04	BR	99.83	99.79	68.20

detection efficiency. Compared to MLDA, the highest percentage of logic detection time saved by NLDA reached 99.83%. Compared to PTLDA, the highest percentage of logic detection time saved by NLDA reached 99.79%. Compared to DCLDA, the highest percentage of logic detection time saved by NLDA reached 83.07%. The above results can be explained by the following reasons: firstly,

MLDA is inefficiency when it deals with large-scale circuits or even impossible to process, because MLDA detects logic circuits based on minterms and the number of minterms increases exponentially with the increase in the number of input variables; secondly, PTLDA needs to convert the logic function into the form of sum of the simplest terms and detects the logic circuits based on the simplest terms. However, for large-scale circuits or complex circuits, PTLDA is inefficiency because converting product terms to simplest terms will consume a lot of time. Moreover, the simplification of logic circuits has always been a difficult point in the field of logic synthesis, PTLDA can not guarantee that logic circuits simplification is effective and thorough; thirdly, DCLDA needs to perform sharp operation on product terms to produce disjointed cubes, and needs to compute the Hamming distances between product terms. Therefore, DCLDA is less efficient in processing the circuits with more product terms; lastly, NLDA describes the logic circuits based on the dynamically sorted ROBDD, which make it has higher logic detection efficiency. Moreover, NLDA directly analyzes and detects the circuits according to the XOR logic characteristics, which make it has higher logic detection accuracy rate.

VI. CONCLUSION

In this paper, we proposed a novel logic detection algorithm, called NLDA, which uses MROBDD to describe logic circuits and detects the logic circuits based on XOR logic judging condition and logic detection judging condition. Experimental results on MCNC benchmark circuits show that compared to MLDA, PTLDA, and DCLDA, NLDA has the highest logic detection accuracy rate and highest logic detection efficiency. Moreover, dynamic variable sorting approach does not work for the logic circuits whose ROBDD scales are not affected by variable sorting. In future work, we will study this problem to further improve the logic detection efficiency of NLDA.

REFERENCES

[1] S. Shirinzadeh, M. Soeken, P.-E. Gaillardon, and R. Drechsler, "Logic synthesis for RRAM-based in-memory computing," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 37, no. 7, pp. 1422–1435, Jul. 2018.
[2] Z. He, J. Liu, L. Xiao, Z. Huo, and X. Wang, "A polarity optimization algorithm taking into account polarity conversion sequence," *IEEE Access*, vol. 7, pp. 54809–54818, 2019.
[3] S. D. Kumar, H. Thapliyal, and A. Mohammad, "FinSAL: FinFET-based secure adiabatic logic for energy-efficient and DPA resistant IoT devices," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 1, pp. 110–122, Jan. 2017.
[4] Y. Xia, K. Mao, and X. Ye, "Detection algorithm for logic functions to benefit from dual logic implementation," *J. Comput.-Aided Des. Comput. Graph.*, vol. 19, no. 12, pp. 1522–1527, 2007.

[5] X.-E. Ye, K.-Y. Mao, and Y.-S. Xia, "Algorithms for detecting dual logic based on product term," *Acta Electron. Sinica*, vol. 37, no. 5, pp. 961–965, 2009.

[6] L. Y. Wang, Y.-S. Xia, X.-X. Chen, and X.-E. Ye, "Logic detection and decomposition algorithm based on disjointed cubes," *Acta Electron. Sinica*, vol. 40, no. 10, pp. 2091–2096, 2012.

[7] Y.-H. Wang, L.-Y. Wang, and Y.-S. Xia, "FPRM conversion using parallel tabular technique with disjointed products," *J. Electron. Inf. Technol.*, vol. 36, no. 9, pp. 2258–2264, 2014.

[8] L. Wang, Y. Xia, and X. Chen, "Logic synthesis and optimization based on dual logic," *J. Comput.-Aided Des. Comput. Graph.*, vol. 24, no. 7, pp. 961–967, 2012.

[9] S. Chatterjee, R. B. Venkata, and K. V. Gajendra, "An improved algorithm for K-terminal probabilistic network reliability analysis," *J. Rel. Stat. Stud.*, vol. 10, no. 1, pp. 15–26, 2017.

[10] A. Bernasconi, V. Ciriani, L. Frontini, V. Liberali, G. Trucco, and T. Villa, "Enhancing logic synthesis of switching lattices by generalized Shannon decomposition methods," *Microprocessors Microsyst.*, vol. 56, pp. 193–203, Feb. 2018.

[11] A. Neutzling, M. G. A. Martins, V. Callegaro, A. I. Reis, and R. P. Ribas, "A simple and effective heuristic method for threshold logic identification," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 5, pp. 1023–1036, May 2018.

[12] D. Ellerman, "Logical entropy: Introduction to classical and quantum logical information theory," *Entropy*, vol. 20, no. 9, pp. 679–681, 2018.

[13] T. Shah, A. Matrosova, M. Fujita, and V. Singh, "Multiple stuck-at fault testability analysis of ROBDD based combinational circuit design," *J. Electron. Test.*, vol. 34, no. 1, pp. 53–65, 2018.

[14] A. Kalaee and V. Rafe, "An optimal solution for test case generation using ROBDD graph and PSO algorithm," *Qual. Rel. Eng. Int.*, vol. 32, no. 7, pp. 2263–2279, 2016.

[15] A. Y. Matrosova, I. E. Kirienko, V. V. Tomkov, and A. A. Miryutov, "Reliability of physical systems: Detection of malicious subcircuits (trojan circuits) in sequential circuits," *Russian Phys. J.*, vol. 59, no. 8, pp. 1281–1288, 2016.

[16] A. U. Hassen, D. Chakraborty, and S. K. Jha, "Free binary decision diagram-based synthesis of compact crossbars for in-memory computing," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 65, no. 5, pp. 622–626, May 2018.

[17] L. Wang and A. E. A. Almaini, "Optimisation of reed-muller PLA implementations," *IEE Proc.-Circuits, Devices Syst.*, vol. 149, no. 2, pp. 119–128, Apr. 2002.

[18] B. A. Jassani, N. Urquhart, and A. E. A. Almaini, "Manipulation and optimisation techniques for Boolean logic," *IET Comput. Digit. Techn.*, vol. 4, no. 3, pp. 227–239, 2010.

ZHENXUE HE received the Ph.D. degree in computer architecture from Beihang University, Beijing, China, in 2018. He is currently a Full Associate Professor with Hebei Agricultural University. He has authored or coauthored more than 20 articles in peer-reviewed journals and proceedings. His research interests include low power integrated circuit design and optimization, multiple-valued logic circuits, and intelligent algorithm. He is a member of China Computer Federation.

JIA LIU was born in Tangshan, Hebei, China, in 1988. She received the M.E. degree from Hebei University, Baoding, China, in 2019. She is currently a Staff with the College of Information Science and Technology, Hebei Agricultural University. Her research interests include electronics design automation, intelligent algorithm, finance data mining, and data analysis.

FAN ZHANG received the M.S. degree from the College of Information Science and Technology, Hebei Agricultural University, Baoding, China, in 2009, and the Ph.D. degree in agricultural informationization from Hebei Agricultural University, in 2012, where she is currently a Lecturer. Her research interests include big data, agricultural informationization, and electronics design automation.

TAO WANG was born in Qianjin, Heyang, Shaanxi, in 1986. He is currently pursuing the Ph.D. degree in microelectronics and solid electronics from Beihang University, Beijing, China, where he has been with the School of Electronic and Information Engineering, since 2014. His current research interests include data fusion, target localization and tracking, and space-sky information networks.

ZHISHENG HUO received the M.S. degree from the College of Computer Science, Shenyang Aerospace University, Shenyang, China, in 2012, and the Ph.D. degree in computer architecture from Beihang University, Beijing, China, in 2018, where he currently holds a postdoctoral position at the School of Computer Science and Engineering. His research interests include big data storage and distributed storage systems.