Synthesis of Reversible and Quantum Circuit using ROCBDD and Mixed-Polarity Toffoli Gate

Hieu Nguyen¹, Linh Tran²
¹Department of Electronics, Ho Chi Minh University of Technology, Vietnam
²Department of Electronics, Ho Chi Minh University of Technology, Vietnam
Corresponding author: Linh Tran (e-mail: linhtran@hcmut.edu.vn).

ABSTRACT In the last decade, the synthesis of reversible logic circuits has become a trending topic because of its future necessity and importance. Many methods have been studied and proposed, for instance, transformation-based, search-based, cycle-based, ESOP-based and BDD-based methods. Each of them has its limitation related to time processing, ancilla and garbage line, quantum cost.

This study develops an algorithm that could synthesize quantum circuits based on mixed-polarity Toffoli gate and a variant of binary decision diagram (BDD) called reduced-ordered-complemented edge-bdd (ROCBDD). It is an optimized method of BDD to reduce nodes in the representation of Boolean functions, which leads to adding more lines and quantum gates in the reversible circuit. First, the differences between synthesis using ROCBDD, the traditional BDD and others methods are introduced. Then, we define a new structure consisting of mixed-polarity Toffoli gates relied on nodes of ROCBDD. An efficient algorithm to match BDD representation to the reversible logic circuit is also mentioned. Finally, the experimental results show that our algorithm has better synthesizing costs than previous BDD-based methods.

INDEX TERMS Mixed-Polarity Toffoli Gate, ROCBDD, reversible circuit, synthesis

I. INTRODUCTION

Many studies about the application of quantum logic circuits [1]–[3] have been introduced and proved ever before. Two main problems solved by quantum logic circuits are reducing circuit power consumption and increasing the density of transistors in an area of layout circuit. In [1], [4], the authors show that the power consumption of a calculation using qubit can be less than KTln2 – which is the least power consumption of the same calculation using a traditional bit. In the definition of the quantum equation, a qubit can express many states simultaneously, which leads to calculations being conducted simultaneously. It solves processing time and resources problems. Its application in many domains is also introduced, like DNA computing [1], optical computing [5], nanotechnology [6] and quantum computing [7]. However, almost all algorithms used to synthesize irreversible traditional logic cannot be carried on to synthesis reversible logic due to two issues: fan-out and feedback. Only cascade structure is accepted in reversible circuits. To solve the earlier problems, many techniques for synthesizing quantum circuit has been researched and developed over the past two decades.

In [8], [9], Maslov et al. proposed a transformation-based method that transforms outputs sequentially and relies on the properties of the pre-selected quantum gate (Toffoli, Fredkin, …) to choose the path from output to input and synthesis the circuit. This first method depends on the size of the truth table, which leads to a considerable processing time when the numbers of inputs are increased. The following method introduced is search-based (heuristic methods), [10]–[12] which iteratively finding the possible path selection using Hamming distance [13]. After this phase, a variety of reversible gates are selected by finding the possible matching reversible gate. Similar to the transformation-based method, this method relies on the size of the truth table, but the results are better due to using Hamming distance to find the best path. In [14], [15], Saeedi et al. introduced a technique called the cycle-based method to decompose Boolean functions into smaller cycles. From each cycle, this method synthesizes it into a quantum circuit. The result of this method depends on the number of cycles and the decomposition process. The ESOP-based method proposed in [16]–[20] was the synthesis algorithm with no adding lines. By using Positive-polarity Reed-Muller expansion [21], this method synthesis quantum
circuits by matching each selection-part in expansion to a built-in template in the library. Its disadvantage is the processing time to build the library when the numbers of input grow up.

Our research starts by using the BDD-based method – which was first introduced by Wille in [22]. The first step of this method is building a BDD [23] for Boolean functions. An advantage of conduct BDD is the capability of large function expression infinite time which overcomes the weakness of previous methods. Then, each node of BDD is matched to a cascade of reversible gates or templates. The additional templates may add more garbage lines to the circuit result due to the properties of the shared nodes of BDD. Many BDD versions are introduced to optimize the algorithm. Three structures called shared BDD, complement edges BDD and advanced ordering BDD were used in synthesis and evaluation individually in [24]. Shared edges BDD reduces lines and quantum costs by sharing nodes for many functions. The complement-edge version decreases the total sizes of BDD (measured by the numbers of nodes) in half, causing a reduction in additional lines. Besides, by using complement edges, the templates are more complicated with only Toffoli gate expression, which lead to higher quantum cost. Meanwhile, advanced ordering BDD required many loops to choose the correct orders of variables to get the simplest structure of BDD. Taking all the advantages of the two last versions, ROCBDD is used to synthesize the quantum circuit. Moreover, we also study an algorithm to match templates at share nodes – an important property of quantum gates. In general, a quantum gate expresses a function $g : B^a \rightarrow B^b$. Let denote $C = \{c_1, c_2, \ldots, c_a\} \subseteq X$ and $D = \{d_1, d_2, \ldots, d_b\} \subseteq X$ with $C \cap D = \emptyset$, in which $C$ is called the set of control lines and $D$ is called the set of target lines.

This research focuses on the Toffoli gate and mixed-polarity Toffoli gate with three inputs and three outputs. From the previous definition:

- 1 input and 1 output: $\{x_1\}$ to $\{1 \oplus x_1\}$. NOT gate (Fig. 1a).
- 2 inputs and 2 outputs: $\{x_1, x_2\}$ to $\{x_1, x_2 \oplus x_1\}$. CNOT gate (Fig. 1b).
- 3 inputs and 3 outputs: $\{x_1, x_2, x_3\}$ to $\{x_1, x_2, x_1 \oplus x_2\}$. Toffoli gate (Fig. 1c).

Mixed-polarity Toffoli gate with 2 or 3 inputs and outputs are defined similarity:

- 2 inputs and 2 outputs: $\{x_1, x_2\}$ to $\{x_1, x_2 \oplus \overline{x_1}\}$. Negative CNOT gate (Fig. 1d).
- 3 inputs and 3 outputs: $\{x_1, x_2, x_3\}$ to $\{x_1, x_2, x_1 \oplus \overline{x_2}\}$. Semi-negative Toffoli gate (Fig. 1e).
- 3 inputs and 3 outputs: $\{x_1, x_2, x_3\}$ to $\{x_1, x_2, x_3 \oplus \overline{x_1}x_2\}$. Negative Toffoli gate (Fig. 1f).

B. QUANTUM LOGIC GATE

To implement a reversible logic function to a quantum circuit, quantum logic gates are used. Differentiating from classic gates, quantum gates have the numbers of inputs equal to the numbers of outputs. These properties allow us to realize quantum circuits by connecting cascade quantum gates. A multiple output Boolean function is mapping $f : B^a \rightarrow B^m$. Let denote $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{f_1, f_2, \ldots, f_m\}$ are the input set and output set respectively. $f$ is called reversible if:

- The number of input is equal to the number of output which means $n = m$.
- The mapping from inputs to output is bijective or any input only maps to a unique output.

Using this definition, a multiple output function having $n$ inputs and $m < n$ outputs can become reversible when adding $(n-m)$ value to output called ancilla lines.

II. PREMILNARIES

A. REVERSIBLE LOGIC FUNCTION

A multiple output Boolean function is mapping $f : B^a \rightarrow B^m$. Let denote $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{f_1, f_2, \ldots, f_m\}$ are the input set and output set respectively. $f$ is called reversible if:
To evaluate each gate, we use quantum cost defined in [8]. The quantum cost for 6 gates is shown in Table 1.

| Figure   | Quantum cost |
|----------|--------------|
| 1a       | 1            |
| 1b       | 1            |
| 1c       | 5            |
| 1d       | 3            |
| 1e       | 5            |
| 1f       | 7            |

TABLE 1 QUANTUM COST FOR MIXED-POLARITY TOFFOLI GATES.

C. FROM BINARY DECISION DIAGRAM – BDD TO ROCBDD

Binary Decision Diagram (BDD) is a graph used to represent a Boolean function. Assume that we have Boolean function \( f \{x_1, x_2, \ldots, x_n \} \).

Let denote: \( f_{x_i} = \{x_1, x_2, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n \} \) and \( f_{x_i} = \{x_1, x_2, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n \} \). Using Shannon decomposition [26], we have:

\[
f = x_i f_{x_i} + \overline{x_i} f_{x_i} \quad (1 \leq i \leq n)
\]

Each node in BDD represents a Shannon expression with a variable. For each \( x_i \), \( f_{x_i} \) and \( f_{\overline{x_i}} \) are called the high node and low node respectively. By using Shannon on the \( f_{x_i} \) and \( f_{\overline{x_i}} \) with another variable, these nodes also have their high and low node (Fig. 2a). In this diagram, the solid line is called positive edge and represented for \( x_i = 1 \) case.

Dot line called negative edge and represented for \( x_i = 0 \) case.

Since \( f_{x_i} \) and \( f_{\overline{x_i}} \) in reversible logic function satisfy \( f_{x_i} \neq f_{\overline{x_i}} \), so another form of Shannon decomposition EXOR form [26]:

\[
f = x_i f_{x_i} \oplus \overline{x_i} f_{\overline{x_i}} \quad (1 \leq i \leq n)
\]

With this form, a function \( f \) can be represented by a quantum gate like the Toffoli gate (Fig. 2b). In Figure 2, the circuit is reconstructed by mapping each node of BDD and adding four lines. When synthesizing high node \( x_1 \) and low node \( x_1 \), two lines are added to the quantum circuit for each node. However, none of the lines are used to synthesize node \( f \). Two previous cases illustrate two types of templates using in matching. The templates for matching and BDD sizes need to be concerned to reduce lines and numbers of gates. Using complement edges is one of the best ways to reduce the size of the BDD structure. This method is based on the transformation of Shannon expansion:

\[
f = x_i f_{x_i} + \overline{x_i} f_{\overline{x_i}} \quad (1)
\]

Instead of using the original form, this expression gives a new way by adding a complement edge definition. In our paper, when illustrating this edge on BDD, a line with -1 is used. The node to which this edge pointing is represented for a complimented function. Using this notation, both positive edge (solid line) and negative edge (dot line) of
traditional BDD structure can become complement edge. As mentioned above, this edge is used to describe expression (1), so in the final diagram, only dot lines with -1 are used. Positive edge is still complemented when applying transformation rules but does not appear in the final diagram.

The example in Fig. 2a has a complement edge, as illustrated in Fig. 3a. By comparing two diagrams in two figures, we could see the number of nodes reduce. In theory, this diagram combines two complement nodes into one node so the total nodes can decrease up to twice. Two important reduction rules used for ROCBDD are described in Fig. 3b. Another example using this form is in Fig. 4. In this figure, both complement edges and negative edges are shown. This property leads us to an idea to use a mixed-polarity Toffoli gate in a matching circuit.

![BDD with complement edges](image)

**FIGURE 3.** BDD with complement edges.

![Example of BDD with complement edges](image)

**FIGURE 4.** Example of BDD with complement edges.

### III. ALGORITHM

#### A. TEMPLATE FOR MATCHING USING MIXED-POLARITY TOFFOLI GATE

As discussed in Section 2, using ROCBDD helps decrease the number of nodes in the diagram, reducing ancilla lines and gates. We study and apply templates with mixed-Toffoli gates to help reduce quantum costs for each template (Table 2).

For structure number 1, using Shannon expression and positive Davio expansion [26]:

\[
 f = x_i f_{x_i} \oplus \overline{x_i} f_{\overline{x_i}} \quad (2)
\]

\[
 = f_{\overline{x_i}} \oplus x_i \left( f_{x_i} \oplus f_{\overline{x_i}} \right) \quad (3)
\]

Using expression (2) leads to the template at column 3 - Circuit adding line and the template in column 4 – Circuit without line are represented for (3). Setting \( f_{x_i} = 1 \) and \( f_{\overline{x_i}} = 1 \), respectively, we calculate two templates (Number 2 and 3) in Table 2.

For structure number 4, using Shannon expression and negative Davio expansion [26]:

\[
 f = x_i f_{x_i} \oplus \overline{x_i} f_{\overline{x_i}} \quad (4)
\]

\[
 = f_{\overline{x_i}} \oplus \overline{x_i} \left( f_{x_i} \oplus f_{\overline{x_i}} \right) \quad (5)
\]

Similarly, using the function (4) and (5), we study the other templates. Especially with structure number 5, we do not have the template for column 4. To explain it, the function for this diagram is: \( f = x_i f_{x_i} \oplus \overline{0} \). However, there are many types of gate structures illustrating for a function, we choose the structure with the least gate cost and quantum cost. The quantum cost for each structure is shown in Table 3.

**TABLE 2**

| No | BDD structure | Circuit adding line | Circuit without adding line |
|----|---------------|---------------------|----------------------------|
| 1  |               | high - high         | high - high                |
|    |               | low - low           | low - low                  |
|    |               | x - x               | x - x                      |
| 2  |               | high - high         | high - high                |
|    |               | low - low           | low - low                  |
|    |               | x - x               | x - x                      |

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
In Table 2, with the template in column 3, an ancilla line is added to make the circuit reversible. Meanwhile, the template in column 4 is not reversible, which means when using this type of template, the following circuit cannot use line high and line low become f. In the rest circuit, if there is no appearance of the line high and low, we can apply the template in column 4. If not, the templates in column 3 must be used and therefore, one ancilla has to be added to the circuit.

| No | Gate Cost | Quantum Cost |
|----|-----------|--------------|
|    | Circuit adding line | Circuit without adding line | Circuit adding | Circuit without adding line |
| 1  | 2          | 2            | 10            | 6               |
| 2  | 1          | 1            | 5             | 3               |
| 3  | 2          | 1            | 6             | 1               |
| 4  | 2          | 2            | 10            | 8               |
| 5  | 1          | -            | 5             | -               |
| 6  | 1          | 2            | 5             | 2               |

Using mixed-polarity Toffoli gates, a function with a complemented variable is represented by a semi-negative Toffoli gate (Fig. 1e) with a quantum cost of 5. If only used Toffoli gate, the circuit for this function has a quantum cost of 6 and is represented by 2 gates (Fig. 5).

**FIGURE 5.** Two templates represent for a function.

In the ROCBDD, there are two special cases shown in Table 4. Case 1 always appears in this diagram, represented the last order variable. The circuit in this case is shown in column 4. The algorithm sees this node like a variable, unlike a function. Case 2 rarely appears in diagrams. In our algorithm, when we meet this structure, it prioritizes matching at column 4.

**TABLE 4**

| No | BDD structure | Circuit adding line | Circuit without adding line |
|----|---------------|---------------------|-----------------------------|
| 1  |               |                     |                             |
| 2  |               |                     |                             |
5: If \( n = 1 \) then matching template of column 4.
6: Else:
7: If node(i) has special cases then
8: Prioritizing to match template in column 4.
9: Template_without_adding_line := 1
10: Minus n.
11: For n downto 2:
12: If line point to node(i) is positive then
13: Prioritizing matching template in column 3.
14: Minus n.
15: If Template_without_adding_line == 1 then
16: Matching template of column 3.
17: Else:
18: Matching template of column 4.

For our algorithm, the first step is to create BDD with appropriate order and then reduce it with complement reduce rules to make BDD become ROCBDD. After constructing the diagram, one node may be pointed by many edges. It means the function represented by this node is shared by many other functions, as shown in Fig. 6. In this case, choosing a template in column 3 or 4 is necessary. Assume that node(i) is pointed by n node or n function using node(i), then our algorithm is using (n-1) template in column 3 and the other’s template in column 4.

Our algorithm is illustrated in Fig. 7.

- Starting with node b–12, it is used 4 times, which is indicated by 4 lines pointing to the node. In this case, node a-52 connects to b-12 in special case number 2, synthesizing node-52 by using the template in column 4 (denote N in Fig 6). The template is added to the quantum circuit (denote N). After that, there are still 2 lines pointing to b-12, one is node a-13 and the other is a-28. Synthesize node a-28 is prioritized because positive edge points to b-12. It is matching by using the template in column 3 (denote C). In three nodes connecting to node b-12, one node is matched using the template in column 4. The last node a-13 is matched using the template in column 3 (denote 3).

- Continue to check node a-13. There are two lines pointing to this node. Node c2-59 with a solid line pointing to a-13 is synthesized by using the template in column 3 (denote C). Node c2-14 is matched by using the template in column 4 (denote N). Similar to nodes a-52 and a-28.

- At node c2-59, only one line points to from node c1-95. It is matched by using the template in column 4. Similar to node c2-94, only one line points to from node c1-95. Because we synthesized at node c1-95, ignore this time.

- Continue to synthesize the rest of the diagram. The result for this example is shown in Fig. 8.

C. EVALUATE THE COMPLEXITY AND GATE COST

The complexity of the proposed algorithm is calculated based on the worst case of the size of BDD. Assume that a single output function has \( n \) variables. The largest size of BDD is \( 2^n \) nodes [22].

- When counting the number of uses at each node, the algorithm traverses all nodes of BDD. It leading the complexity of this step is \( O\left(2^n\right) \).

- The following step is matching each line pointing to a node with a template. Assume that \( k_1, k_2, \ldots, k_m \) is the number of lines pointing to each node of the BDD.
Denote $k = \max\left(k_1, k_2, \ldots, k_{2^n}\right)$. The complexity of matching at each node is $O(k_1), O(k_2), \ldots, O(k_{2^n})$. Because this step requires matching all nodes, the complexity is the sum of all the complexity at each node. It means the complexity is $O(k_1) + O(k_2) + \ldots + O(k_{2^n}) = O(k)$. Thus, the total complexity of the proposed method is $O(2^n) + O(k) = O(2^n)$ because $k$ is always less than $n$.

Similarly, a reversible function with $n$ variables has $n2^n$ in the worst case. It leads to complexity in this case as $O(n2^n)$.

The results of synthesis circuits are bounded by the size of the BDD. Since many research studied the bounds of gate cost and adding lines in theory [22], [24], [27]. By using the proposed theoretical results combining with our algorithm, the upper bounds of gate cost and adding lines when implementing by our algorithm can be proved:

- A BDD representing a single-output function with $n$ variables has $2^n$ nodes in the worst case [22]. Using ROCBDD, our algorithm can, at best, reduce total nodes number in half. Thus, the worst number of nodes never reaches $2^n$. In the worst case, all nodes are matched to number 1 or 4 of table 1. So our algorithm has upper bound $2x2^n$ gates.

- A BDD representing a reversible function with $n$ variables has $n2^n$ nodes in the worst case when each BDD of single-output function is built individually. Thus, with our algorithm the number of nodes never reaches $n2^n$. In the worst case, each reversible function can be realized in a reversible logic function with less than $n.2^{n+1}$ gates.

- The number of lines adding to the circuit has maximum depends on the number of nodes. Using our algorithm, the numbers of lines adding to the circuit are always less than $2^n$ with a single-output case and $n.2^{n+1}$ with a reversible function.

IV. THE RESULTS AND DISCUSSION

This section shows our evaluation of the experimental results of our algorithm. Our proposed algorithm with new templates is implemented using Python on top of the BDD package CUDD [28]. Our benchmarks functions are provided by RevLib [29]. In this test, we only proposed the results of functions used in the previous study [22], [24]. The parameters for these tests are the number of lines in the synthesis circuit, gate cost (GC) and quantum cost (QC). The numbers of input and output are denoted PI and PO, respectively. All experiments have been carried out on a personal laptop with an Intel Core i7 processor, 1.8GHz and 8GB RAM.

First, we compare the results of our algorithm with the results of each diagram (which are shared node BDD, complement edges BDD and variable ordering BDD) in [24], as shown in Table 5.

| Func. | PI/ PO | Shared Nodes | Comp. Edges | Var. Order | Our Alg. |
|-------|--------|--------------|-------------|------------|----------|
|        | L | GC | L | GC | L | GC |
| 3-17 6 | 3/3 | 10 | 20 | 8 | 17 | 7 | 12 |
| 4-49 7 | 4/4 | 18 | 45 | 16 | 45 | 15 | 42 |
| 4mod5 8 | 4/1 | 9 | 13 | 8 | 16 | 7 | 8 |
| alj-11 81 | 4/4 | 19 | 45 | 16 | 43 | 16 | 42 |
| alu 9 | 5/1 | 14 | 29 | 11 | 25 | 7 | 9 |
| decoder- enable 32 | 3/4 | 9 | 9 | 14 | 9 | 14 | 6 |
| decoder-4 10 | 2/4 | 7 | 7 | 6 | 11 | 6 | 11 |
| ex-1 82 | 3/3 | 8 | 13 | 7 | 14 | 5 | 7 |
| fredkin_3 | 3/3 | 5 | 6 | 5 | 6 | 5 | 6 |
| graycode6_11 | 6/6 | 16 | 20 | 11 | 15 | 11 | 15 |
| hwb4 28 | 3/3 | 18 | 18 | 5 | 12 | 5 | 10 |
| hwb7 29 | 7/7 | 36 | 88 | 18 | 50 | 21 | 61 |
| hwb5 15 | 5/5 | 32 | 91 | 27 | 85 | 28 | 88 |
| hwb6 16 | 6/6 | 53 | 167 | 46 | 157 | 46 | 159 |
| hwb7 15 | 7/7 | 84 | 284 | 74 | 276 | 73 | 281 |
| hwb8 64 | 8/8 | 129 | 456 | 116 | 442 | 112 | 449 |
| miller 5 | 3/3 | 8 | 15 | 8 | 16 | 8 | 16 |
| mini-alu 84 | 4/2 | 11 | 20 | 10 | 22 | 10 | 20 |
| mod522_17 | 5/5 | 19 | 42 | 12 | 28 | 11 | 20 |
| one-two-three 27 | 3/3 | 10 | 14 | 9 | 16 | 9 | 16 |
| peres 4 | 3/3 | 7 | 9 | 5 | 7 | 5 |
| rd2 19 | 3/2 | 8 | 15 | 6 | 10 | 6 | 10 |
| rd3 68 | 5/3 | 20 | 49 | 13 | 34 | 13 | 34 |
| rd73 69 | 7/3 | 38 | 105 | 25 | 73 | 25 | 73 |
| rd84 70 | 8/4 | 52 | 140 | 34 | 104 | 34 | 104 |
| sym6 63 | 6/1 | 17 | 34 | 14 | 29 | 14 | 29 |
| sym9 71 | 9/1 | 35 | 79 | 27 | 62 | 27 | 62 |

Comparing each parameter of the results by using our method and the previous methods, the minimum value of each one is made bold. The results show that our algorithm is better in most cases (except $4mod5_8$, $alu_9$, ex-1_82). In the diagrams, the nodes are separated and shared nodes are less, which leads to bad results. When the numbers of input increase, the improvements of our algorithm can be observed clearly (for instance, in hwb function and rd function).

Second, the results in [22] and our results are compared. In general, our algorithm brings better results than the results using BDD-Based Synthesis. Especially at rd and hwb functions, the GC and QC reduce significantly. However, the lines reduce slightly and in hwb8_64 case, the line is higher than previous results. In two cases $4mod5_8$ and $alu_9$, it does not reach the expected results, the same as in Table 5.

| Func. | PI/ PO | BDD-Based Synthesis | Our Algorithm |
|-------|--------|----------------------|---------------|
| 4mod5 8 | 4/1 | 7 | 8 | 24 | 7 | 12 |
| decoder 10 | 2/4 | 6 | 11 | 27 | 5 | 4 |
| mini-alu 84 | 4/2 | 10 | 20 | 60 | 8 | 16 |
| alu 9 | 5/1 | 7 | 9 | 29 | 9 | 17 |
| rd2 19 | 5/3 | 11 | 34 | 98 | 12 | 26 |
| hwb5 15 | 5/5 | 28 | 88 | 276 | 26 | 64 |
| sym6 63 | 6/1 | 14 | 29 | 93 | 12 | 19 |
| mod5adder 66 | 6/6 | 32 | 96 | 292 | 27 | 78 |
| hwb6 14 | 6/6 | 46 | 159 | 507 | 46 | 115 |

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
V. CONCLUSION

This paper introduces an algorithm to synthesize reversible quantum circuits using ROCBDD and mixed-polarity Toffoli gate. Our algorithm is expected to reduce all parameters: lines, GC and QC. The experimental results in Section 4 show that our algorithm has better results in reducing GC and QC comparing with previous algorithms. However, our algorithm is limited when facing total lines numbers, which rely on the total shared nodes in the diagram.

Our algorithm has a trade-off between lines and GC of result circuits. In future work, we intend to develop another algorithm aiming to reduce all synthesizing costs of the reversible circuit. For instance, we can apply mix-polarity Toffoli gates in another method like ESOP-based method, which has the best results in lines but limits in GC.

REFERENCES

[1] C. H. Bennett, “Logical Reversibility of Computation,” IBM J. Res. Dev., vol. 17, no. 6, pp. 525–532, Nov. 1973, doi: 10.1147/rd.17.0525.

[2] R. Landauer, “Irreversibility and Heat Generation in the Computing Process,” IBM J. Res. Dev., vol. 5, no. 3, pp. 183–191, Jul. 1961, doi: 10.1147/rd.53.0183.

[3] M. Swathi and B. Rudra, “Implementation of Reversible Logic Gates with Quantum Gates,” in 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC), NV, USA, Jan. 2021, pp. 1557–1563. doi: 10.1109/CCWC51732.2021.9376060.

[4] C. Sharma, H. Pahuja, M. Dadhwal, and B. Singh, “Study of Reversible Logic Synthesis with Application in SOC: A Review,” IOP Conf. Ser. Mater. Sci. Eng., vol. 225, p. 012250, Aug. 2017, doi: 10.1088/1757-899X/225/1/012250.

[5] R. Cuykendall and D. R. Andersen, “Reversible optical computing circuits,” Opt. Lett., vol. 12, no. 7, p. 542, Jul. 1987, doi: 10.1364/OL.12.000542.

[6] J. S. Hall, “Nanocomputers and reversible logic,” Nanotechnology, vol. 5, no. 3, pp. 157–167, Jul. 1994, doi: 10.1088/0957-4484/5/3/002.

[7] M. A. Nielsen and I. L. Chuang, Quantum Computation and Quantum Information: 10th Anniversary Edition, Cambridge: Cambridge University Press, 2010. doi: 10.1017/CBO9780511976667.

[8] D. M. Miller, D. Maslov, and G. W. Dueck, “A transformation based algorithm for reversible logic synthesis,” in Proceedings of the 40th conference on Design automation - DAC ’03, Anaheim, CA, USA, 2003, p. 318. doi: 10.1145/775832.775915.

[9] D. Maslov, “Linear depth stabilizer and quantum Fourier transformation circuits with no auxiliary qubits in finite-neighbor quantum architectures,” Phys. Rev. A, vol. 76, no. 5, p. 052310, Nov. 2007, doi: 10.1103/PhysRevA.76.052310.

[10] P. Gupta, A. Agrawal, and N. K. Jha, “An Algorithm for Synthesis of Reversible Logic Circuits,” IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 25, no. 11, pp. 2317–2330, Nov. 2006, doi: 10.1109/TCAD.2006.871622.

[11] J. Donald and N. K. Jha, “Reversible logic synthesis with Fredkin and Peres gates,” ACM J. Emerg. Technol. Comput. Syst., vol. 4, no. 1, pp. 1–19, Mar. 2008, doi: 10.1145/1330521.1330523.

[12] S. C. Chua and A. K. Singh, “Search-Based Reversible Logic Synthesis Using Mixed-Polarity Gates,” in Design and Testing of Reversible Logic, vol. 577, A. K. Singh, M. Fujita, and A. Mohan, Eds. Singapore: Springer Singapore, 2020, pp. 93–113. doi: 10.1007/978-981-13-8821-7 6.

[13] D. Maslov and G. W. Dueck, “Reversible Cascades With Minimal Garbage,” IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 23, no. 11, pp. 1497–1509, Nov. 2004, doi: 10.1109/TCAD.2004.836735.

[14] [14]M. Saeedi, M. S. Zamani, M. Sedighi, and Z. Sasanian, “Reversible circuit synthesis using a cycle-based approach,” ACM J. Emerg. Technol. Comput. Syst., vol. 6, no. 4, pp. 1–26, Dec. 2010, doi: 10.1145/1877745.1877747.

[15] M. Saeedi, M. Sedighi, and M. Saheb Zamani, “A library-based synthesis methodology for reversible logic,” Microelectron. J., vol. 41, no. 4, pp. 185–194, Apr. 2010, doi: 10.1016/j.mejo.2010.02.002.

[16] Alhagi, M. Lukac, L. Tran, and M. Perkowski, “Two-stage approach to the minimization of quantum circuits based on ESOP minimization and addition of a single ancilla qubit.”

[17] C. Bandyopadhyay, D. Roy, D. K. Kole, K. Datta, and H. Rahaman, “ESOP-Based Synthesis of Reversible Circuit Using Improved Cube List,” in 2013 International Symposium on Electronic System Design, Singapore, Dec. 2013, pp. 26–30. doi: 10.1109/ISED.2013.12.

[18] K. Fazel, M. A. Thornton, and J. E. Rice, “ESOP-based Toffoli Gate Cascade Generation,” in 2007 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, Victoria, BC, Canada, Aug. 2007, pp. 206–209. doi: 10.1109/PACRIM.2007.4313212.

[19] Y. Sanaee and G. W. Dueck, “Generating Toffoli networks from ESOP expressions,” in 2009 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, Victoria, BC, Canada, Aug. 2009, pp. 715–719. doi: 10.1109/PACRIM.2009.5291232.

[20] Y. Sanaee and G. W. Dueck, “ESOP-Based Toffoli Network Generation with Transformations,” in 2010 40th IEEE International Symposium on Multiple-Valued Logic, Barcelona, Spain, 2010, pp. 276–281. doi: 10.1109/ISMVL.2010.58.

[21] J. Hu, G. Ma, and G. Feng, “Efficient Algorithm for Positive-polarity Reed-muller Expansions of reversible circuits,” in 2006 International Conference on Microelectronics, Dhahran, Saudi Arabia, Dec. 2006, pp. 63–66. doi: 10.1109/ICM.2006.373267.

[22] R. Wille and R. Drechsler, “BDD-based synthesis of reversible logic for large functions,” in Proceedings of the 46th Annual Design Automation Conference on ZZZ - DAC ’09, San Francisco, California, 2009, p. 270. doi: 10.1145/1629911.1629984.

[23] Bryant, “Graph-Based Algorithms for Boolean Function Manipulation,” IEEE Trans. Comput., vol. C-35, no. 8, pp. 677–691, Aug. 1986, doi: 10.1109/TCS.1986.1676819.

[24] R. Wille and R. Drechsler, “Effect of BDD Optimization on Synthesis of Reversible and Quantum Logic,” Electron. Notes Theor. Comput. Sci., vol. 253, no. 6, pp. 57–70, Mar. 2010, doi: 10.1016/j.entcs.2010.02.006.

[25] C. S. Cheng, A. K. Singh, and L. GOPAL, “Efficient Three Variables Reversible Logic Synthesis Using Mixed-polarity Toffoli Gate,” Proc. Comput. Sci., vol. 70, pp. 362–368, 2015, doi: 10.1016/j.procs.2015.10.035.
[26] S. M. Ridha Taha, *Reversible Logic Synthesis Methodologies with Application to Quantum Computing*, 1st ed. 2016. Cham: Springer International Publishing: Imprint: Springer, 2016. doi: 10.1007/978-3-319-23479-3.

[27] R. Drechsler and D. Sieling, “Binary decision diagrams in theory and practice,” *Int. J. Softw. Tools Technol. Transf.*, vol. 3, no. 2, pp. 112–136, May 2001, doi: 10.1007/s100090100056.

[28] *Binary Decision Diagrams (BDDs) in pure Python and Cython wrappers of CUDD, Sylvan, and BuDDy*: University of Colorado at Boulder. [Online]. Available: https://web.archive.org/web/20150317121927/http://vlsi.colorado.edu/~fabio/CUDD/node1.html

[29] R. Wille, D. Gro, L. Teuber, G. W. Dueck, and R. Drechsler, “RevLib: An Online Resource for Reversible Functions and Reversible Circuits,” in *38th International Symposium on Multiple Valued Logic (ismvl 2008)*, Dallas, TX, USA, May 2008, pp. 220–225. doi: 10.1109/ISMVL.2008.

**HIEU NGUYEN** received the B.S. degree in Control Engineering and Automation from Ho Chi Minh City University of Technology (HCMUT), Vietnam (2019). Currently, he is studying M.S in Electronics Engineering at HCMUT. He is also working as lecturer at Faculty of Electrical-Electronics Engineering, Ho Chi Minh City University of Technology – VNU HCM.

**LINH H. TRAN** received the B.S. degree in Electrical and Computer Engineering from University of Illinois, Urbana – Champaign (2005), M.S. and PhD. in Computer Engineering from Portland State University (2006, 2015). Currently, he is working as lecturer at Faculty of Electrical-Electronics Engineering, Ho Chi Minh City University of Technology – VNU HCM. His research interests include quantum/reversible logic synthesis, computer architecture, hardware-software co-design, hardware design targeting FPGAs and data analysis.