Hindawi
Active and Passive Electronic Components
Volume 2019, Article ID 8292653, 9 pages
https://doi.org/10.1155/2019/8292653

Research Article

BDD-Based Topology Optimization for Low-Power DTIG FinFET Circuits

Haiyan Ni, Jianping Hu, Xuqiang Zhang, and Haotian Zhu

Faculty of Information Science and Technology, Ningbo University, Ningbo 315211, China

Correspondence should be addressed to Haiyan Ni; nihaiyan@nbu.edu.cn

Received 1 February 2019; Accepted 12 June 2019; Published 18 July 2019

Academic Editor: Gerard Ghibaudo

This paper proposed a logic synthesis method based on binary decision diagram (BDD) representation. The proposed method is optimized for dual-threshold independent-gate (DTIG) FinFET circuits. The algorithm of the BDD-based topology optimization is stated in detail. Some kinds of feature subgraph structures of a BDD are extracted by the extraction algorithm and then fed to mapping algorithm to get a final optimized circuit based on predefined DTIG FinFET logic gates. Some MCNC benchmark circuits are tested under the proposed synthesis method by comparing with ABC, DC tools. The simulations show that the proposed synthesis method can obtain performance improvement for DTIG FinFET circuits.

1. Introduction

As a 3D transistor, FinFET is more efficient than the traditional devices because it can suppress short channel effect (SCE) and drain induced barrier lowering (DIBL) [1–6]. FinFET can operate in common-gate (CG) mode, whose two gates are tied together naturally and can be used like a traditional single-gate device with improved performances, or in independent-gate (IG) mode, whose two gates can be used as two separated single-gate devices in parallel or series on special conditions [2].

Today's VLSI is usually designed by "standard cell" method [7]. Among the design flow, synthesis is an important process, which transforms the high level design to low level netlists form composed by standard cells. Currently, standard cell libraries are basically built on the basis of CMOS or CG FinFET devices. The synthesis tools, such as commercial tools like Synopsys Design Compiler (DC), public-domain tools like ABC [8], and synthesis algorithms like factorization-based methods, usually utilizing these single-gate standard cell libraries to optimize the circuit topology. However, according to our research, DTIG FinFET-based circuits have excellent performances and can be used in modern VLSI circuits [9–11]. So it has an emerging need to develop a comprehensive method based on DTIG FinFETs.

As a powerful representation of logic function, binary decision diagram (BDD) is widely used in computer science to solve graph algorithms and matrix-operation, even artificial intelligence problems. It also can be applied in VLSI design to construct the topology of the circuits and detect and optimize the circuits [12–14]. The efficiency of the BDD is always determined by the variable order of the functions. Since the variable ordering problem is a NP-complete [15, 16], there are many approximation heuristic methods have been found to efficiently solve this problem [12, 17–19].

The paper described a circuit topology optimization method based on BDD technology to compose the DT IG FinFET circuits by using the predefined DTIG FinFET basic logic cells. It is organized as follows. In Section 2, we introduce the predefined DTIG FinFET logic gates briefly, and in Section 3 we state some theorems and an algorithm of feature modules extraction in BDD graph. The mapping algorithm is also included in Section 3. And in Section 4, we realize the algorithms and verify the effectiveness of the method by testing the MCNC benchmarks. Finally, we conclude in Section 5.

2. DTIG FinFET Cell Library

Compared with single-gate devices, such as CMOS or CG FinFET, DTIG FinFET can design more flexible circuits by

TABLE 1: Performance comparison of some example cells.

Cells	T.Count		Delay (ps)		Power (μW)	
	CG	IG	CG	IG	CG	IG
NAND	4	2	5.6	5.1	11.8	5.5
NOR	4	2	5.5	4.7	11.8	3.0
NAND3	6	4	9.2	9.7	8.5	4.9
AOI	6	4	7.7	7.3	15.6	5.9
OAI	6	4	6.6	6.4	16.8	9.5
XOR	10	8	4.0	3.4	54.3	31.7

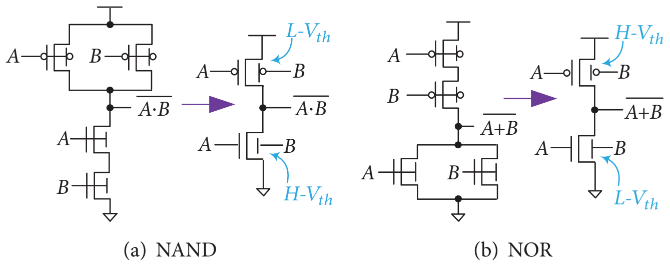

FIGURE 1: The basic logic cells realized by the CG FinFETs and DTIG FinFETs, respectively. (a) NAND, (b) NOR.

using of the low-threshold (low-V_{th}) and high-threshold (high-V_{th}) devices [2, 3, 9, 11, 20]. We have built a mini DTIG FinFET logic cells library for further using and as shown in Figure 1 are two examples and their CG comparisons. The DTIG FinFET logic cells have more compact structures than the CG FinFET logic cells, and thus they would have more advantages in transistor count, delay, and power dissipation than CG FinFET cells, as examples shown in Table 1. All gates in the library, which are composed by low-V_{th} transistors and high-V_{th} ones with parameters extracted from TCAD simulations, have been verified by Hspice with the BSIMIMG model from UC Berkeley [21].

3. The Algorithms of Synthesis DTIG Logic Circuits

Firstly, we give some definitions about BDD and feature structure. Next, we give some theorems related to BDD subgraph extraction and prove them. Finally, we describe the implementations of BDD-based extraction algorithm and mapping algorithm.

3.1. Definitions. Binary decision diagrams (BDD) is proposed by Akers [22] as a method to representation of logic function. The methods based on BDD have been widely used in the representation and design of VLSI [23]. For the convenience of further discussion, we introduce several definitions related to BDD and its subgraph extraction firstly.

Definition 1 (binary decision diagram (BDD)). BDD has been defined in [22, 24] detailed; here we briefly describe some concepts. A BDD, as an example shown in Figure 2(a), is a rooted directed acyclic graph which is used to represent a Boolean function as

$$f = v_1 v_4 + \bar{v}_1 v_3 + \bar{v}_1 \bar{v}_4 \bar{v}_5 + \bar{v}_2 v_3 \qquad (1)$$

The BDD graph can be represented in formal language as $G = \langle V, E \rangle$, where V and E are the node set and edge set, respectively. V contains two types of nodes, named as nonterminal nodes (circular form in the figure), and terminal nodes (block form in the figure). Every nonterminal node, labelled with an input variable $v_i \in G$ ((i = 1, 2, ..., n), has two children, *low* $(v_i) \in V$ and *high* $(v_i) \in V$, and two relative edges, *else* $(v_i) \in E$ and *then* $(v_i) \in E$, connecting to the two children, respectively. Here in this paper, the *else* (v_i) and *then* (v_i) edges are denoted as dotted line and solid line in the figures, respectively. A terminal node $v_i \in V$ has not any child and outgoing edges and is only labelled with a value $(v_i) \in \{0, 1\}$.

When a BDD is used to compute a logic function $f(v_1, v_2, \ldots, v_n)$, we recursively compute the function from the root node (here assuming that it is v_i) to the terminal node 0 or 1:

$$f(v_i) = \bar{v}_i f(low(v_i)) + v_i f(high(v_i)) \qquad (2)$$

where $\bar{v}_i$ is the complementary logic of v_i. Because the production of 0 and any logic is 0, we can omit the paths from root to 0-terminal and reserve the computations from root to 1-terminal.

The BDD mentioned in this paper actually refers to reduced ordered binary decision diagram (ROBDD) [25–27], which is a type of variant BDD with input variables specially ordered and simplified structure with distinct nodes. There are many algorithms to get the ROBDD from BDD by efficiently ordering the variables and removing the redundant variables in literatures [15–19].

Definition 2 (feature structure). A feature structure is a special subgraph in a BDD, $G_s = G_i \subseteq G$ $(i = 1, 2, 3, \ldots)$,

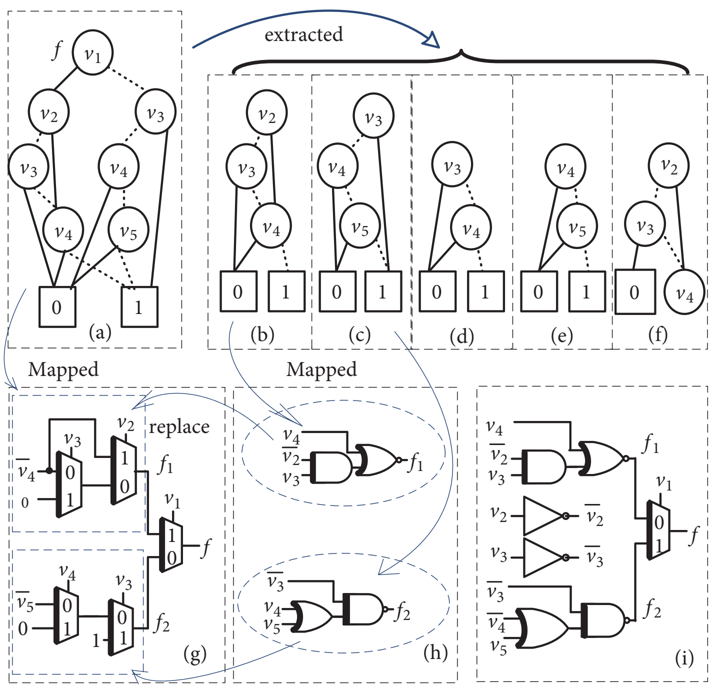

FIGURE 2: (a) A BDD example representing the Boolean logic equation (1), (b)–(f) its extracted BDD subgraphs, (g) MUX realization directly from (a), (h) the AOI and OAI cells mapped by the extracted BDD subgraphs (e), and (f), (i) the optimized result.

which can be mapped to a DTIG FinFET basic logic gate, such as AND/NAND, OR/NOR, AOI, XOR, and MUX. Here G_i is a subgraph in G rooted by v_i with its children and the descendants.

From Figure 2(a), we can extract some BDD subgraph, as shown in Figures 2(b)–2(f). Among them, subgraphs in Figures 2(c)–2(e) can be realized with the DTIG FinFET logic gates that are AOI, OAI, AND, and NOR, respectively. Therefore, all these structures are feature structures. For example, Figure 2(b) is an extracted subgraph from Figure 2(a) that can be realized by an AOI cell, and Figure 2(c) can be realized by an OAI, as shown in Figure 2(h). The structure in Figure 2(f) cannot be mapped to any logic cell; therefore it is not a feature structure.

On the other hand, for a nonterminal node in BDD, we can always realize by a MUX cell directly [22, 28]. So, we can map the BDD shown in (1) to a MUX-based netlist directly as shown in Figure 2(g).

3.2. Theorems of Extraction Algorithm

Theorem 3. *For a nonterminal node v_i of a BDD, we assume that its two children are v_x with node function $f(v_x)$ and v_y with node function $f(v_y)$. If a nonterminal node v_j with function $f(v_j)$ and one of its two outgoing edges are connecting to v_i and another edge connecting to v_x or v_y, the nodes group (v_i, v_j, v_x, v_y) can construct as a feature structure of AND/NAND/OR/NOR.*

Proof. Assuming that $v_x = high(v_i) = low(v_j)$, $v_y = low(v_i)$ and $v_i = high(v_j)$, as shown Figure 3(a).

According to Definition 1, there is a function shown in (3) and the derivations (4) and (5) from Figure 3(a).

$$\begin{aligned} f(v_j) &= \bar{v}_j f(v_x) + v_j \left(v_i f(v_x) + \bar{v}_i f(v_y)\right) \\ &= f(v_x)\left(v_i + \bar{v}_j\right) + f(v_y)\,\bar{v}_i v_j \end{aligned} \tag{3}$$

$$= f(v_x)\,\overline{\bar{v}_i v_j} + f(v_y)\,\bar{v}_i v_j \tag{4}$$

$$= f(v_x)\,\overline{\bar{v}_i v_j} + f(v_y)\,\bar{v}_i v_j \tag{5}$$

From (4) and (5), we can map the BDD subgraph to one of four feature structures of NAND, AND, OR, and NOR as shown in Figures 3(b)–3(e).

Similarly, the other cases satisfying the condition of Theorem 3 will get similar results and also can be proven easily. □

Theorem 4. *For a node group (v_i, v_x, v_y, v_j) satisfying the condition of Theorem 3, assuming that there exists a nonterminal node v_k with function $f(v_k)$ and v_x is a common child of v_j and v_i and v_y is just a child of v_i, when one outgoing edge of v_k connects to node v_j and the other edge connects to node v_x, then the nodes v_i, v_x, v_y, v_j, v_k and their relative edges construct a feature structure of three-input AND/NAND/OR/NOR logic.*

Otherwise, when one outgoing edge of v_k connects to node v_j and the other edge connects to node v_i or v_y, the nodes v_i, v_x, v_y, v_j, v_k and their relative edges construct a feature structure of three-input AOI/OAI logic.

Proof. Firstly we consider the first case as shown in Figure 4(a).

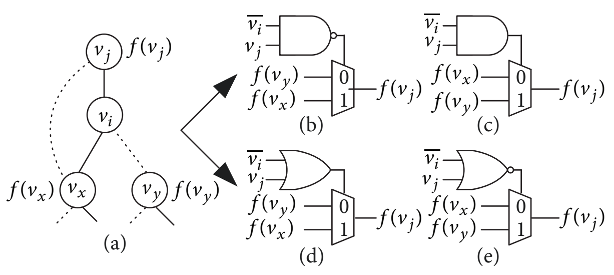
FIGURE 3: One case of Theorem 3 and relative feature structures. (a) One case of BDD subgraph, (b) NAND structure, (c) AND structure, (d) OR structure, and (e) NOR structure.

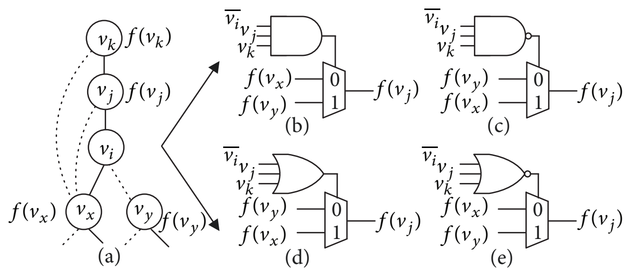
FIGURE 4: One case of Theorem 4 and relative feature structures: (a) one case of BDD subgraph, (b) AND3, (c) NAND3, (c) OR3, and (d) NOR3.

Assuming that v_x = high(v_i) = low(v_j) = low(v_k) and v_y = low(v_i), according to (2) in Definition 1, there is (6) and its derivations as follows:

$$\begin{aligned} f(v_k) &= \overline{v}_k v_x + v_k f(v_j) \\ &= \overline{v}_k f(v_x) + v_k \left(f(v_x)\overline{\overline{v}_i v_j} + f(v_y)\overline{v}_i v_j \right) \quad (6) \\ &= f(v_x)\left(\overline{v}_k + v_i + \overline{v}_j\right) + f(v_y)\left(v_k \overline{v}_i v_j\right) \\ &= f(v_x)\left(\overline{v_k \overline{v}_i v_j}\right) + f(v_y)\left(v_k \overline{v}_i v_j\right) \quad (7) \\ &= f(v_x)\left(\overline{v}_k + v_i + \overline{v}_j\right) + f(v_y)\left(\overline{\overline{v}_k + v_i + \overline{v}_j}\right) \quad (8) \end{aligned}$$

From (7) and (8), we can get the results as shown in Figures 4(b)–4(e).

Now we consider another case in Theorem 4; assuming that v_j = *high*(v_k), v_i = *low*(v_k), v_x = *high*(v_i) = *low*(v_j) = *low*(v_k) and v_y = *low*(v_i), according to (2) in Definition 1, there is (9) and its derivations as follows:

$$\begin{aligned} f(v_k) &= v_k f(v_j) + \overline{v}_k \left(v_i f(v_x) + \overline{v}_i f(v_y)\right) \\ &= v_k \left(f(v_x)\overline{\overline{v}_i v_j} + f(v_y)\overline{v}_i v_j \right) \\ &\quad + \overline{v}_k \left(v_i f(v_x) + \overline{v}_i f(v_y)\right) \quad (9) \\ &= f(v_x)\left(v_i + v_k\overline{v}_j\right) + f(v_y)\left(\overline{v}_i\left(\overline{v}_k + v_j\right)\right) \\ &= f(v_x)\left(v_i + v_k\overline{v}_j\right) + f(v_y)\left(\overline{v_i + v_k\overline{v}_j}\right) \quad (10) \\ &= f(v_x)\left(\overline{\overline{v}_i\left(\overline{v}_k + v_j\right)}\right) + f(v_y)\left(\overline{v}_i\left(\overline{v}_k + v_j\right)\right) \quad (11) \end{aligned}$$

From (10) and (11), we can get the results of Theorem 4. □

Theorem 5. *For two nodes v_i, v_j in a BDD graph, assume that v_i is the low child and the high child of v_j at the same time, if the low child v_i has two outgoing edges, then and else, connecting to v_x and v_y, respectively, while the high child v_i has two outgoing edges, then and else, connecting to v_y and v_x, respectively. The subgraph including v_j, v_i, v_x, v_y can be construct as a feature structure of XOR/XNOR logic.*

Proof. From Theorem 5, we can show it in Figure 5.

According to (2) in Definition 1, we have

$$\begin{aligned} f(v_j) &= v_j\left(v_i f(v_x) + \overline{v}_i f(v_y)\right) \\ &\quad + \overline{v}_j\left(\overline{v}_i f(v_x) + v_i f(v_y)\right) \quad (12) \\ &= f(v_x)\left(v_j v_i + \overline{v}_j \overline{v}_i\right) + f(v_y)\left(v_j \overline{v}_i + \overline{v}_j v_i\right) \\ &= f(v_x)\left(v_j \odot v_i\right) + f(v_y)\left(v_j \oplus v_i\right) \quad (13) \end{aligned}$$

From (13), the subgraph can be constructed as XOR/XNOR logic, as shown in Figures 5(b) and 5(c). □

3.3. BDD-Based Node Extraction Algorithm. According to the definitions and the theorems above mentioned, we have developed a feature structure extraction algorithm for a given BDD, which contains four subroutines to finish the extraction procedure. The subroutine 1 outputs the reduced ordering form of the BDD graph from an input logic function f. The extracted BDD subgraphs of the feature structures will be obtained through the processes from subroutines 2 to 4

```
    Algorithm of Subroutine 1:
    **Input**: $f = f(v_1, v_2 \ldots v_i, \ldots v_j \ldots v_n)$
    **Output**: G[]: node set of BDD form
1: G[] ⟵ bdd_init($f$)
2: sum_src ⟵ bdd_nodecount(G[])
3: for i = 1 to n do
4:     for j = i to n do
5:         bdd_swapvar($v_i$, $v_j$)
6:         sum_ex ⟵ bdd_nodecount(G[])
7:         if sum_ex ≤ sum_src then
8:             sum_src ⟵ sum_ex
9:         else { sum_ex > sum_src }
10:            bdd_swapvar($v_i$, $v_j$)
11:        end if
12:    end for
13: end for
```

Algorithm 1: The algorithm of subroutine 1 in the BDD-based extraction algorithm.

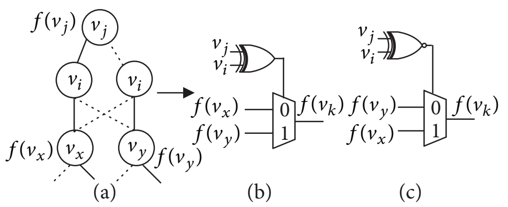

Figure 5: The case in Theorem 5: (a) BDD subgraph, (b) XOR, and (c) XNOR.

successively. In each subroutine, some parts of the BDD are marked as the different feature structures. The order of the algorithm flow is carefully arranged. As shown in Table 1, in the proposed DTIG FinFET cell library, NAND/NOR cells can reduce the number of transistors more than the single-gate gates, AOI/OAI/NAND3/NOR3 cells the second, and XOR cells the least. The extraction order of the feature structure is in this order to get the most improvement. When the whole algorithm procedure is finished, we will obtain the optimized BDD and the feature structures, which can be fed to the mapping algorithm to further process.

In the algorithm of the subroutine 1, as shown in Algorithm 1, we generate an initial reduced ordered BDD from the input logic function by using a sorting algorithm like the traditional *bubble sort* algorithm. When subroutine 1 is finished, an optimal ordered BDD form will be obtained.

The subroutine 2, as shown in Algorithm 2, searches for a father v_j of each node v_i in the optimal ordered BDD from subroutine 1.

We mark v_j as a father of v_i and v_x, v_y as two children of v_i. If the node group (v_i, v_j, v_x, v_y) meets the condition of Theorem 3, i.e., both v_i and v_x (or v_y) are the children of v_j and we extract the subgraph containing the nodes v_j, v_i, v_x, v_y and their relative edges and then mark the subgraph as a feature structure of OR/NOR (f_{OR-NOR}) or AND/NAND ($f_{AND-NAND}$). When this subroutine is finished, all feature structures of OR/NOR or AND/NAND are extracted and stored in the sets *Gso* and *Gsa*, respectively.

The algorithm of the subroutine 3, as shown in Algorithm 3, searches for a node group (v_i, v_j, v_k, v_x, v_y) which meets the condition of Theorem 4. First we check each subgraph in the result set *Gsa* from subroutine 2 and the origin BDD graph set *G* from subroutine 1; if a node v_k exists in *G* which satisfies the cases of Theorem 4, we extract a new subgraph containing the node v_k, and the corresponding subgraph in the set *Gsa* and their edges, and we mark the new subgraph as a new feature structure of AND3/NAND3 or AOI. Then, we store this feature structure into the set *Gsa3* for AND3/NAND3 or set *Gsaoi* for AOI, respectively. Finally, the corresponding subgraph in the set *Gsa* should be deleted because it has been covered by the new generated feature structure.

According to Theorem 5, algorithm of the subroutine 4, as shown in Algorithm 4, searches for the special groups of nodes in the optimal ordered BDD from subroutine 1 and then constructs them as new feature structures. If the nodes of a group satisfy the conditions of Theorem 5, (a) a node (denoted as v_k) is the same father of the other two nodes (denoted as v_i and v_j) of the same variable, (b) there exist two nodes that are the same children of v_i and v_j, and (c) $low(v_j) = high(v_i)$ and $low(v_i) = high(v_j)$, then we extract the group and their edges as a feature structure of XOR logic and then store it into set *Gsxor*.

When these subroutines are all finished, the proposed extraction algorithm generates the optimal ordered BDD form of a given logic function and obtains all the feature structures in the BDD.

```
Algorithm of subroutine2:
Input: G[]: node set from subroutine 1
Output: Gsa[]: set of AND/NAND2 sub-graph
Output: Gso[]: set of OR/NOR sub-graph
1:  for each $v_i$ in G[] do
2:   $(v_x, v_y) \longleftarrow$ children$(v_i)$
3:   for each $v_j$ in G[] do
4:     if low$(v_j) = v_i$ then
5:       If high(vj) = vx OR high(vj) = vy then
6:       $f_{\text{OR-NOR}} \longleftarrow f_{\text{OR-NOR}}$mark$(v_i, v_i, v_x, v_y, E(v_j, v_i, v_x, v_y))$
7:       Gso[k] $\longleftarrow f_{\text{OR-NOR}}$
8:       end if
9:     else if high$(v_j) = v_i$ then
10:     if low$(v_j) = v_x$ OR low$(v_j) = v_y$ then
11:        $f_{\text{AND-NAND}} \longleftarrow$ mark$(v_i, v_i, v_x, v_y, E(v_j, v_i, v_x, v_y))$
12:        Gsa[] $\longleftarrow f_{\text{AND-NAND}}$
13:      end if
14:    end if
15:  end for
16: end for
```

Algorithm 2: The algorithm of subroutine 2 in the BDD-based extraction algorithm.

```
Algorithm of Subroutine3:
Input: Gsa: set of AND/NAND2 sub-graph from subroutine2;
Input: G[]: node set of BDD from subroutine 1
Output: Gsaoi[]: set of AOI/OAI sub-graph
Output: Gsand3[]: set of AND/NAND31 sub-graph
1: for each gs in Gsa do
2:    $v_j \longleftarrow$root(gs)
3:    for each $v_k$ in BDD do
4:      if low$(v_k)$=$v_j$ OR high$(v_k)$=$v_j$ then
5:        if $v_k$ satisfy Theorem 4 case 1 then
6:          $f_{\text{ANDNAND31}} \longleftarrow$ mark$(v_i, v_j, v_k, v_x, v_y, E(v_i, v_j, v_k, v_x, v_y))$
10:         Gsa3[] $\longleftarrow f_{\text{ANDNAND31}}$
11:       else if $v_k$ satify Theorem 4 case2 then
12:         $f_{\text{AOIOAI}} \longleftarrow$ mark$(v_i, v_j, v_k, v_x, v_y, E(v_i, v_j, v_k, v_x, v_y))$
13:         Gsaoi[] $\longleftarrow f_{\text{AOIOAI}}$
14:       end if
15:     end if
16:   end for
17: end for
```

Algorithm 3: The algorithm of subroutine 3 in the BDD-based extraction algorithm.

3.4. The Mapping Algorithm. The extraction algorithms described above only aims at simplifying Boolean functions. In the next logic synthesis steps, we need to replace the logic gates with physical cells in the cell library by using a mapping algorithm. We propose the feature structure mapping algorithm with four steps. In step 1, after reading the BDD and its subgraphs from the extraction algorithm, we exclude the redundant, or covered, subgraphs. Then, in step 2, we map the source BDD to a circuit composed of MUXs completely and map the feature structures to the IG FinFET logic gates. In step 3, we give the final optimized circuit by replacing some MUXs subcircuits with logic cells. Step 1 to step 3 are shown as examples in Figures 2(a)–2(h) and the result is as shown in Figure 2(i).

4. Algorithm Implementation

The synthesis algorithms including extraction and mapping are both implemented in MATLAB platform, and for comparison of the circuit optimization effectiveness, the ABC and DC synthesis tools are also applied to the same circuits. For a fair comparison, all methods use the same DTIG FinFET cell

```
    Algorithm of Subroutine 4
1:  **Input**: G[]: node set of source BDD from subroutine 1
2:  **Output**: Gsxor[]: set of XOR/NXOR sub-graph
3:  for each $v_k$ in G[] do
4:      for each $v_i$ in G[] do
5:          for each $v_j$ in G[] do
6:              if low($v_k$) = $v_i$ AND high($v_k$) = $v_j$ OR
                  high($v_k$) = $v_i$ AND low($v_k$) = $v_j$ then
7:                  if value($v_j$) = value($v_i$) then
8:                      if low($v_i$) = high($v_j$) AND high($v_i$) = low($v_j$) then
9:                          ($v_x$,$v_y$) ⟵ children($v_i$,$v_j$)
10:                         $f_{XORNXOR}$ ⟵ mark($v_i, v_j, v_k, v_x, v_y$, E($v_i, v_j, v_k, v_x, v_y$))
11:                         Gsxor[]⟵$f_{XORNXOR}$
12:                     end if
13:                 end if
14:             end if
15:         end for
16:     end for
17: end for
```

ALGORITHM 4: The algorithm of subroutine 4 in the BDD-based extraction algorithm.

library we built in Section 2. Finally, all the circuits from ABC, DC, and the proposed method are simulated by using Hspice with the BSIMIMG model from UC Berkeley [21]. Figure 6 presents the simulation results of the MCNC benchmark circuits.

As shown in Figure 6(a), for almost all tested circuits, the count of transistors in the circuit optimized by this work is the least among the comparisons. Therefore, in most cases, the proposed method can get the most effective area since the area occupation is determined by the number of the transistors.

The average power dissipation can be expressed by

$$P_{av} = NP_{av1} = N\left(p_{low}P_{low} + p_{high}P_{high}\right) \qquad (14)$$

where P_{av} is the average power of a circuit, N is the transistor count in a circuit, P_{avl} is the average power of one transistor, P_{low} and P_{high} are the average power of one low-V_{th} and high-V_{th} IG FinFET, respectively, and p_{low} and p_{high} are the possibility of low-V_{th} and high-V_{th} IG FinFET in a circuit, respectively.

From (14), if the possibility of transistors in kinds of circuits is close to each other, the average power of a circuit is determined by the count of transistor in a circuit. From Figure 6(b), we can see the trend of the power dissipation is close to the transistor count trend shown in Figure 6 and still almost all the circuits synthesized by this work have the least power dissipation, and the fact fits the prediction of (14) very well. The delay analysis is more complicated than power dissipation. The maximum delay of a circuit depends on the critical path. The more transistors on the critical path are, the greater delay should be. For a DTIG FinFET circuit, the high-threshold device has more delay because it has low on-current, which further reduces the switch speed and increases the delay. So as shown in Figure 6(c), we find that the circuit synthesized by this work has no obvious advantage in terms of delay.

The power delay product (PDP) can evaluate a circuit more comprehensively because it considers both delay and power consumption. As seen from Figure 6(d), compared with ABC and DC, all of the circuits synthesized by this work have obvious advantage of PDP.

5. Conclusion

In this paper, we have presented a BDD-based synthesis method to optimize DTIG FinFET circuits. We search the BDD graph of an input logic to find feature structures and map them to DTIG FinFET basic logic gates. The algorithms are implemented in MATLAB and compared with ABC and DC by simulation of the MCNC benchmark circuits. The result shows that the proposed method can significantly improve the performances of DTIG FinFET circuits in area occupation, power consumption, and PDP.

Data Availability

All data used to support the study had been included in the paper and can be accessed freely.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research is based upon work supported by National Natural Science Foundation of China under Grant No.

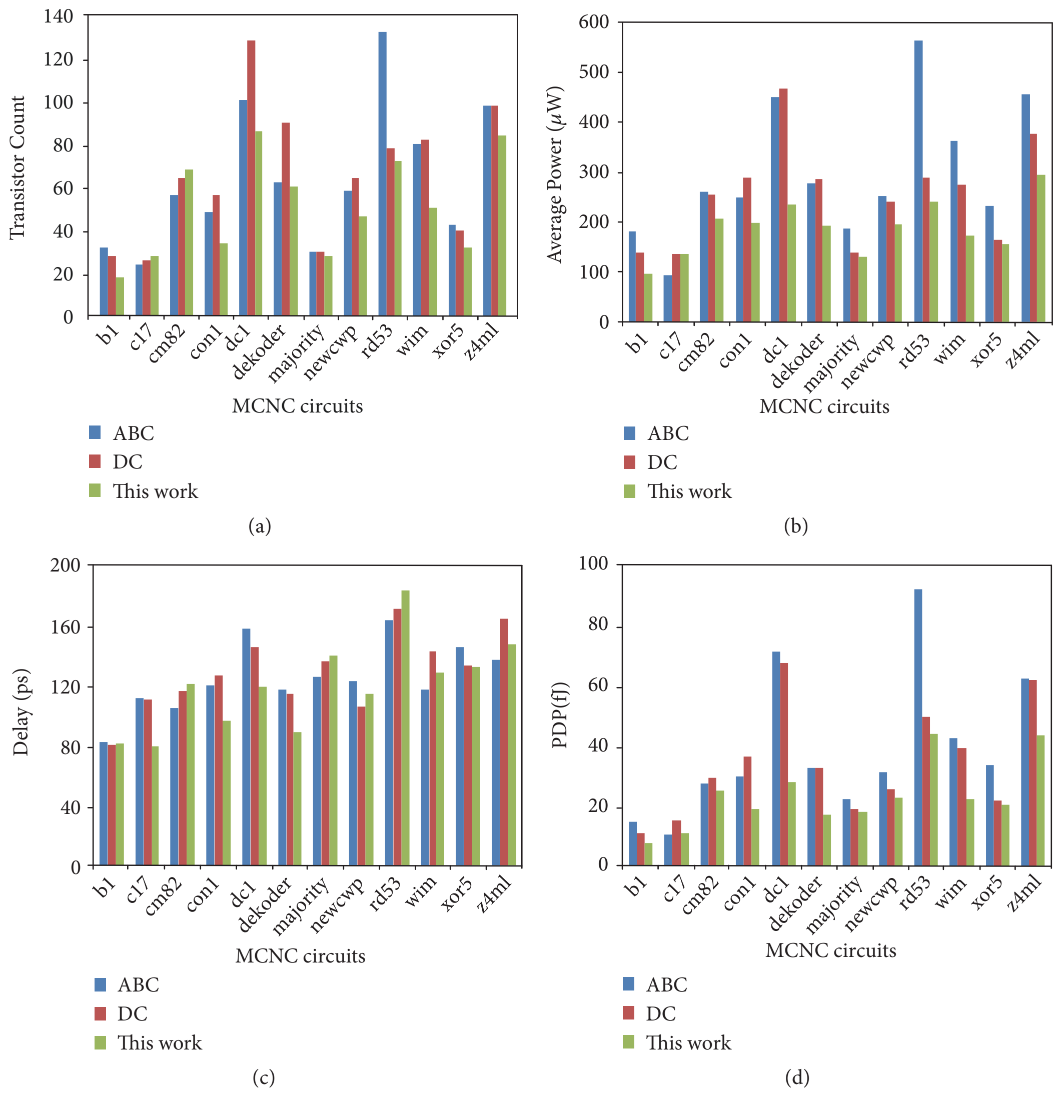

Figure 6: Comparison between the simulation results of ABC, DC, and this work.

61671259 and Zhejiang Provincial Natural Science Foundation (No. LY19F010005). The study is funded by National Natural Science Foundation of China [61671259] and Zhejiang Provincial Natural Science Foundation [LY19F010005].

References

[1] C. Shin, "State-of-the-art silicon device miniaturization technology and its challenges," *IEICE Electronics Express*, vol. 11, no. 10, Article ID 20142005, 2014.

[2] P. Mishra, A. Muttreja, and N. K. Jha, "FinFETcircuit design," *Nanoelectronic Circuit Design*, pp. 23–54, 2011.

[3] M. Rostami and K. Mohanram, "Dual-Vth independent-gate FinFETs for low power logic circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 3, pp. 337–349, 2011.

[4] S. Chaudhuri and N. K. Jha, "FinFET logic circuit optimization with different FinFET styles: Lower power possible at higher supply voltage," in *Proceedings of the International Conference on Vlsi Design and 2014 International Conference on Embedded Systems*, pp. 476–482, 2014.

[5] S. Tawfik and V. Kursun, "Multi-Threshold Voltage FinFET Sequential Circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 1, pp. 151–156, 2011.

[6] M. Imani, S. Patil, and T. S. Rosing, "Hierarchical design of robust and low data dependent FinFET based SRAM array," in *Proceedings of the IEEE/ACM International Symposium on Nanoscale Architectures*, pp. 63–68, 2015.

[7] T. Wang, X. Cui, K. Liao et al., "Low power high performance FinFET standard cells based on mixed back biasing technology," *IEICE Transactions on Electronics*, vol. E99.C, no. 8, pp. 974–983, 2016.

[8] R. Brayton and A. Mishchenko, "ABC: An academic industrial-strength verification tool," in *Proceedings of the Computer Aided Verification, International Conference*, vol. 6174, pp. 24–40, Edinburgh, Uk, CAV 2010.

[9] X. Zhang, J. Hu, and X. Luo, "Optimization of dual-threshold independent-gate FinFETs for compact low power logic circuits," in *Proceedings of the 16th IEEE International Conference*

on Nanotechnology - IEEE NANO 2016, pp. 529–532, Sendai, Japan, August 2016.

[10] H. Yang, J. Hu, and H. Zhu, "Novel SRAM cells using dual-threshold independent-gate FinFETs," in *Proceedings of the 17th IEEE International Conference on Nanotechnology, NANO 2017*, pp. 358-359, Pittsburgh, Pa, USA, July 2017.

[11] H. Ni, J. Hu, H. Yang, and H. Zhu, "Comprehensive optimization of dual threshold independent-gate FinFET and SRAM cells," *Active and Passive Electronic Components*, vol. 2018, Article ID 4512924, 10 pages, 2018.

[12] W. Lenders and C. Baier, "Genetic algorithms for the variable ordering problem of binary decision diagrams," in *Proceedings of the International Conference on Foundations of Genetic Algorithms*, pp. 1–20, 2005.

[13] U. S. Costa, A. M. Moreira, D. Dharbe, C. Universitrio, and L. Nova, "Advances in bdd reduction using parallel genetic algorithms," in *Proceedings of the IEEE 10th International Workshop on Logic & Synthesis (IWLS2001)*, pp. 84–89, 2001.

[14] P. Porwik, K. Wrobel, and P. Zaczkowski, "Some practical remarks about Binary Decision Diagram size reduction," *IEICE Electronics Express*, vol. 3, no. 3, pp. 51–57, 2006.

[15] S. Tani, K. Hamaguchi, and S. Yajima, "The complexity of the optimal variable ordering problems of shared binary decision diagrams," in *Proceedings of the in Algorithms & Computation, International Symposium, ISAAC 93*, Hong Kong, December 1993.

[16] B. Bollig and I. Wegener, "Improving the variable ordering of OBDDs is NP-complete," *IEEE Transactions on Computers*, vol. 45, no. 9, pp. 993–1002.

[17] H. Fujii, G. Ootomo, and C. Hori, "Interleaving based variable ordering methods for ordered binary decision diagrams," in *Proceedings of the IEEE/ACM International Conference on Computer-aided Design*, 1993.

[18] O. Brudaru, C. Rotaru, and I. Furdu, "Static segregative genetic algorithm for optimizing variable ordering of ROBDDs," in *Proceedings of the International Symposium on Symbolic & Numeric Algorithms for Scientific Computing*, 2011.

[19] M. Takapoo and M. B. Ghaznavi-Ghoushchi, "IDGBDD: The novel use of ID3 to improve genetic algorithm in BDD reordering," in *Proceedings of the International Conference on Electrical Engineering/electronics Computer Telecommunications & Information Technology*, 2010.

[20] H. Zhu, J. Hu, H. Yang, Y. Xiong, and T. Yang, "A topology optimization method for low-power logic circuits with dual-threshold independent-gate FinFETs," in *Proceedings of the 27th International Symposium on Power and Timing Modeling, Optimization and Simulation, PATMOS 2017*, pp. 1–6, Greece, September 2017.

[21] N. Paydavosi, S. Venugopalan, Y. S. Chauhan et al., "BSIM—SPICE models enable FinFET and UTB IC designs," *IEEE Access*, vol. 1, pp. 201–215, 2013.

[22] S. B. Akers, "Binary decision diagrams," *IEEE Transactions on Computers*, vol. 27, no. 6, pp. 509–516, 1978.

[23] S. ichi Minato, "Techniques of BDD/ZDD: brief history and recent activity," *IEICE Transaction on Information and Systems*, vol. E96.D, no. 7, pp. 1419–1429, 2013.

[24] R. Drechsler and D. Sieling, "Binary decision diagrams in theory and practice," *International Journal on Software Tools for Technology Transfer*, vol. 3, no. 2, pp. 112–136, 2001.

[25] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Transactions on Computers*, vol. C-35, no. 8, pp. 677–691, 1986.

[26] M. Raseen, P. Chandana Prasad, and A. Assi, "An efficient estimation of the ROBDD's complexity," *Integration, the VLSI Journal*, vol. 39, no. 3, pp. 211–228, 2006.

[27] K. S. Brace, R. L. Rudell, and R. E. Bryant, "Efficient implementation of a BDD package," in *Proceedings of the 27th ACM/IEEE Design Automation Conference*, pp. 40–45, June 1990.

[28] J. M. Rabaey, A. P. Chandrakasan, and B. Nikolic, *Digital Integrated Circuit-A Design Perspective*, Prentice-Hall, Inc, 2003.