

Complexity Analysis for Garbage-Free Reversible Computation of Boolean Functions with Variable Ancilla

Zaid Al-Wardi 

Electrical Engineering Department, College of Engineering, Mustansiriyah University, Baghdad, Iraq

Email: alwardi@uomustansiriyah.edu.iq

Article Info

Received 09/08/2024
Revised 17/10/2026
Accepted 24/10/2026

Abstract

Reversible computation is a fundamental concept in quantum circuits and low-power design. As the demand for powerful and energy-efficient computing grows, novel design methods for reversible circuits become critical for achieving high performance with low power consumption. Current reversible circuit synthesis methodologies fall into two classes, each minimizing a different parameter in circuit complexity, but often result in relatively high values in another parameter. Post-synthesis optimizations moderate complexity parameters, basically by trading off between them. However, these optimizations depend on specific gate patterns and may not always meet design constraints effectively. This paper proposes a novel recursive decomposition scheme for large Boolean functions using a single-target gate model in reversible circuit synthesis. The proposed scheme systematically and efficiently manages the trade-off between reversible circuit complexity measures, offering gradually varying parameters at each level in the recursion. The scalability of the proposed approach is presented as a comprehensive complexity analysis for the mathematical model. Rather than providing a globally optimal solution, this approach offers a range of solutions, allowing designers to choose the most suitable option based on specific design constraints.

Keywords: Embedding, Garbage-free, MPMCT, Optimization, Reversible Circuit Synthesis, Single-Target Gate, Trade-off

1. Introduction

Reversible circuit design is a critical area in the field of non-conventional computing, as the demand for ever-more powerful and efficient systems continues to grow. However, as computational circuits become larger and more complex, conventional semiconductor-based technologies and traditional design methods can become inefficient. Reversible computing can act as the bridge between conventional technology and future computing paradigms, because it is a key component in the development of quantum computing, low-power circuits, and cryptography [1]-[3].

Reversible logic circuit synthesis covers the scope of performing Boolean-based computations. However, the conventional circuit design flow is much more elaborate when compared to the preliminary design tools and methodologies available for reversible logic circuit designers [4].

One challenge is that Boolean functions and operations are not inherently reversible, whereas reversible circuits, by definition, can only perform reversible operations. To address this, Boolean functions are embedded within larger computations that satisfy the conditions of reversibility. However, this

embedding requires additional lines with constant inputs, which increases the complexity of the computation [5]-[8].

There are two main approaches to reversible circuit synthesis. The first approach includes truth-table-based implementation with a minimal number of constant input lines. Truth tables grow exponentially; consequently, the gate counts of circuits synthesized using these methodologies also grow exponentially, resulting in limited scalability of this approach, such as ESOP, XOR, output permutations, or transformation-based synthesis [9]-[12].

The problem is solved by the second approach, which includes hierarchical methodologies to partition the function into smaller parts, such as BDD, HDL, and other function decomposition-based methodologies [13]-[16]. These methodologies are more scalable and usually accomplish the computation using fewer gates, with the serious drawback of exploiting high numbers of circuit lines. Constant input lines are utilized in these computations, where each line is to be represented by a qubit when implemented in quantum technology, which is a very limited resource [17],[18].

Known methodologies result in extreme circuits, which have either minimal line counts, with high gate counts and limited scalability, or circuits that have small gate counts, but with high line counts. The literature on reversible circuit design proposes post-synthesis optimization techniques to moderate circuit parameters by trading off lines with gates. These optimization techniques are applied depending on the existence of special gate patterns in the reversible circuits [19]-[21].

This paper proposes a decomposition scheme for large Boolean functions that operate in a recursive manner. At each level of recursion, the scheme generates a solution characterized by distinct circuit parameters. This flexibility allows designers to choose a solution tailored to their specific design constraints, which can be critical in scenarios where traditional methods may not provide feasible options. Unlike other methodologies that seek a single globally optimal solution, our approach provides a spectrum of solutions, each with its own trade-offs. The focus of this work on the theoretical underpinnings of the proposed scheme provides a detailed complexity analysis to demonstrate its efficacy and potential impact.

In addition to providing the complexity analysis of the computation, the proposed scheme allows the designer to have flexible choices for the appropriate number of ancilla, based on specific design constraints. This makes the proposed scheme a powerful and scalable tool for designing efficient reversible circuits.

2. Preliminaries

This section introduces reversible gates, which are basic elements in reversible circuits. It also introduces other concepts, such as embedding of Boolean functions, inverse computation, ancilla, and garbage-free computations.

2.1. Reversible Gates and Circuits

Reversibility implies no loss of information when applying a reversible gate. In other words, the number of input bits should be equal to the number of output bits, where each bit is graphically represented by a vertical line across the circuit diagram. A reversible gate performs a bijective mapping of inputs to the output, such that the inputs can be computed back from the outputs [22]. Reversible NOT, CNOT and Toffoli gates are shown in Fig. 1 (a), (b) and (c) respectively.

Despite being reversible, a Toffoli gate can be used to implement the irreversible Boolean conjunction AND operation. This can be achieved by supplying the target line with an initial value of false. A Toffoli gate performs the bijective mapping $(x_1, x_2, 0) \leftrightarrow (x_1, x_2, (x_1 \cdot x_2))$. This is known as embedding an irreversible Boolean operation within a larger reversible circuit. As shown in Fig. 2(a), embedding involves introducing constant inputs to the circuit.

A Mixed Polarity Multiple Control Toffoli MPMCT gate is a generalization of a Toffoli gate with a set of N Boolean variables, $\{x_1, \dots, x_N\}$, which act as control inputs, and one target line y . The MPMCT gates allow not only positive controls $\bullet$, but also negative controls $\circ$, which activate the inversion of the target line when the specific negative control lines are false. [23] The target line of an MPMCT gate is

$$y \oplus \bar{y}$$

(a) NOT

$$\begin{array}{c} x \text{---} \bullet \text{---} x \\ y \text{---} \oplus \text{---} x \wedge y \end{array}$$

(b) CNOT

$$\begin{array}{c} x_1 \text{---} \bullet \text{---} x_1 \\ x_2 \text{---} \bullet \text{---} x_2 \\ y \text{---} \oplus \text{---} (x_1 \cdot x_2) \wedge y \end{array}$$

(c) Toffoli

$$\begin{array}{c} x_1 \text{---} \circ \text{---} x_1 \\ x_2 \text{---} \bullet \text{---} x_2 \\ x_3 \text{---} \bullet \text{---} x_3 \\ x_4 \text{---} \circ \text{---} x_4 \\ y \text{---} \oplus \text{---} (\bar{x}_1 \cdot x_2 \cdot x_3 \cdot \bar{x}_4) \wedge y \end{array}$$

(d) MPMCT

$$\begin{array}{c} x_1 \text{---} \text{---} x_1 \\ \vdots \text{---} \text{---} \vdots \\ x_N \text{---} \text{---} x_N \\ \text{---} \text{---} f \\ y \text{---} \oplus \text{---} f \wedge y \end{array}$$

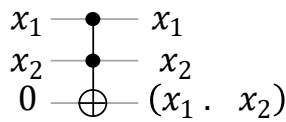
(e) Single target gate

Figure 1. Reversible logic gates.

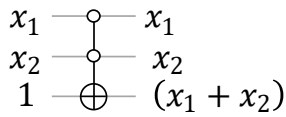
inverted only when the conjunction of all variables is evaluated to true, i.e., when all the positive controls are true and all the negative controls are false.

Fig. 1(d) shows the graphical representation of an MPMCT gate, where the Boolean expression of the control AND term between all control variables is presented in the proper polarity of each control. To this end, a multi-variable Boolean AND term can be embedded using an MPMCT gate; see Fig. 2(a). An MPMCT gate can also be used to embed a Boolean disjunction OR term, as shown in Fig. 2(b).

A single-target gate is defined with further generalization, such that more sophisticated functionality is achieved, with one target line y , N control lines $\{x_1, \dots, x_N\}$, and a control function $f(x_1, \dots, x_N)$. The target variable y , of a single-target gate is inverted only when the control function f evaluates to true [24]. In other words, a single-target gate performs the bijective mapping $(x_1, \dots, x_N, y) \leftrightarrow (x_1, \dots, x_N, f(x_1, \dots, x_N) \wedge y)$, as shown in Fig. 1(e). To this end, a single-target gate, with sufficient control variables, can be used to embed any arbitrary single output Boolean function. The above bijective mappings show that, by applying any of these gates, all variables, except the target-line variable, maintain their initial value.



(a) AND



(b) OR

Figure 2. Embedded Boolean operations

2.2. Garbage-free Computations with Ancilla

A reversible computation requires, usually, more than one operation. In other words, several gates are expected to be applied successively to the circuit lines to perform the required computation. This cascade of gates is known as a reversible logic circuit.

For example, considering the simple Boolean function $f = (\bar{x}_1 \cdot x_2) + (x_2 \cdot \bar{x}_3)$, the function is represented in sum-of-products (SOP) form as a Boolean disjunction OR of two cubes, $f = P_1 + P_2$. A cube, in the context of a Boolean function in SOP representation, is a conjunction term that represents a specific combination of input variables. The function contains irreversible operations; therefore, embedding is inevitable to implement the function. Hence, constant inputs must be used, as illustrated in Fig. 3.

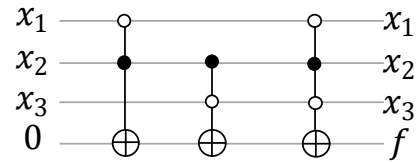
One approach to synthesize the function f using ESOP-based synthesis, which guarantees only one constant-input line, is to compute f as mainly implemented by using XOR form, where $f = P_1 + P_2 = P_1 \wedge P_2 \wedge (P_1 \cdot P_2)$ see Fig. 3(a).

If the function is slightly larger, e.g., with only three cubes, the XOR expansion of such a function contains seven product terms. This approach is ancilla-free, but it is exponential in gate count, i.e., it has very limited scalability.

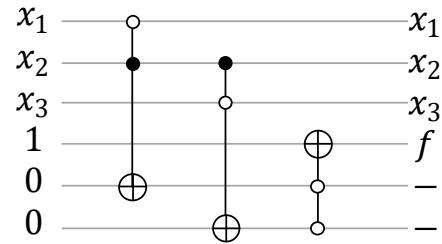
The computation can be approached differently, by splitting the function into two cubes and implementing each cube using a single-target gate. A third single-target gate is then applied to perform an OR operation of the two cubic terms to compute the main function, as presented in Fig. 3(b). In this example three gates are considered, the first two gates perform intermediate computations, which are used by the third gate to perform the final computation. The two constant inputs that are used to facilitate the intermediate computations are considered as garbage outputs.

If the function is slightly larger, e.g., with three cubes in its SOP representation, the implementation requires an additional constant input to be utilized.

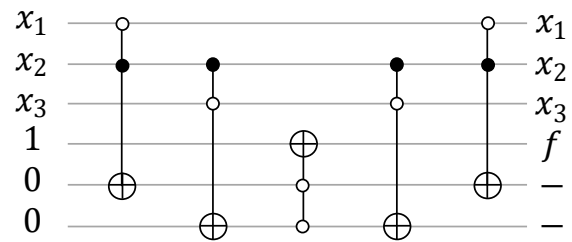
In this approach, both line and gate counts increase linearly, which is not feasible when mapping the circuits to quantum technology [17]. Ancilla refers to a constant input that is temporarily used for intermediate computations and is reset



(a) ESOP-based computation



(b) SOP-based computation



(c) Garbage-free SOP-based computation

Figure 3. Computing $f = (\bar{x}_1 \cdot x_2) + (x_2 \cdot \bar{x}_3)$.

later to its initial value [25]. Using an ancilla allows garbage-free computation, in which inverse computations reset the ancilla [26].

It can be shown that successively applying any gate twice results in an identity mapping, i.e., no change in the variables. In other words, all the gates are self-invertible. Generally, to compute the inverse of a mapping, means applying the gates in reverse order, as shown in Fig. 3(c).

Both approaches are not practical for large functions; the first, e.g., ESOP-based synthesis, has limited scalability due to the exponential growth of gate counts. The second approach has also limited scalability, due to its linear accumulation of lines.

Post-synthesis optimization is usually applied to tune the circuit parameters, by a trade-off between gates and lines. Unfortunately, these optimization techniques have only limited control over the parameters, because they depend on the existence of special patterns in the circuits [17]-[19].

3. Motivation

The objective of this work is to present a systematic design methodology that provides better control of the trade-off level between circuit parameters. In this paper, the single-target gates are considered as the building blocks in a specific reversible circuit design. Consequently, the implementations of these gates must follow the constraints of that circuit, including the number of provided ancilla.

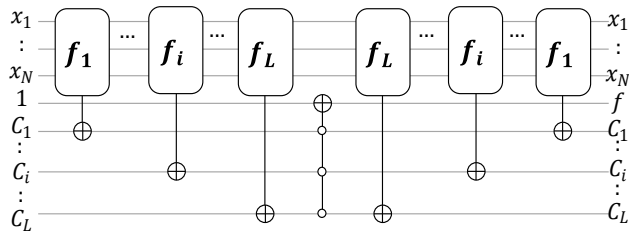


Figure 4. The partitioning scheme for computing a Boolean function f by utilizing ancilla

Assume a single-target gate, controlled by the Boolean function f , which has an SOP representation with K cubes, i.e., $f(x_1, \dots, x_N) = \sum_{j=1}^K P_j$, where $\sum$ represents a Boolean disjunctive operator OR. A question that may arise here is how this gate can be synthesized by utilizing L ancilla, and a minimum number of MPMCT gates β ?

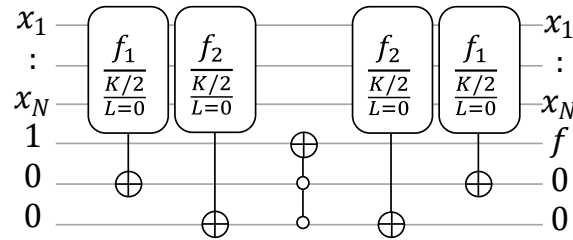
An intuitive possible approach to achieve this objective is by sub-setting or partitioning the set of cubes that represent the

desired function f , into L sub-sets, hence defining L sub-functions, $\{f_1, \dots, f_L\}$, i.e., $f = \sum_{i=1}^L f_i$, as shown in Fig. 4.

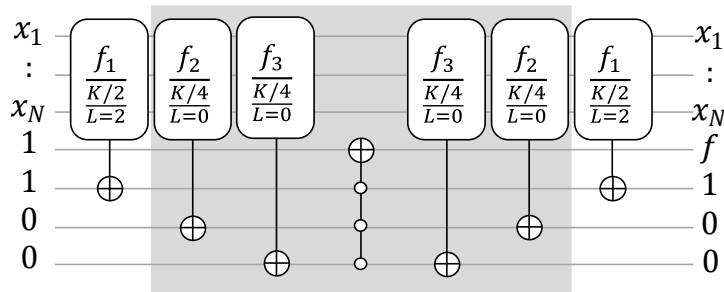
Each sub-function is represented as a single-target gate, which has its target line applied to a different ancilla. For example, the target line of f_1 is applied to C_1 , and the target line of f_2 is applied to C_2 until the target line of the last sub-function f_L is applied to C_L .

The first sub-function f_1 assigns its output to the first ancilla and utilizes the spared $L-1$ ancilla, i.e., $\{C_2, \dots, C_L\}$, for intermediate computations of f_1 . Since the ancilla is reset internally to its initial values, they become available to be reused for further computations.

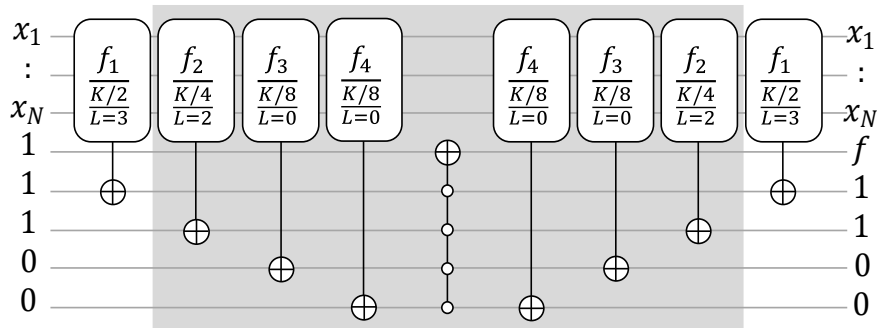
The second sub-function f_2 has its target output line applied to ancilla C_2 . Hence, utilizes the spared $L-2$ ancilla, i.e., $\{C_3, \dots, C_L\}$, for intermediate computations of f_2 . This continues until the last sub-function f_L assigns its output to the last ancilla C_L and hence spares no ancilla to be utilized for its internal computations.



(a) $L=2$



(b) $L=3$



(c) $L=4$

Figure 5. The proposed scheme for computing the function f with different L ancilla

In general, according to this scheme, the i^{th} sub-function f_i has its target line applied to the i^{th} ancilla and can utilize $(L - i)$ ancilla, for the internal computations of f_i .

The target lines of all sub-functions are considered controls for an MPMCT gate, which performs as a disjunction OR to compute the overall function. Afterward, all intermediate sub-functions are inversely computed by repeating the set of single-target gates in reverse order, such that all the ancilla are reset to their initial default values.

The question in this work is how to optimize the sub-setting of the cubes of the function f , such that it computes f with fewer MPMCT gates? In other words, how to best partition a function into smaller sub-functions to minimize the number of MPMCT gates required to compute such a function.

4. Methodology

This section introduces the sub-setting of the cubes of a certain function to define smaller sub-functions, and how these sub-functions can be implemented recursively using specific numbers of ancilla. The recursive decomposition breaks down large Boolean functions, generating solutions with varying circuit parameters at each recursion. This process ensures that designers can select from a range of solutions, each tailored to different design constraints. The scheme's flexibility is a significant departure from traditional approaches that look for a single, globally optimal solution. The primary objective is to provide a practical tool for designers faced with specific constraints, offering potentially the only viable solutions in some cases.

The first level of sub-setting considers splitting the cubes into groups, i.e., partitioning the function f into two sub-functions $\{f_1, f_2\}$. In this case, two ancillas are needed to compute the two sub-functions. If the circuit provides only two ancillas, i.e., $L = 2$, to compute f , both lines are used in computing the two sub-functions, sparing no ancilla for the computation of the two sub-functions. In other words, the single-target gates that compute the two sub-functions must be computed using an ancilla-free methodology, e.g., using ESOP-based synthesis.

It can be shown that the best sub-setting can be achieved when each sub-function is represented using approximately $\lceil \frac{K}{2} \rceil$ cubes. These two sub-functions are computed with an ancilla-free methodology, i.e., $L = 0$, as explained in Fig. 5(a). In Fig. 5, the single-target gates are marked with three parameters: the control function f_i , the number of cubes that represent f_i , and the number of ancilla used to implement this gate.

The next level of sub-setting cubes is when three ancilla are used, i.e., $L = 3$. The function is partitioned into three sub-functions, where the three ancilla are used to intermediately compute the sub-functions $\{f_1, f_2, f_3\}$ respectively. Then the disjunction of all sub-functions is computed using an MPMCT gate, before the sub-functions are inversely computed for resetting the ancilla, as shown in Fig. 5(b).

The central shaded part in Fig. 5(b) shows that the computation scheme of the last two sub-functions $\{f_2, f_3\}$ is identical to the computation of f in Fig. 5(a). On the other hand, the first sub-

function f_1 is computed to the first ancilla, sparing two ancillas to be used to implement f_1 , which can be recursively implemented by splitting f_1 into two sub-functions, as explained earlier with the main function f , as shown in Fig. 5(a).

The first sub-function is implemented recursively by calling the implementation of a function with one less ancilla. On one hand, the sub-function f_1 is computed by utilizing two ancillas; on the other hand, the sub-function pair $\{f_2, f_3\}$ is also computed according to a similar scheme. Consequently, it is fair to subset the cubes of the main function into two sub-groups, each with approximately $\lceil \frac{K}{2} \rceil$ cubes, and assign the first half to the first sub-function f_1 , while further splitting the second sub-group into two quarters of approximately $\lceil \frac{K}{4} \rceil$ cubes, and assign each quarter to one of the latter two sub-functions, f_2 , and f_3 .

This splitting scheme can be recursively extended to a new level, with each additional ancilla. For example, when the computation utilizes a fourth ancilla, see Fig. 5(c), in which the central shaded part of the scheme shows the computation of the latter three sub-functions by utilizing three ancilla, while similarly the first sub-function is computed. In this scheme, the sub-groups of $\lceil \frac{K}{2} \rceil$, $\lceil \frac{K}{4} \rceil$, $\lceil \frac{K}{8} \rceil$ and $\lceil \frac{K}{8} \rceil$ cubes are assigned to sub-functions f_1, f_2, f_3 and f_4 respectively.

The generalized formulation for the proposed garbage-free computation scheme of function f , by utilizing $L \geq 2$ ancilla is:

1. For the single-output Boolean function $f(x_1, \dots, x_N) = \sum_{j=1}^K P_j$, define a set of L sub-functions, such that the function f is the disjunction of all these sub-functions, i.e., $f = \sum_{i=1}^L f_i$.
2. Apply L single-target gates to the ancilla, such that the ancilla C_i is the target line of the single-target gate, which is controlled by the sub-function f_i .
3. Assign a subset of the cubes of f to each sub-function, as in (1) below, such that:

$$f_i = \sum_{j=1 + \frac{2^{(i-1)-1}}{2^{(L-1)}}K}^{\frac{2^{i-1}-1}{2^L}K} P_j, \quad i = 1, \dots, L-1 \quad (1)$$

while the last sub-function f_L is defined as in (2) below:

$$f_L = \sum_{j=1 + \frac{2^{(L-1)-1}}{2^{(L-1)}}K}^K P_j \quad (2)$$

4. Recursively compute the single-target gate of each sub-function f_i by using $L - 1$ ancilla.
5. Apply an MPMCT gate with negative control to all ancilla to compute the disjunction of all sub-functions, which computes the final value of f .
6. For garbage-free computation, inversely compute the sub-functions by applying the same gate structure in reverse order.

Before going to the next section, a few notes should be mentioned:

- The sub-setting assumes the number of cubes to be a power of two, such that each division always results in an integer. In the case of an odd number of cubes, the first sub-function can be defined with one less cube than the other sub-group.
- The last two sub-functions are both computed using an ancilla-free implementation, e.g., ESOP-based computation. That means f_{L-1} is an exception to point 4 above, because applying the formula means that f_{L-1} is to be implemented by utilizing one ancilla, while the proposed scheme is designed for $L \geq 2$.
- ESOP-based synthesis computes the function f . Therefore, the last two ancilla, which compute the last two sub-functions $\{f_{L-1}, f_L\}$, using ESOP-based implementation, are both initially false “0”. On the other hand, the proposed scheme computes the function f , when the line to which f is computed, is initially “1”. In other words, all constant inputs are initially true, except the last two ancilla.

5. Complexity Analysis

This section provides the complexity analysis of the proposed scheme and shows the upper bound for single-target gates' complexity expansion.

First, the complexity of computing the desired function f without utilizing ancilla must be found. For example, ESOP-based synthesis converts the function such that it is computed by applying XOR operations. The relation between the SOP form and its ESOP expansion can be explained by the simple relation $A + B = A \wedge B \wedge AB$, in which the left-hand side contains two terms in an SOP expression, while the right-hand side contains three terms in an ESOP expression. Generally, the relation between the number of cubes in the SOP representation of a function and the number of conjunction terms in the equivalent ESOP expansion is given by (3) below:

$$\beta_{K,0} = 2^K - 1 \quad (3)$$

where $\beta_{K,0}$ means that there exists an implementation for a single-target gate, activated by a Boolean function, which has an SOP representation of K cubes, and with no ancilla, $L = 0$, using β MPMCT gates. The first stage in partitioning the implementation is the function cubes sub-setting into two sub-functions, each to be represented by a single-target gate, as shown in Fig. 2(a), where $L = 2$ ancilla are utilized in the computation. To this end, the MPMCT-gate-count with two ancilla is expressed as in (4) below:

$$\beta_{K,2} = 4 \beta_{\lfloor \frac{K}{2} \rfloor,0} + 1 \quad (4)$$

The next level introduces three ancillas $L = 3$, as shown in Fig. 5(b). The implementation at this level is exactly three times larger than the previous step of sub-setting. Equation (5) gives an MPMCT-gate-count applicable for each new ancilla utilized in the computation, which can lead to the following recursive equation, when $L \geq 3$:

$$\beta_{K,L} = 3 \beta_{\lfloor \frac{K}{2} \rfloor, L-1} \quad (5)$$

Equation (6) shows a non-recursive expression for the MPMCT-gate-count complexity:

$$\beta_{K,L} = 3^{(L-2)} \left(4 \left(2^{\lfloor \frac{K}{2^{L-1}} \rfloor} - 1 \right) + 1 \right) \quad (6)$$

where $2 \leq L \leq \lceil \log_2 K \rceil + 1$.

It can be shown, from (6), that a minimal MPMCT-gate-count $\check{\beta}$ is achieved when the number of ancilla used in the implementation is $\check{L} = \lceil \log_2 K \rceil$ lines and is given by (7) below:

$$\check{\beta}_K = 13 \left(3^{\lceil \log_2 K \rceil - 2} \right) \quad (7)$$

6. Discussion

The previous section shows the upper-bound analysis of the complexity expansion of the proposed implementation scheme. A numerical comparison shows that the growth of the ESOP-based synthesis is exponential and hence reaches very large gate counts. In comparison, the proposed algorithm can keep the growth fairly slow by efficiently choosing the number of ancillas, as shown in Table 1.

Table 1. The expansion of the ESOP-based implementation in comparison to the proposed minimal implementation.

K	$\beta_{K,0}$	$\check{\beta}_K$	$\check{L}$
1	1	1	0
2	3	3	0
4	15	13	2
8	255	39	3
16	65535	117	4
32	4967295	351	5
64	$\cong 1.844 \times 10^{19}$	1053	6
128	$\cong 3.403 \times 10^{39}$	3105	7
256	$\cong 1.158 \times 10^{77}$	9315	8
512	$\cong 1.341 \times 10^{154}$	27945	9
1024	$\cong 1.797 \times 10^{308}$	83835	10

The rate of growth in the proposed implementation can be simply described by the following condition statement: when the size of the function doubles, use the proposed implementation scheme with one more ancilla, and the overall gate count is multiplied by three.

The numbers in Table 1 are the upper bounds, or worst-case scenarios, and there exist Boolean functions that are computed with fewer MPMCT gates. An example of a pattern that reduces the XOR expansion of a function occurs when two cubes in an SOP expression have a complementary polarity in one of their variables, e.g., $x.A + \bar{x}.B = x.A \wedge \bar{x}.B$ because of the conjunction of the two terms $(x.A.\bar{x}.B) = 0$. This means that the number of overall conjunction terms in the ESOP expressions is likely to be less than the worst-case scenario. This can reduce the gate counts in both the ESOP-based

implementation and the proposed scheme because it involves ESOP-based synthesis at its base level. Rearranging the cubes' order in the SOP expression to increase the probability of complementary variables in adjacent cubes can be considered to reduce the overall gate counts.

The main contribution of this work is exploiting a recursive decomposition scheme for Boolean functions, offering a spectrum of solutions with varying circuit parameters. This approach provides designers with the flexibility to select the most appropriate solution based on their unique constraints. The novelty of this work comes from approaching the complexity growth in terms of function size and also the number of ancilla utilized. The paper emphasizes that the purpose of this work is not to present a single best solution, but to offer a range of feasible solutions. Given the theoretical nature of this work, it focused on the mathematical model and complexity analysis, rather than experimental results or direct comparisons with existing methodologies.

7. Conclusions

Our proposed systematic recursive decomposition scheme for Boolean functions represents a significant advancement in providing designers with a range of solutions to choose from, based on specific design constraints. The desired function is implemented as a single-target reversible gate composed of a number of MPMCT gates, with variable ancilla. This work is purely theoretical, to establish a new framework for approaching Boolean function decomposition with a focus on flexibility and practical applicability, but the current focus is on the foundational theoretical aspects. The complexity analysis by itself shows how Boolean function implementation grows with respect to function size, and what the impact is of introducing a new ancilla to the circuit. Future work may include defining criteria for rearranging the cubes, with an empirical validation.

Acknowledgements

The author would like to acknowledge Mustansiriyah University, URL: (www.uomustansiriyah.edu.iq), for the support, which was instrumental in completing this work.

Conflict of interest

The author declares that there are no conflicts of interest regarding the publication of this manuscript.

Author Contribution Statement

Zaid Al-Wardi identified and proposed the problem, suggested the approach to tackle the problem, developed the theory and performed the computations, and discussed the results and contributed to the final manuscript.

Use of LLM and AI

None declared.

Research funding

The author received no external financial funding for the research, authorship, and publication of this article.

Data Availability

Not applicable.

Abbreviations and Symbols

β	MPMCT gate count for function f implementation
$\tilde{\beta}$	Minimum MPMCT gate count implementation of f
C_l	Constant input reversible line.
CNOT	Controlled NOT reversible gate
ESOP	Exclusive Sum-of-Products
f	Single output Boolean function
K	Number of cubes in SOP representation of f
L	Number of ancilla utilized for computing f
$\mathbf{L}$	Number of ancilla for minimum MPMCT gate count
MPMCT	Mixed-polarity multiple-control Toffoli gate
P_j	Boolean product (conjunction) term
SOP	Sum-of-Products
x_i	Binary-valued input

References

- [1] M. P. Frank and K. Shukla, "Quantum foundations of classical reversible computing," *Entropy*, vol. 23, no. 6, Art. no. 701, Jun. 2021. doi: <https://doi.org/10.3390/e23060701>.
- [2] A. K. Rajput, S. Chouhan, and M. Pattanaik, "Low power Boolean logic circuits using reversible logic gates," in *Proc. International Conference on Advances in Computing, Communication and Control (ICAC3)*, Mumbai, India, Dec. 2019, pp. 1–6. doi: <https://doi.org/10.1109/ICAC347590.2019.9036799>.
- [3] Q.-B. Luo, X.-Y. Li, G.-W. Yang, and Q. Li, "Quantum reversible circuits for GF(2⁸) multiplication based on composite field arithmetic operations," *Quantum Information Processing*, vol. 22, Art. no. 58, Jan. 2023. doi: <https://doi.org/10.1007/s11128-022-03799-w>.
- [4] Z. Al-Wardi, R. Wille, and R. Drechsler, "Towards VHDL-based design of reversible circuits," in *Proceedings of Reversible Computation: 9th International Conference*, Kolkata, India, 2017, pp. 102–108, doi: https://doi.org/10.1007/978-3-319-59936-6_8.
- [5] F. -J. Chao and Y. -C. Chen, "Decomposition Attack on Structural Logic Locking of Reversible Circuits," in *Proc. 43rd International Conference on Computer Design (ICCD)*, Richardson, TX, USA, 2025, pp. 646–653, doi: <https://doi.org/10.1109/ICCD65941.2025.00098>.
- [6] Z. Al-Wardi, "Reducing reversible circuit line count using Boolean techniques," in *Proc. 3rd International Scientific Conference of Engineering Sciences (ISCES)*, Diyala, Iraq, May 2023, pp. 42–47. doi: <https://doi.org/10.1109/ISCES58193.2023.10311466>.
- [7] R. Wille, M. Haghparast, S. Adarsh, and M. Tanmay, "Towards HDL-based synthesis of reversible circuits with no additional lines," in *Proc. 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Westminster, CO, USA, Nov. 2019, pp. 1–7. doi: <https://doi.org/10.1109/ICCAD45719.2019.8942156>.
- [8] M. Miao, J. Tang, G. Hu and H. Zhang, "BDD2Seq: Enabling Scalable Reversible-Circuit Synthesis via Graph-to-Sequence Learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, Singapore, Jan. 2026, vol. 40, no. 2, pp. 863–872. doi: <https://doi.org/10.1609/aaai.v40i2.37054>.
- [9] H. Rienr, R. Ehlers, B. d. O. Schmitt, and G. De Micheli, "Exact synthesis of ESOP forms," in *Advanced Boolean Techniques: Selected Papers from the 13th International Workshop on Boolean Problems*, Switzerland: Springer, 2020, ch. 7, pp. 177–194. doi: https://doi.org/10.1007/978-3-030-20323-8_8.
- [10] A. Bernasconi, A. Berti, V. Ciriani, G. M. Del Corso, and I. Fulginiti, "XOR-AND-XOR logic forms for autosymmetric functions and applications to quantum computing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 6, pp. 1861–1872, Jun. 2023. doi: <https://doi.org/10.1109/TCAD.2022.3213214>.

- [11] K. Datta, I. Sengupta, H. Rahaman, and R. Drechsler, "An evolutionary approach to reversible logic synthesis using output permutation," in *Proc. 8th International Design and Test Symposium (IDT)*, Marrakesh, Morocco, Dec. 2013, pp. 1–6. doi: <https://doi.org/10.1109/IDT.2013.6727117>.
- [12] D. M. Miller, D. Maslov, and G. W. Dueck, "A transformation based algorithm for reversible logic synthesis," in *Proc. 40th Annual Design Automation Conference (DAC)*, Anaheim, CA, USA, Jun. 2003, pp. 318–323, doi: <https://doi.org/10.1145/775832.775915>.
- [13] K.S. Fong and M. Motani, "SyREC: A Symbolic-Regression-Based Ensemble Combiner," in *Proc. 36th International Conference on Tools with Artificial Intelligence (ICTAI)*, Herndon, VA, USA, Oct. 2024, pp. 774–781, doi: <https://doi.org/10.1109/ICTAI62512.2024.00114>.
- [14] A. Hawash, A. Awad, and B. Abdalhaq, "Reversible circuit synthesis time reduction based on subtree-circuit mapping," *Applied Sciences*, vol. 10, no. 12, Art. no. 4147, Jun. 2020. doi: <https://doi.org/10.3390/app10124147>.
- [15] D. Bu, J. Deng, P. Tang, and S. Yang, "Reversible circuit synthesis method using sub-graphs of shared functional decision diagrams," *The Computer Journal*, vol. 66, no. 10, pp. 2574–2592, Oct. 2023. doi: <https://doi.org/10.1093/comjnl/bxac107>.
- [16] H. Lee, K. C. Jeong, D. Han and P. Kim, "An algorithm for reversible logic circuit synthesis based on tensor decomposition," *ACM Transactions on Quantum Computing*, vol. 5, no. 3, pp. 1–28, 2024, doi: <https://doi.org/10.1145/3673242>.
- [17] L. Chhangte and A. Chakrabarty, "Mapping quantum circuits in IBM Q devices using progressive qubit assignment for global ordering," *New Generation Computing*, vol. 40, no. 1, pp. 311–338, Apr. 2022. doi: <https://doi.org/10.1007/s00354-022-00163-5>.
- [18] R. Rietsche, C. Dremel, S. Bosch, L. Steinacker, M. Meckel, and J.-M. Leimeister, "Quantum computing," *Electronic Markets*, vol. 32, no. 4, pp. 2525–2536, Dec. 2022. doi: <https://doi.org/10.1007/s12525-022-00570-y>.
- [19] R. Wille, M. Soeken, D. M. Miller, and R. Drechsler, "Trading off circuit lines and gate costs in the synthesis of reversible logic," *Integration, the VLSI Journal*, vol. 47, no. 2, pp. 284–294, Mar. 2014. doi: <https://doi.org/10.1016/j.vlsi.2013.08.002>.
- [20] N. Abdessaied, R. Wille, M. Soeken, and R. Drechsler, "Reducing the depth of quantum circuits using additional circuit lines," in *Reversible Computation: 5th International Conference, RC 2013, Victoria, BC, Canada, July 4–5, 2013, Proceedings*, A. De Vos and R. Wille, Eds. Berlin, Germany: Springer, 2013, pp. 221–233. doi: https://doi.org/10.1007/978-3-642-38986-3_18.
- [21] R. Wille, M. Soeken, and R. Drechsler, "Reducing the number of lines in reversible circuits," in *Proc. 47th Design Automation Conference (DAC)*, Anaheim, CA, USA, Jun. 2010, pp. 647–652. doi: <https://doi.org/10.1145/1837274.1837439>.
- [22] S. S. Samrin, R. Patil, S. Itagi, S. C. Chetti, and A. Tasneem, "Design of logic gates using reversible gates with reduced quantum cost," *Global Transitions Proceedings*, vol. 3, no. 1, pp. 136–141, Jun. 2022. doi: <https://doi.org/10.1016/j.gltp.2022.04.011>.
- [23] M. M. Rahman, M. Soeken, and G. W. Dueck, "Dynamic template matching with mixed-polarity Toffoli gates," in *Proc. 2015 IEEE 45th International Symposium on Multiple-Valued Logic (ISMVL)*, Waterloo, ON, Canada, May 2015, pp. 72–77. doi: <https://doi.org/10.1109/ISMVL.2015.44>.
- [24] N. Abdessaied, M. Soeken, and R. Drechsler, "Technology mapping for single target gate based circuits using Boolean functional decomposition," in *Proceedings of Reversible Computation: 7th International Conference, 2015, Grenoble, France, 2015*, pp. 219–232. doi: https://doi.org/10.1007/978-3-319-20860-2_14.
- [25] M. Soeken, L. Tague, G. W. Dueck, and R. Drechsler, "Ancilla-free synthesis of large reversible functions using binary decision diagrams," *Journal of Symbolic Computation*, vol. 73, pp. 1–26, Mar. 2016, doi: <https://doi.org/10.1016/j.jsc.2015.03.002>.
- [26] Y. Ding, X.-C. Wu, A. Holmes, A. Wiseth, D. Franklin, M. Martonosi, and F. T. Chong, "SQUARE: Strategic quantum ancilla reuse for modular quantum programs via cost-effective uncomputation," in *Proc. 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, Valencia, Spain, May 2020, pp. 570–583, doi: <https://doi.org/10.1109/ISCA45697.2020.00054>.