Università degli Studi di Milano
Dipartimento di Informatica
Doctoral Program in Computer Science

Advanced Methods in Quantum and Reversible Circuit Synthesis for Boolean and Multivalued Logic

Asma Taheri Monfared

supervisor Prof. Valentina Ciriani

co-supervisor Dr. Majid Haghparast

chair of the doctoral program Prof. Roberto Sassi

September 2024 - Cycle 37

Abstract

The advancement of quantum computing significantly depends on the effective design and synthesis of quantum reversible circuits. This thesis introduces innovative methods for quantum circuit synthesis, with a particular emphasis on Boolean functions and multi-valued logic systems. We investigated the inherent regularities in dimension-reducible functions to develop novel methodologies for generating compact reversible circuits. These techniques lead to substantial reductions in quantum cost and area while enhancing the overall efficiency of quantum circuit design. Additionally, we extend our work to autosymmetric and dimension-reducible functions, demonstrating improved circuit compactness and cost-effectiveness. We also explore the application of the Projected Sum of Product (PSOP) decomposition technique, which facilitates the optimization of gate count and circuit depth, addressing essential challenges in modern quantum hardware. This method simplifies complex operations, enabling the design of practical quantum circuits that are both efficient and scalable. In the latter part of this thesis, we transition to multi-valued logic, specifically focusing on ternary and quaternary circuits. We propose novel designs for quantum reversible ternary decoders, multiplexers, and demultiplexers that aim to reduce circuit width while enhancing overall performance, thus overcoming the limitations of traditional binary systems. Our research also includes the development of a quantum ternary image processing circuit utilizing ternary reversible gates to lower quantum cost and enhance functionality. We present a balanced ternary reversible comparator that showcases significant improvements in quantum cost and efficiency, highlighting the advantages of balanced ternary logic in quantum applications. Finally, we establish a comprehensive framework for optimizing quaternary reversible circuits, concentrating on scalable designs for multiplexers and demultiplexers, which are crucial components in arithmetic logic units. This thesis contributes to the field of quantum computing by advancing the synthesis of efficient quantum circuits. We exploit function regularities and decomposition techniques to optimize quantum binary circuit designs, resulting in enhancements in circuit size and cost. Additionally, we develop multi-valued reversible circuits that demonstrate greater efficiency compared to existing designs. These findings not only enhance theoretical understanding but also lay the groundwork for future research and practical applications in quantum technologies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The reversible computation paradigm has gained considerable interest as a viable alternative to traditional irreversible logic. In reversible computation, all operations are bijective, allowing computations to move both forward and backwards. This characteristic enables the retrieval of inputs from outputs and vice versa, ensuring that information processing is lossless. The initial enthusiasm for reversible computation stems from the groundbreaking research conducted by Landauer and Bennett. [83, 11] proposed that logical reversibility could play a vital role in developing energy-efficient circuits. Their research shows that the loss of information associated with non-reversibility is fundamentally related to a circuit's power consumption, as energy dissipation is directly proportional to the energy required to represent the signal [51, 150]. In theory, since reversible computations do not result in information loss, energy dissipation could be eliminated. Additionally, reversible computation has been applied in a range of areas, including adiabatic circuits, encoder design, on-chip interconnects, and quantum computing [109, 143]. Numerous experiments have shown the successful implementation of universal quantum gate sets in various physical systems, including atomic-scale applications [4], superconducting circuits [48, 117], linear optical systems [142, 50], and other physical architectures. Quantum computing has the potential to significantly reduce the computational complexity of many problems, providing greater efficiency compared to classical computing. For instance, by employing quantum algorithms, only $\sqrt{N}$ steps are needed to search an unstructured database, as opposed to the N steps required by classical algorithms [57, 31, 148].

In recent times, advancements in quantum architectures have reignited interest in quantum computing and ensuring the creation of innovative secure cryptographic protocols. Consequently, the area of quantum logic synthesis has garnered significant attention. Many quantum algorithms, such as Grover's search algorithm, often necessitate the computation of oracles, which are subroutines defined as classical logic functions [109]. The conventional approach to synthesizing quantum oracles generally involves two main stages: reversible logic synthesis and quantum compilation. Given that the evolution of quantum systems is governed by reversible unitary operators, classical logic functions must initially be implemented as reversible circuits. Subsequently, each reversible gate is broken down into a series of basic unitary quantum gates according to a specified quantum gate library. These two steps should aim to minimize the overall gate count of the quantum circuits. Recently, new techniques for reversible circuit synthesis and quantum compilation have been introduced in the literature [91, 130, 128, 92]. Among these, one method leverages a structural regularity known as autosymmetry to synthesize compact quantum circuits, as discussed in [13].

Furthermore, multiple-valued logic is attracting significant interest as future challenges for binary logic are expected to be considerable, primarily due to significant thermal and reliability concerns [123]. In the domain of reversible binary logic, reversible multiple-valued logic has also been shown to be more secure in quantum cryptography and more efficient in quantum information processing [10, 30, 133]. Additionally, it offers lower interconnection complexity, decreased power consumption, and enhanced error tolerance in quantum computations [81, 96].

In multi-valued systems, ternary logic exhibits enhanced noise immunity [46] and represents the most cost-effective radix, determined by analyzing the width and depth of number representation [63, 80]. Furthermore, recent studies indicate that quantum computers utilizing ternary logic are 37% more compact than those using binary logic [61]. In quantum ternary systems, each unit of information is referred to as a qutrit. Theoretically, reversible ternary logic can be employed to design quantum computing structures [106]. Consequently, researchers have recently dedicated considerable attention to reversible ternary logic, leading to numerous contributions in the literature regarding ternary reversible circuit designs [1, 105, 121].

It is important to note that ternary arithmetic functions can be represented in two forms: balanced and unbalanced. In the unbalanced representation, a qutrit is depicted by the values 0, 1, or 2, while in the balanced representation, it is represented by the values $\hat{1}$, 0, or 1. Compared to unbalanced ternary representation, balanced ternary representation allows for a more efficient implementation of arithmetic functions. When designing efficient arithmetic circuits, balanced ternary representations offer significant advantages, including straightforward ternary inversion, the removal of the additional sign digit, equivalence between rounding to the nearest integer and truncation, ease in generating partial products, a reduced likelihood of carry generation, and shorter carry ripple structures in comparison to unbalanced ternary representations [46].

Another popular logic multi-valued systems is quaternary. In this logic, two bits can be combined into quaternary values to represent binary logic functions [77], which is consider as a limitation in ternary logic. In quantum quaternary logic, the memory unit is referred to as a qudit. Recently, numerous essential circuits have been developed using quaternary reversible logic, including comparators, parallel adders, full adders, half adders, subtractors, and decoders [136, 59, 111, 118, 108].

In general, metrics such as quantum cost, garbage outputs, and constant (ancilla) inputs play essential roles in the design and optimization of quantum and reversible circuits. Minimizing these key factors can significantly enhance the overall efficiency of quantum reversible logic design, resulting in circuits that are more effective and resource-efficient. A sample circuit is included in Figure 1.1 to illustrate these concepts. The following sections provide explanations of these metrics:

- Quantum cost refers to the number of primitive gates, required to realize the circuit.

- The number of garbage outputs indicates the outputs which are generated to preserve one-to-one mappings but have unimportant values.

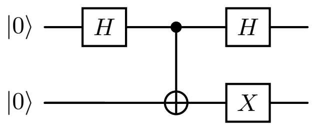


Figure 1.1: Simple quantum circuit.

- The number of constant inputs signifies the number of inputs which must be maintained constant in order to synthesize the specified logic function.

In this thesis, the aim is to reduce the total gate count of quantum circuits to improve their execution efficiency concerning quantum cost and circuit size. Our investigation into quantum circuit optimization is organized around two interconnected approaches:

In **Part I: Exploiting Function Regularities and Decomposition for Quantum Synthesis**, we focus on minimizing the cost of quantum circuits by utilizing inherent regularities in Boolean functions and implementing decomposition techniques. In **Chapter 2**, we introduce the fundamental concepts of decomposition and quantum computing, emphasizing function regularities, PSOP decomposition, and quantum synthesis. **Chapter 3** discusses quantum circuit synthesis for dimension-reducible Boolean functions, demonstrating through experiments that the proposed strategy enables the computation of compact quantum circuits for D-reducible functions. This results in area reductions (measured by the number of elementary quantum gates) of approximately 38% from standard ESOP forms. Furthermore, when considering XAG-based quantum compilation [91], experimental findings indicate an area gain (measured in T-gates) of about 21%. It is noteworthy that XAG-based quantum compilation typically yields highly compact implementations. The outcomes of this research have been published at the DSD Conference 2023 [27]. **Chapter 4** builds on this by investigating the synthesis of autosymmetric and dimension-reducible functions, emphasizing improvements in circuit compactness and cost efficiency. **Chapter 5** explores the application of PSOP decomposition techniques, focusing on the challenges of optimizing gate count and circuit depth for quantum circuit synthesis. This chapter details the proposed pre-processing method and the reconstruction strategy, demonstrating that after PSOP decomposition and quantum synthesis of the components, it is possible to reconstruct the original function. The experimental results indicate that the proposed pre-processing phase exhibits better outcomes for 61% of the benchmarks, with an average gain of about 22% in terms of T-gates. The result of this research has been published at the DSD Conference 2024 and invited for a special issue in the WiPiEC Journal-Works in Progress in Embedded Computing Journal [26].

Part II: Multivalued Reversible Designs for Quantum Circuits, where we transition to multi-valued logic, beginning with **Chapter 6**, where we introduce the fundamental concepts of multi-valued reversible logic, with focus on ternary and quaternary logic. In **Chapter 7** we present innovative designs for a quantum ternary decoder, multiplexer, and demultiplexer, highlighting the benefits of the proposed ternary logic in minimizing circuit width and improving performance. Specifically, the enhancements in quantum cost, garbage outputs, and constant inputs for the proposed multiplexers are 48%, 4%, and 9%, respectively. Moreover, the newly designed demultiplexer shows improvements in quantum cost (48%), garbage outputs (7%), and constant inputs (5%) when compared to its predecessor [68]. The outcomes of this research were published in the Quantum Information Processing journal [135]. **Chapter 8** introduces a novel quantum ternary image processing circuit that utilizes quantum ternary gates, with a focus on reducing quantum cost while enhancing functionality. Notably, our design exhibits a 20.56% reduction in quantum cost, along with 11% and 100% improvements in constant inputs and garbage outputs, respectively, compared to its counterpart in [42]. The outcomes of this research were published in the Quantum Information Processing journal [134]. **Chapter 9** discusses the development of a balanced ternary reversible comparator, demonstrating significant advancements in quantum cost and efficiency. Specifically, the proposed balanced ternary 1-qutrit comparator shows substantial improvements in quantum cost (65%), the number of constant inputs (50%), and garbage outputs (33%) when compared to existing unbalanced comparator designs [41, 103, 147]. **Chapter 10** addresses quaternary reversible circuit optimization for multiplexers and demultiplexers, which are crucial for arithmetic logic units. Our proposed designs exhibit reduced quantum cost, garbage outputs, and constant inputs compared to previous designs in [59, 72, 74]. The result of this research were published in the IEEE Access journal [102].

Finally, **Chapter 11** concludes the thesis by illustrating how the proposed approaches address significant challenges in quantum computing, including circuit size, quantum cost, and scalability. This contribution enhances practical applications in quantum computing and sets the stage for future advancements in quantum circuit design.

Part I

Exploiting Function Regularities and Decomposition for Quantum Synthesis

Chapter 2

Preliminaries on Decomposition and Quantum Compilation

2.1 Function Regularities

2.1.1 The Concept of Dimension Reducible Functions

In this section, we explore the concept of dimension reducibility (*D-reducibility*), a particular regularity based on *affine spaces*[38, 39], which has been detailed in [17, 15, 16]. Recall that a vector subspace V of the classical Boolean vector space $(\{0,1\}^n, \oplus)$ is a subset of $\{0,1\}^n$ containing the zero vector $\mathbf{0} = (00\dots0)$, such that for each v_1 and v_2 in V we have that $v_1 \oplus v_2$ is still in V. We have the following:

Definition 1 *Let $\alpha \in \{0,1\}^n$ be a Boolean point (or vector), the set $A = \alpha \oplus V = \{\alpha \oplus v \mid v \in V\}$ is an* affine space *over V with* translation point *α.*

Example 1 *In Figure 2.1, columns correspond to the variables $x_1, x_2, \cdots x_6$, and the set $A = \alpha \oplus V$ is an* affine space *over the vector space $V = \{000000, 001001, 010010, 011011, \dots, 111111\}$, with translation point $\alpha = 000001 \in \{0,1\}^6$. The dimension of A is the dimension of the corresponding vector space V and α is a point in $\{0,1\}^6$. As shown in the figure, the vector space $V = \{000000, 001001, 01 0010, 011011, \dots, 111111\}$ and the vector $000001 \in \{0,1\}^6$. The set $A = \alpha \oplus V = 000001 \oplus V = \{000001, 001000, 010011, 011010, \dots, 111110\}$ is an* affine space *over V.*

Definition 2 *Consider an affine space A over a vector space V. The canonical translation point α_A is the minimum point of A in binary order.*

Definition 3 *Consider a vector space V of dimension k, sorted in binary order from index 0 to $2^k - 1$. The canonical basis B_V of V consists of the vectors with indices $2^0, 2^1, \dots, 2^{k-1}$.*

Example 2 *In the vector space V shown in Figure 2.1, rows are indexed from 0 to $2^k - 1$ and the points with indices $2^0, 2^1, 2^2, ..., 2^{k-1}$ (k=3) form the* canonical basis *B_A of V. The* canonical translation point *α_A is the point of A with index 0. Generally, the canonical representation of an* affine space *is given by its canonical*

A	V
$x_1x_2x_3x_4x_5x_6$	$x_1x_2x_3x_4x_5x_6$
0 0 0 0 0 1	0 0 0 0 0 0
0 0 1 0 0 0	0 0 1 0 0 1
0 1 0 0 1 1	0 1 0 0 1 0
0 1 1 0 1 0	0 1 1 0 1 1
1 0 0 1 0 1	1 0 0 1 0 0
1 0 1 1 0 0	1 0 1 1 0 1
1 1 0 1 1 1	1 1 0 1 1 0
1 1 1 1 1 0	1 1 1 1 1 1

Figure 2.1: An *affine space* A and the corresponding vector space V.

translation point and its canonical basis. In each canonical basis vector, the variable corresponding to the first 1-component from left is called canonical variable. *The variables that are not canonical in the canonical basis are called* non-canonical variables.

Example 3 *Consider the affine space A in Figure 2.1. We have that* $\alpha_A = 000001$ *and* $B_A = \{001001, 010010, 100100\}$*, and the canonical variables in* v_1, v_2*, and* v_3 *are equal to* x_3, x_2*, and* x_1*, respectively. Therefore, the non-canonical variables are* x_4, x_5*, and* x_6*.*

An *affine space* can be algebraically represented by a *pseudoproduct* consisting of an AND of XORs or literals [38]. There are several ways to express *affine space* characteristic functions as a pseudoproduct, out of them we use the *canonical expression* (CEX) [87].

The following definition describes how to derive the CEX expression from an *affine space* [38]:

- each non-canonical variables appears in exactly one XOR factor, and each XOR factor is composed by one non-canonical variable and possibly some canonical variables;

- the canonical variables that appear in the XOR factor corresponding to the non-canonical variable x are the canonical variables (if any) appearing with value 1 in the vectors of B_A where x is equal to 1;

- all the canonical variables are not complemented, a non-canonical variable is complemented in its XOR factor if and only if its corresponding component in α_A is 0.

Example 4 *In the* affine space *A the canonical translation point* α_A *is 000001 and the canonical basis* B_A *is* $\{001001, 010010, 100100\}$*. The first vector in* B_A *illustrates that the canonical variable is* x_3 *and the non-canonical variable is* x_6*, the second vector shows that* x_2 *is canonical variable and* x_5 *is the non-canonical variable, and* x_1 *and* x_4 *are the canonical and non-canonical variables, respectively, in the third vector in* B_A*. Also, according to the vector* α_A*,* x_5 *and* x_4 *are complemented while* x_6 *is not complemented. Therefore, the CEX expression for this example is equal to* $(x_3 \oplus x_6)(x_2 \oplus \bar{x}_5)(x_1 \oplus \bar{x}_4)$*.*

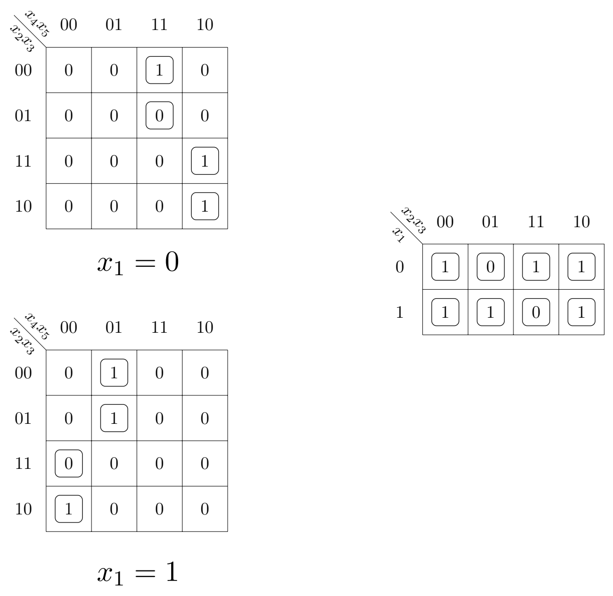


Figure 2.2: Karnaugh maps of a D-reducible function f (left) and its corresponding projection f_A (right).

Based on these definitions, we can now explain the regular functions that are considered *D-reducible*. Essentially, the points (or minterms) of *D-reducible* functions are entirely located within an *affine space* that is strictly smaller than the entire Boolean cube $\{0,1\}^n$. The formal definition of a D-reducible function is given below [17]:

Definition 4 *A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is* D-reducible *if $f \subseteq A$, where $A \subset \{0,1\}^n$ is an affine space with a dimension strictly smaller than n.*

When a function f is classified as *D-reducible*, it can be expressed as $f = \chi_A \cdot f_A$, where A represents the smallest affine space containing f and is referred to as the associated *affine space* of f. Additionally, χ_A denotes the characteristic function of A, while f_A signifies the projection of f onto A.

Example 5 *The Karnaugh map on the left side of Figure 2.2 illustrates the D-reducible function $f = \{00011, 01010, 01110, 10001, 10101, 11000\}$. According to the Karnaugh map on the right side of the figure, the new function f_A depends on three variables x_1, x_2, x_3. It should be noted that although the number of the onset minterms is the same for both f and f_A, they are compacted in a smaller space (map) in the f_A function. If we synthesize f and f_A in the classical SOP framework, we obtain f and f_A equal to $\overline{x}_1x_2x_3\overline{x}_4 + x_1\overline{x}_2\overline{x}_3x_4 + \overline{x}_1\overline{x}_2\overline{x}_3x_4x_5 + x_1x_2\overline{x}_3\overline{x}_4\overline{x}_5$ and $\overline{x}_3 + x_1\overline{x}_2 + \overline{x}_1x_2$, respectively. As a result, the overall number of products is reduced from 4 to 3, and the overall number of literals is reduced from 18 to 5. Moreover, the canonical basis can be derived in polynomial time exploiting the Gauss-Jordan elimination as described in [85]. Finally, a more compact form for the function f can be derived as $(x_1 \oplus x_4)(x_2 \oplus x_5)(\overline{x}_3 + x_1\overline{x}_2 + \overline{x}_1x_2)$, where $(x_1 \oplus x_4)(x_2 \oplus x_5)$ is the CEX representing the affine space χ_A.*

2.1.2 The Concept of Autosymmetric Functions

Another particular type of regularity can be observed in autosymmetric functions based on affine spaces, which is also based on affine space [19, 20, 87]. Informally, the idea behind this concept is to call k-autosymmetric any Boolean function f depending on n variables that can be simplified to a smaller function f_k, depending only on $n-k$ variables. Following this idea, the degree of regularity of a Boolean function f is determined by its *autosymmetry degree* k, where $0 \leq k \leq n$. If $k=0$, the function has no regularity. However, if $k \geq 1$, the function is identified as autosymmetric. Generally, an autosymmetric function f can be expressed through its *restriction function* f_k as follows:

$$f(x_1, x_2, \ldots, x_n) = f_k(y_1, y_2, \ldots, y_{n-k})$$

where the variables $y_1, y_2, \ldots, y_{n-k}$ are XOR combinations of subsets of the original input variables x_i. These combinations are denoted as $\mathrm{XOR}(X_i)$, where $X_i \subseteq \{x_1, x_2, \ldots, x_n\}$. The equations:

$$y_i = \mathrm{XOR}(X_i), \quad i = 1, \ldots, n-k$$

are termed as *reduction equations*. Thus, the autosymmetry test involves determining the value of k, the restriction f_k, and the reduction equations.

It is worth mentioning that autosymmetric functions depend in general on all their input variables, i.e., they are non-degenerate. However, degenerate functions, which do not depend on all their variables, are always classified as autosymmetric.

It is possible to synthesize the function f through its restriction f_k. This reconstruction requires an additional logic level of XOR gates, which take the original variables as inputs and generate the new $n-k$ variables that serve as inputs to the circuit for f_k. Typically, the function f_k can be synthesized using different techniques for any logic minimization problem. For instance, in [14, 18, 21], an XAG representation of the restricted function has been derived to improve the evaluation of multiplicative complexity and in [13] this representation has been leveraged to design compact reversible and quantum circuits.

As demonstrated in [20, 19], any k-autosymmetric function f has a corresponding k-dimensional vector space L_f, which contains all minterms α such that $f(x) = f(x \oplus \alpha)$ for all $x \in \{0,1\}^n$. The vectors in L_f are sorted by increasing binary order and indexed from 0 to $2^k - 1$. The subset of vectors indexed by $2^0, 2^1, \ldots, 2^{k-1}$ is termed the canonical basis B_L of L_f. There are k fully independent variables in L_f, which are known as *canonical variables* and assume all the possible combinations of $0, 1$ values in the vectors of the vector space L_f. Conversely, *noncanonical variables* in L_f, which are not canonical, assume either a constant value or are formed as linear combinations of the canonical variables. In the canonical basis B_L of L_f, canonical variables are those corresponding to the first 1-component from the left. In order to define the function f in its compact form f_k, these variables are essential.

The restriction f_k represents a more compact form of f, containing only $|S(f)|/2^k$ minterms, where $|S(f)|$ represents the total number of minterms in f. As discussed in [13], the reversible synthesis of f can be simplified to the synthesis of its restriction f_k, which can be determined in time polynomial in the dimension of some standard representations of the input function f, as for example a Reduced Ordered

$x_2x_3 \backslash x_4x_5$	00	01	11	10
00	0	0	$\boxed{1}$	0
01	0	0	0	0
11	0	0	0	$\boxed{1}$
10	0	0	0	$\boxed{1}$

$x_1 = 0$

$x_2x_3 \backslash x_4x_5$	00	01	11	10
00	0	$\boxed{1}$	0	0
01	0	$\boxed{1}$	0	0
11	0	0	0	0
10	$\boxed{1}$	0	0	0

$x_1 = 1$

$y_1y_2 \backslash y_3y_4$	00	01	11	10
00	0	0	$\boxed{1}$	0
01	0	0	0	0
11	0	0	0	$\boxed{1}$
10	0	0	0	$\boxed{1}$

Figure 2.3: Karnaugh maps of an 1-autosymmetric function f (left) and its corresponding projection f_k (right).

Binary Decision Diagram (ROBDD) representation [19]. The formal definition of a autosymmetric function is given below [22]:

Definition 5 *A Boolean function f is k-autosymmetric, $0 \leq k \leq n$, if its vector space L_f has dimension k. The function f is autosymmetric if it is k-autosymmetric with $k \geq 1$.*

Example 6 *Figure 2.3 illustrates the Karnaugh map of the function $f = \{00011, 01010, 01110, 10001, 10101, 11000\}$. According to the explanation of the algorithm given in reference [19], which details the process to determine the vector space L_f associated to f, we get $L_f = \{00000, 11011\}$. We can also determine the autosymmetry degree by calculating $\log_2 |L_f|$. In this case, $k = 1$, which indicates that the function f is 1-autosymmetric. This means that all the points in the function f can be projected in the space $\{0,1\}^4$ with respect to the reduction equations y_1, y_2, y_3 and y_4. Consequently, the restricted function f_1 is a more compact representation form of the function f with only 3 minterms which is shown in the right side of the figure. In this example the canonical basis is 11011, which designates x_1 as the canonical variable while x_2, x_3, x_4 and x_5 are non-canonical variables. Therefore we can obtain the reduction equations, which are $y_1 = x_1 \oplus x_2$, $y_2 = x_3$, $y_3 = x_1 \oplus x_4$, and $y_4 = x_1 \oplus x_5$. The restricted function f_1 shows a simplified form of the original function f, having only 3 minterms, which are $\{0011, 1010, 1110\}$, as can be observed in the Karnaugh map on the right side of the figure.*

2.2 PSOP Decomposition

2.2.1 Projections of Functions

In this section, we review some fundamental concepts of Boolean space partitioning and we show the projections exploited in this specific decomposition approach.

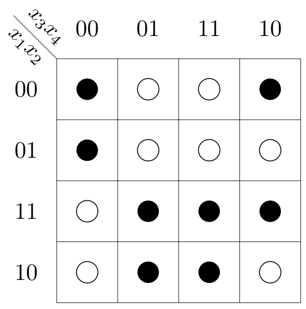


Figure 2.4: The Boolean space $\{0,1\}^4$ partitioned into the two distinct sets $B_{x_1=p}$ (black points) and $B_{x_1 \neq p}$ (white points), with $p = x_2.x_3 + x_4$.

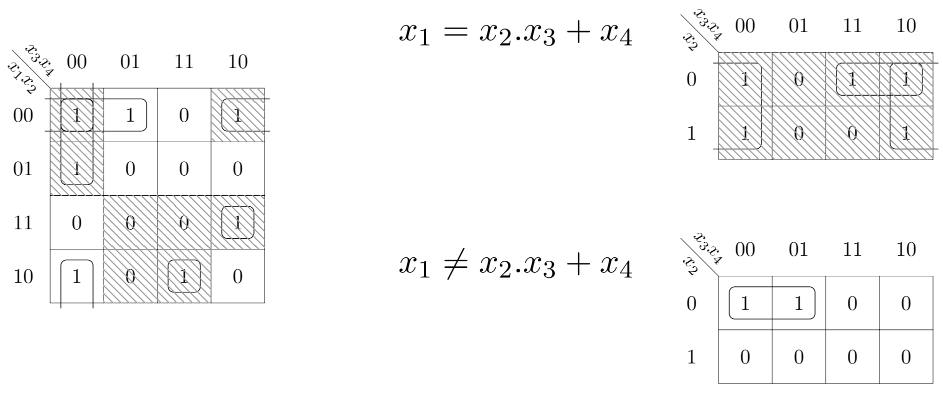


Figure 2.5: Karnaugh maps of a function f (left) and its corresponding projections onto $f|_{x_1=p}$ and $f|_{x_1 \neq p}$ (right), with $p = x_2.x_3 + x_4$.

Let us consider the Boolean space defined by variables $x_1, x_2, \ldots, x_n$, where x_i represents one of these variables. Suppose p denotes a function over a subset of variables excluding x_i, denoted by $\{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\}$. The Boolean space $B = \{0,1\}^n$ can then be partitioned into two distinct subsets: the set $B_{x_i=p}$, where x_i equals the function p, and the set $B_{x_i \neq p}$, where x_i is not equal to the function p. Formally, we have $B_{x_i=p} = \{(v_1, \ldots, v_n) \in \{0,1\}^n \mid v_i = p(v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n)\}$, and $B_{x_i \neq p} = \{(v_1, \ldots, v_n) \in \{0,1\}^n \mid v_i \neq p(v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n)\}$, respectively.

Example 7 *Let us consider the function $p = x_2.x_3 + x_4$ and the variable x_1 in the Boolean space $\{0,1\}^4$, which can be partitioned into two sets. The first set consists of the subspace where $x_1 = p$ and the second one consists of the subspace where $x_1 \neq p$. Figure 2.4 depicts a Karnaugh map illustrating these two sets, the black points correspond to $B_{x_i=p} = \{0000, 0010, 0100, 1001, 1011, 1101, 1110, 1111\}$, while the white points correspond to $B_{x_i \neq p} = \{0001, 0011, 0101, 0110, 0111, 1000, 1010, 1100\}$.*

It is important to note that the Boolean space partitions evenly into these two sets, showing the following general property [25]: When x_i is a Boolean variable and p is a function represented as $p : \{0,1\}^{n-1} \to \{0,1\}$ on variables $\{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\}$, the sets $B_{x_i=p}$ and $B_{x_i \neq p}$ are such that:

1. $B_{x_i=p} \cup B_{x_i \neq p} = \{0,1\}^n$
2. $|B_{x_i=p}| = |B_{x_i \neq p}| = 2^{n-1}$
3. $B_{x_i=p} \cap B_{x_i \neq p} = \emptyset$

In the Boolean space $B = \{0,1\}^n$, the most straightforward partitioning occurs when p equals 1 (or 0). In this context, $B_{x_i=1}$ and $B_{x_i \neq 1}$ denote the subspaces of B where x_i equals 1 and x_i equals 0, respectively. These subspaces can be characterized using functions x_i and $\overline{x}_i$, as discussed in [12]. Note that $B_{x_i=1}$ and $B_{x_i \neq 1}$ correspond to the fundamental partitions in the *classical Shannon decomposition*

$$f = x_i f|_{x_i=1} + x_i f|_{x_i \neq 1} \,,$$

where $f|_{x_i=1}$ and $f|_{x_i \neq 1}$ denote the two cofactors of f obtained by setting x_i with 1 and 0, respectively.

In [29, 67], a Boolean functional decomposition method is introduced, generalizing the classical Shannon decomposition as follows

$$f = (\overline{x}_i \oplus p) f|_{x_i=p} + (x_i \oplus p) f|_{x_i \neq p} \,.$$

This method projects the function f onto the two complementary subsets $B_{x_i=p}$ and $B_{x_i \neq p}$ of the Boolean space $B = \{0,1\}^n$. The expressions $\overline{x}_i \oplus p$ and $x_i \oplus p$ represent the characteristic functions of $B_{x_i=p}$ and $B_{x_i \neq p}$, respectively. It is noteworthy that that the Shannon decomposition is a specific instance of this partitioning method, where $x_i \oplus 1$ equals $\overline{x}_i$ and $\overline{x}_i \oplus 1$ equals x_i.

Example 8 *Figure 2.5 illustrates the Karnaugh map of the function* $f = \overline{x}_1\overline{x}_2\overline{x}_3\overline{x}_4 + \overline{x}_1\overline{x}_2\overline{x}_3x_4 + \overline{x}_1\overline{x}_2x_3\overline{x}_4 + \overline{x}_1x_2\overline{x}_3\overline{x}_4 + x_1\overline{x}_2\overline{x}_3\overline{x}_4 + x_1\overline{x}_2x_3x_4 + x_1x_2x_3\overline{x}_4$, *in the right side. Consider* $B_{x_1=p}$ *and* $B_{x_1 \neq p}$ *as two projecting sets with* $p = x_2.x_3 + x_4$. *As shown in the left side of this figure, the function* f *can be projected onto the two spaces* $B_{x_1=p}$ *and* $B_{x_1 \neq p}$. *The resulting projected functions depend on* x_2, x_3, x_4 *and can be represented by* $f|_{x_1=p} = \overline{x}_2\overline{x}_3\overline{x}_4 + \overline{x}_2x_3\overline{x}_4 + \overline{x}_2x_3x_4 + x_2\overline{x}_3\overline{x}_4 + x_2x_3\overline{x}_4$ *and* $f|_{x_1 \neq p} = \overline{x}_2\overline{x}_3\overline{x}_4 + \overline{x}_2\overline{x}_3x_4$, *respectively.*

It is crucial to note that *Hamming distances* can vary when points are projected onto different subspaces. As a result, they may be merged into larger terms and may reveal new implications not apparent in the original function. For instance, consider the two points described in Example 8, represented by the minterms $\overline{x}_1\overline{x}_2\overline{x}_3x_4$ and $x_1\overline{x}_2\overline{x}_3\overline{x}_4$. Initially, their Hamming distance is equal to 2 (due to differences in the variables x_1 and x_4). After the projection onto the subspace where $x_1 \neq p$, these two points are represented by the minterms $\overline{x}_2\overline{x}_3\overline{x}_4$ and $\overline{x}_2\overline{x}_3x_4$. Now, their Hamming distance is reduced to 1 as they become more similar; thus it is possible to combine them into the larger product $(\overline{x}_2\overline{x}_3)$ in $f|_{x_1 \neq p}$. It should be noted that $(\overline{x}_2\overline{x}_3)$ is an implicant of $f|_{x_1 \neq p}$, but it is not an implicant of f directly.

2.2.2 PSOP Representation of Functions

The PSOP decomposition and synthesis approach involves creating a circuit for f by exploiting p and the two projected functions $f|_{x_i=p}$ and $f|_{x_i\neq p}$. Constructing circuits for these projected functions is generally simpler compared to f, as they involve one fewer variable, often resulting in more compact circuit designs. The functions p, $f|_{x_i=p}$ and $f|_{x_i\neq p}$ can be synthesized using various logic minimization techniques, including SOP synthesis, EXOR-based synthesis and also *quantum* synthesis methods. When expressed as sums of products, they form the *Projected Sum of Products* form, abbreviated as PSOP(f), as defined in [12].

Definition 6 *Let $f|_{x_i=p}$ and $f|_{x_i\neq p}$ denote the projections of f onto $B_{x_i=p}$ and $B_{x_i\neq p}$, respectively. The* PSOP *of f with respect to p is expressed as*

$$PSOP(f) = (\overline{x}_i \oplus p)f|_{x_1=p} + (x_i \oplus p)f|_{x_1\neq p}\,,$$

where p, $f|_{x_i=p}$, and $f|_{x_i\neq p}$ are expressed as SOP forms.

It should be highlighted that to achieve an overall minimal form, additional steps should be taken to reduce the SOP form for p along with the two projected SOP forms $f|_{x_i=p}$ and $f|_{x_i\neq p}$ after projection.

Definition 7 *Let $\tilde{p}$ be a minimal SOP form for the function p and let the* minimal SOP expressions *for the projections of f onto the sets $B_{x_i=p}$ and $B_{x_i\neq p}$ be represented by $\tilde{f}|_{x_i=p}$ and $\tilde{f}|_{x_i\neq p}$, respectively. The* minimal PSOP *of f with respect to p is expressed as:*

$$PSOP(f) = (\overline{x}_i \oplus \tilde{p})\tilde{f}|_{x_1=p} + (x_i \oplus \tilde{p})\tilde{f}|_{x_1\neq p}\,.$$

Example 9 *Consider the function f described in Example 8, the minimal SOP form of this function is $\overline{x}_1\overline{x}_2\overline{x}_3+\overline{x}_1\overline{x}_3\overline{x}_4+\overline{x}_1\overline{x}_2\overline{x}_4+\overline{x}_2\overline{x}_3\overline{x}_4+x_1\overline{x}_2x_3x_4+x_1x_2x_3\overline{x}_4$. As shown in Figure 2.5, the function f is projected onto the two sets. The minimal SOP forms for the sets $B_{x_1=p}$ and $B_{x_1\neq p}$, can be represented by $\tilde{f}_{x_1=p} = \overline{x}_4 + \overline{x}_2x_3$ and $\tilde{f}|_{x_1\neq p} = \overline{x}_2\overline{x}_3$, respectively. As the minimal SOP form of p is $\tilde{p} = x_2x_3 + x_4$, the overall minimal PSOP form for f is then $(\overline{x}_1 \oplus (x_2x_3 + x_4))(\overline{x}_4 + \overline{x}_2x_3) + (x_1 \oplus (x_2x_3 + x_4))(\overline{x}_2\overline{x}_3)$.*

Another valuable representation is the *Pr-SOP* (Projected Sum of Products) for the function f, which is also recognized as the *PSOP with remainder* [12]. This form includes a *remainder* component that encompasses all products present in the SOP expression of f intersecting both projection sets. These products are termed *crossing products*, while products entirely included in one of the two projection sets are referred to as *non-crossing products*.

Definition 8 *Let $f|_{x_i=p}$ and $f|_{x_i\neq p}$ denote the projections of all non-crossing products in a SOP representation of f, and let r denote the sum of all crossing products. The* Pr-SOP *of f with respect to p is expressed as:*

$$Pr\text{-}SOP(f) = (\overline{x}_i \oplus p)f|_{x_i=p} + (x_i \oplus p)f|_{x_i\neq p} + r\,.$$

Minimizing all SOP expressions, we derive a *minimal PSOP with remainder.*

Definition 9 *Let $\tilde{p}$ and $\tilde{r}$ be minimal SOPs form for the function p and the remainder r, respectively. Let the minimal SOP expressions for the projections of all non-crossing products of f onto $B_{x_i=p}$ and $B_{x_i\neq p}$ be represented by $\tilde{f}|_{x_i=p}$ and $\tilde{f}|_{x_i\neq p}$, respectively. The* minimal Pr-SOP *of f with respect to p is expressed as:*

$$\textit{Pr-SOP}(f) = (\overline{x}_i \oplus \tilde{p})\tilde{f}|_{x_1=p} + (x_i \oplus \tilde{p})\tilde{f}|_{x_1\neq p} + \tilde{r}\,.$$

Example 10 *Consider the Karnaugh map in Figure 2.5. The minterms $\overline{x}_1\overline{x}_2\overline{x}_3$ and $\overline{x}_2\overline{x}_3\overline{x}_4$ intersect both $B_{x_i=p}$ and $B_{x_i\neq p}$, it means that they are not fully contained within either projection set, making them crossing products. As a result, the* Pr-SOP *of f with respect to p is given by this expression:*

$$(\overline{x}_1 \oplus (x_2x_3 + x_4))(\overline{x}_4 + \overline{x}_2x_3) + (\overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_2\overline{x}_3\overline{x}_4)$$

Algorithms and heuristic approaches for minimizing PSOP expressions have been proposed and analyzed in [25, 28].

2.3 Quantum Synthesis

2.3.1 ESOP Forms

In the Exclusive-or Sum-of-Products (ESOP) form, a Boolean function is expressed using a series of AND gates at the first level and one multi-input XOR gate at the next level. ESOP forms continue to be of interest in research due to their important applications in various emerging technologies. Compared to standard SOP forms, ESOP provides several benefits, such as requiring fewer products to realize randomly generated functions [125, 126], more compact representations for arithmetic or communication circuits [127], higher testability properties [54, 66] and security [82, 98]. Furthermore, due to the inherent reversibility of the XOR operation [119], ESOP forms are essential for reversible logic circuits synthesis and quantum computing [47].

Recall that, for a Boolean function to be expressed in ESOP form, it must contain a set of product terms involving the input variables. In the off-set, each minterm should be covered by an even number of these product terms or not at all, whereas in the on-set, each minterm must be covered by an odd number of these product terms.

It is possible for a function to have multiple ESOP forms that are structurally different but semantically equivalent. The goal of ESOP minimization is to find the form with the lowest cost based on a given cost metric.

Example 11 *In Figure 2.2, the function f_A can be represented in ESOP form with 2 products and 3 literals as follows:*

$$ESOP(f_A) = 1 \oplus \overline{x}_2x_3 \oplus x_1x_3$$

ESOP minimization is a computationally very hard problem that has garnered significant interest of algorithm designers. due to its potential for reducing the overall costs of hardware realizations and software implementations. For this purpose, several heuristic and exact methods have been designed [119, 97, 115, 116, 122, 130].

Based on heuristic methods, although ESOP forms can be identified fast, only a subset of the possible search space is examined. It is also possible to produce small ESOP forms using these methods in reasonable time, but they suffer from local minima that are difficult to escape. In exact methods, while a minimal number of product terms can be found, more than 8 Boolean variables and a few product terms can be hardly handled. In [119] a novel approach for exact synthesis based on Boolean satisfiability is presented, which is particularly fast and unaffected by the number of Boolean variables involved.

2.3.2 Quantum Logic and Gates

The applications of quantum computing in number theory, encryption, search, and scientific computation make it one of the most promising emerging computing paradigms [151, 37, 146].

In a binary quantum system, the unit of information (memory) is termed as qubit (quantum bit). There are two basis states for a qubit which are depicted by $|0\rangle$ and $|1\rangle$, corresponding to the states 0 and 1 for a classical bit. Each of these states are called a qubit state and can be represented by 2×1 matrix as below:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

In addition, a qubit can be defined as linear combination of the states, this technique is known as superposition. The following equation defines the notation for the superposition (ψ):

$$\psi = \alpha|0\rangle + \beta|1\rangle$$

where α and β are complex quantities that represent probability amplitudes of basis states, and ψ is the wave function. The probability measurement of the occurrences for states $|0\rangle$ and $|1\rangle$ are equal to $|\alpha|^2$ and $|\beta|^2$, respectively. The sum of these probabilities is described with following notation:

$$|\alpha|^2 + |\beta|^2 = 1$$

Quantum computation can be performed with the composition of elementary quantum operations or gates in a quantum system. In the quantum circuit, there is a small library of gates that interact with one or two qubits.

2.3.3 Reversible Circuits and Quantum Compilation

The *reversible circuit* ensures one-to-one correspondence between each input vector and each output vector. Accordingly, the number of inputs equals the number of outputs. In these circuits, constant inputs are called *ancilla inputs*, while *garbage outputs* are produced to maintain the one-to-one mapping although their specific values are not important. Generally, reversible circuits are constructed from a sequence of Mixed-Polarity Multiple-Control (MPMC) Toffoli gates, which are able to implement any logic function.

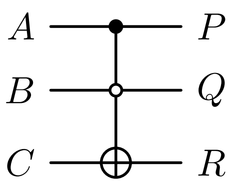


Figure 2.6: The symbolic representation of MPMC Toffoli gates.

Figure 2.6 illustrates MPMC Toffoli gates: the conventional symbol $\oplus$ represents the target line, while control lines are indicated by $\bullet$ for positive connections and $\circ$ for negative connections. If there are no control connections on the MPMC Toffoli gate, it operates as a NOT gate; with a positive control connection, it functions as a C-NOT gate; and when there are only positive connections, it functions as a Multiple-Control Toffoli gate[110].

More formally, given a set of circuit lines $X = x_1, x_2, ..., x_n$, an *MPMC Toffoli gate* $T(C, t)$ has *control lines* $C = \{x_{j1}, x_{j2}, ..., x_{jc}\} \subset X$ and a *target line* $t \in X \backslash C$. The gate maps $t \rightarrow t \oplus (x_{j1}^{p1} \wedge x_{j2}^{p2} \wedge \ \ldots \ \wedge x_{jl}^{pl})$, where each literal x_{ji}^{pi} is either a propositional variable $x_{ji}^{1} = x$ or its negation $x_{ji}^{0} = \bar{x}$. All remaining other lines are passed through unaltered. Note that, whenever t is equal to 0, the MPMC Toffoli gate computes the AND of all control lines on the target line.

A common approach in reversible logic design is ESOP-based synthesis, which is based on the relationship between product terms in ESOP (Exclusive Sum of Products) forms and MPMC Toffoli gates [47, 45, 58]. In an ESOP expression, MPMC Toffoli gates are derived from product terms, with their literals serving as control lines. The circuit effectively computes the XOR of all product terms on the same target line because all gates act on the same target line.

As a preliminary step towards building quantum circuits, reversible circuit synthesis plays an important role in quantum computing. A Quantum compilation step is then required to convert a reversible circuit containing MPMC Toffoli gates into a quantum circuit. In this process, each reversible gate can be decomposed into basic quantum gates, following a specified quantum gate library [89, 94]. In this thesis, we use the Clifford+T library, which includes Pauli, Hadamard, CNOT gates, and the T gate, which is considered the most expensive one [110]. It is worth mentioning that design of quantum logic gates and circuits by using this set makes them fault tolerant with error correcting codes [3, 95].

- X gate: This gate is a 1-qubit operation gate and complements the state of a qubit. Generally, this gate is equivalent to a classical NOT gate. The corresponding unitary matrix for this gates is shown below:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

- S gate: The *phase gate* (S gate) is a 1-qubit operation in Clifford+T gate set and represents a 90-degree rotation around the z-axis, the corresponding matrix for this gate is:

$$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$$

- H gate: The *Hadamard gate* is one of the 1-qubit operations in the set of Clifford+T gate and often used to create superposition. In this gate, the state $|0\rangle$ transforms to $(|0\rangle + |1\rangle)/\sqrt{2}$ and the state $|1\rangle$ transforms to $(|0\rangle - |1\rangle)/\sqrt{2}$. The corresponding unitary matrix for this gate is shown as below:

$$H = \tfrac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

- T gate: This 1-qubit gate is part of the Clifford+T gate set, which is universal for quantum computation. The corresponding unitary matrix for the T gate is given by:

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{bmatrix}$$

- C-NOT gate: The *Controlled-NOT* (C-NOT) is the only 2-qubit operation in the set of Clifford+T gate. In this gate, the state of one qubit called *target* is complemented according to the state of the other qubit called *control*, the corresponding matrix is:

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Table 2.1 reports the classical cost in terms of Hadamard, $CNOT$s, T gates, and ancillary qubits of the realization of k-controlled MPMC Toffoli gates with the algorithm described in [88].

Table 2.1: The cost of k-controlled Toffoli gates in number of T, H and $CNOT$ gates.

k	T	H	$CNOT$	ancillary qubits
2	7	2	6	0
3	16	6	14	1
≥ 4	8k-8	8k-12	4k-6	$\lceil \frac{k-2}{2} \rceil$

Recent advancements in quantum compilation techniques have led to more efficient circuit designs. In [91], for instance, a novel method described and focused on compiling quantum circuits representations from XAG (XOR-And Graph) representations, reducing the number of T gates required for implementing Toffoli gates. [120] and [144] provide more detailed information on reversible circuits and effective quantum compilation strategies.

Chapter 3

Quantum Synthesis of Dimension Reducible Functions

This chapter introduces a novel approach to quantum oracles by demonstrating the effective generation of reversible circuits. Classical methods, which use less compact reversible circuits, often result in higher quantum costs. In response, this study proposes a new technique that streamlines the synthesis process by leveraging the inherent regularities of dimension-reducible Boolean functions. The results of the proposed approach show that it generates significantly more compact circuits, hence reducing the overall cost. The experimental results also demonstrated that the effectiveness of the proposed approach leads to considerable area savings and cost reduction in the quantum circuits.

3.1 Introduction

As stated in the previous chapter, reversible logic synthesis and quantum compilation are two fundamental steps in the synthesis of quantum oracles. Since the evolution of quantum systems is governed by reversible unitary operators, classical logic functions must first be synthesized into reversible circuits. These circuits are then converted into quantum circuits by decomposing the reversible gates into sequences of basic quantum gates using a predefined gate library. In recent times, several new approaches for reversible circuit synthesis and quantum compilation have been proposed in the literature. One such method, discussed in [22], utilizes a structural regularity called *autosymmetry* to synthesize compact quantum circuits.

This chapter introduces a novel approach for quantum synthesis of Boolean functions that exhibit a different structural regularity known as *Dimension reducibility, or D-reducibility.* D-reducible functions are characterized by their minterms being confined to an affine subspace that is smaller than the full Boolean cube. These functions are notably prevalent in classical benchmarks, with the experimental results in [17] indicating that approximately 70% of functions the classical ESPRESSO benchmark suite [145] have at least one D-reducible output. Additionally, the decomposition of D-reducible functions can be computed efficiently in polynomial time, making them an attractive target for quantum synthesis.

The proposed approach utilizes D-reducibility to design more compact quantum circuits. Circuits optimized for quantum compilation can be obtained by the quantum synthesis process by structurally decomposing these functions. A major benefit of this approach is that the resulting reversible circuits have an uncomputing component that relies solely on CNOT gates instead of T-gates, which are more resource-intensive and do not require extra input lines. The experimental results demonstrate the effectiveness of this approach, achieving reductions in circuit area measured by the number of basic quantum gates of approximately 38% compared to standard ESOP forms. When applied to XAG-based quantum compilation [91], the approach achieves a 21% reduction in T-gates, demonstrating its potential for generating efficient quantum circuits. The outcomes presented in this chapter have been published at the DSD Conference 2023 [27].

The chapter is organized as follows. Section 3.2 describes our proposed new methodology for construction of reversible circuit for dimension reducible functions. Section 3.3 shows the evaluation of the proposed decomposition and reports our experiments on a set of benchmarks. Finally, the conclusion of this work is given in Section 3.4.

3.2 Reversible Circuits for D-Reducible Functions

As already reviewed in the previous section, there is a natural correspondence between ESOP expressions and reversible circuits, based on the fact that product terms in an ESOP expression can be easily represented with a reversible MPMC Toffoli gate. Thus, given an ESOP expression, one can extract a sequence of MPMC Toffoli gates whose control lines correspond to the literals in the product terms of the ESOP form, and whose target line corresponds to the output of the function. Notice that all gates act on the same target line, thus realizing the exclusive OR sum of all product terms. The main problem of this approach is due to the fact that MPMC Toffoli gates are only an intermediate representation. They need to be mapped into elementary quantum gates using an additional synthesis step, whose goal is to transform the reversible circuit of MPMC Toffoli gates into a functionally equivalent quantum circuit implementation. Unfortunately, this mapping step might be very onerous, especially for MPMC Toffoli gates controlled by many variables, thus leading to quantum circuits of high size and depth.

In this section, we therefore propose to exploit the regularity of D-reducible functions to ease their reversible synthesis, in order to obtain a final quantum circuit of reduced size and depth. Given a D-reducible function $f = \chi_A \cdot f_A$, the idea is to concatenate two reversible circuits: a circuit computing the characteristic function χ_A of the affine space A and a reversible circuit implementing the projection f_A. Since the characteristic function χ_A may depend on all input variables, the inputs of the circuit for χ_A are all variables: non-canonical and canonical ones. The circuit for f_A, instead, depends only on the canonical variables. To compute f, we then need a final Toffoli gate computing the AND between the two subfunctions χ_A and f_A, with f as target line. Note that this approach requires three additional lines (and therefore 3 new qubits in the quantum implementation of the reversible circuit for f): one line for χ_A, one for f_A, and finally one for f. The overall circuit structure is shown in Figure 3.1.

Let us now discuss how to implement the circuits for χ_A and f_A. The circuit for f_A can be derived from an ESOP representation of the projection f_A. Since

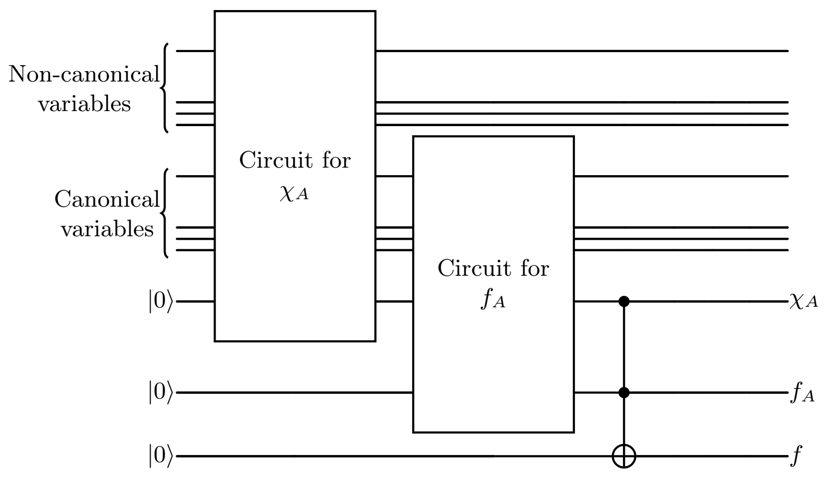


Figure 3.1: A reversible circuit for a D-reducible function based on the decomposition $f = \chi_A \cdot f_A$.

f_A depends only on the canonical variables that are a subset of the input variables, we can reasonably expect that its ESOP representation will contain product terms with less literals. Therefore, the corresponding MPMC Toffoli gates will be controlled by less input lines, thus leading to a quantum circuit of smaller size and depth after the quantum compilation with respect to the *Clifford+T* library. A reversible circuit for χ_A can be implemented directly from its canonical CEX expression. Indeed, recall from Section 2.1.1 in Chapter 2, a CEX consists of an AND of XOR factors, and each XOR factor can be implemented using CNOT gates. In particular, a XOR factor of k literals can be implemented using $k-1$ CNOTs. Finally, an MPMC Toffoli gate is required to implement the AND of all XOR factors. The number of control lines in input to this gate corresponds to the number of XOR factors, and therefore to the number of non-canonical variables of the affine space A. Interestingly, we do not need to introduce new input lines to represent each XOR factor, as proved in the following proposition [27].

Proposition 1 *Let* $\chi_A : \{0,1\}^n \to \{0,1\}$ *be the canonical CEX expression representing an affine subspace* A *of* $\{0,1\}^n$*. Then, a reversible circuit for* χ_A *can be implemented without adding new input lines.*

Proof. An affine subspace $A \subseteq \{0,1\}^n$ can be defined by linear equations that relate the input variables. The canonical CEX expression χ_A, which representing an affine subspace A of $\{0,1\}^n$, uses only XOR and AND operations. These operations can be implemented directly using reversible gates like CNOT and Toffoli gates without adding extra input lines. ■

Observe that the modification of the non-canonical variables has no effect on the successive calculation of the projection f_A, as f_A depends only on the canonical variables. Once the overall function f has been computed on the output line (the last line in Figure 3.1), we can restore the non-canonical variables to their initial values applying the so-called *uncomputing procedure* [110]: we uncompute the XOR factors stored in the non-canonical variables by re-applying the CNOTs in reverse order. This disentangles the variables, reverting them to their initial values. The overall methodology is summarized in the algorithm in Figure 3.2.

Algorithm 1 Reversible synthesis of D-reducible functions

```
1: INPUT:
2:     $f$                    ▷ D-reducible Boolean function with affine space $A$
3:     $f_A$                  ▷ Projection of $f$ onto $A$
4:     $\chi_A$               ▷ Characteristic function of $A$
5: OUTPUT:
6:     $Q$                    ▷ Quantum circuit for $f$
7: $Q_A \leftarrow$ ReversibleSynthesis($f_A$)
8: $Q_{\chi_A} \leftarrow$ ReversibleSynthesis($\chi_A$)
9: $Q \leftarrow$ Toffoli($Q_A, Q_{\chi_A}$)
10: $Q \leftarrow$ UncomputingProcedure($Q$)
11: return $Q$
```

Figure 3.2: Reversible synthesis of D-reducible functions.

Example 12 *Consider the function $f = x_1x_2\overline{x}_3x_4 + x_1\overline{x}_2x_3$, as represented on the Karnaugh map shown on the left side of Figure 3.3. This function is D-reducible and can be projected onto $f_A = \overline{x}_2 + x_2x_4$, with canonical variables x_2 and x_4, as shown on the Karnaugh map shown on the right side of Figure 3.3. It can be decomposed as follows $f = x_1(x_2 \oplus x_3)(\overline{x}_2 + x_2x_4)$, where $\chi_A = x_1(x_2 \oplus x_3)$ and $f_A = \overline{x}_2 + x_2x_4$. To derive a reversible circuit of f exploiting this decomposition, we first need to represent f_A in ESOP form. For this example, since the two products $\overline{x}_2$ and x_2x_4 are disjoint, we can immediately derive an ESOP form simply replacing the OR operator with a XOR: $f_A = \overline{x}_2 \oplus x_2x_4$. To implement the reversible computation of χ_A we only need a CNOT for computing $x_2 \oplus x_3$ onto the line corresponding to the non-canonical variable x_3, and a Toffoli gate for computing the AND between the first factor x_1 and the factor $x_2 \oplus x_3$ (first and second gate in Figure 3.4). The projection f_A can be computed with two gates corresponding to the two terms in its ESOP form (third and forth gate in Figure 3.4). Then, the next Toffoli gate computes the AND between χ_A and f_A. Finally, the last CNOT gate implements the uncomputing procedure. Figure 3.5 shows a reversible circuit derived from the ESOP representation $f = x_1x_2\overline{x}_3x_4 \oplus x_1\overline{x}_2x_3$ of the same function. This circuit contains only two MPMC Toffoli gates, and may appear more convenient than the one exploting the D-reducibility properties. However, the first circuit contains Toffoli gates with at most two control variables, while the gates in the second circuit are controlled by up to 4 variables. This has a very important impact on their quantum implementations. Indeed, if we compute the cost in terms of T, H, and CNOT gates of these two implementations (using the costs reported in Table 2.1, in Chapter 2) we can easily verify that the first circuits requires 21 T gates, 6 H gates and 21 CNOTs, leading to an overall size of* **48 gates**. *On the other hand, the second circuits has a cost of 40 T gates, 26 H gates and 24 CNOTs, and an overall size of* **90 gates**.

Finally, observe that the same strategy can be applied also when the two parts of the decompostion, i.e., f_A and χ_A, are represented in XAG form, as discussed in Section 3.3.

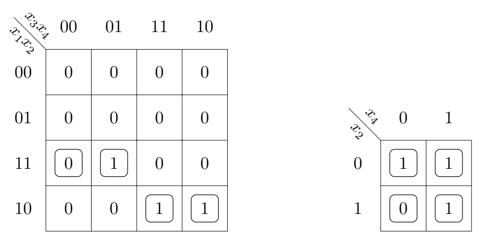


Figure 3.3: Karnaugh maps of a D-reducible function f (left) and its corresponding projection f_A (right).

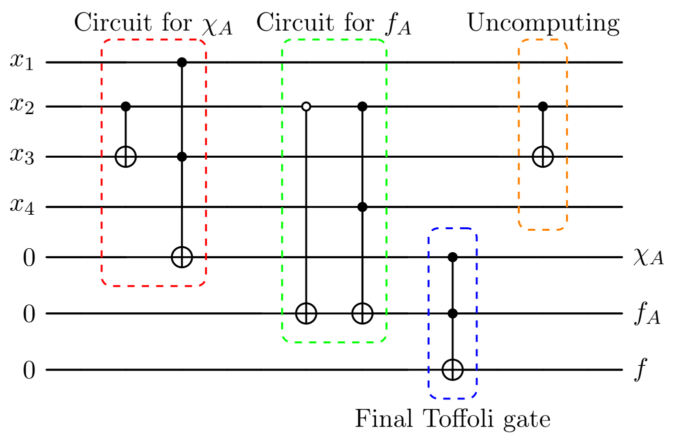


Figure 3.4: Reversible circuit, with the uncomputing procedure, for the D-reducible function f of Example 12, derived from its decomposition as $f = \chi_A \cdot f_A$. After quantum compilation with the *Clifford+T* library, the size of the circuit becomes equal to 48 elementary quantum gates.

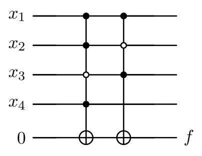


Figure 3.5: Reversible circuit for the D-reducible function f of Example 12, derived without exploiting the D-reducible decomposition. After quantum compilation with the *Clifford+T* library, the size of the circuit becomes equal to 90 elementary quantum gates.

3.3 Experimental Results

In this section we evaluate the proposed decomposition in the context of reversible circuit synthesis and quantum compilation. In particular, we conducted two different experimental evaluations. The first one, discussed in Section 3.3.1, aims at measuring to what extent the D-reducibility property can be exploited to imple-

Table 3.1: Comparison between quantum circuits computed without and with the D-reducible decomposition

		without decomposition				**with decomposition**				
Benchmark_output	**input**	**T**	**H**	**CNOT**	**ancillae**	**T**	**H**	**CNOT**	**ancillae**	**T gain**
b10_3	15	488	448	133	27	406	354	129	23	17%
dk48_2	15	512	492	200	30	110	92	45	5	79%
dk48_4	15	520	500	204	31	80	76	38	5	85%
gary_2	15	488	448	133	27	406	354	129	23	17%
gary_4	15	744	686	246	44	646	588	198	36	13%
in0_3	15	232	216	71	13	181	154	65	9	22%
in0_5	15	856	812	306	53	615	556	198	33	28%
in2_5	19	1520	1410	491	89	1118	1002	307	63	26%
in2_9	19	2144	2002	715	127	1878	1736	581	107	12%
in5_9	24	1568	1492	574	94	1423	1342	504	83	9%
m181_1	15	135	114	26	7	103	70	46	5	24%
newtpla_0	15	200	188	66	12	134	112	38	7	33%
newtpla_2	15	704	652	208	42	382	322	94	20	45%
rckl_6	32	1928	1862	817	120	1814	1748	759	106	6%
spla_21	16	752	724	298	45	142	120	56	7	81%
spla_32	16	1560	1492	592	93	1094	1024	387	64	30%
t2_6	17	494	452	165	29	287	262	91	17	42%
vg2_2	25	1152	1096	421	70	527	462	111	30	54%
vg2_6	25	520	492	182	30	231	182	64	11	56%
vtx1_5	27	4096	3984	1739	244	1287	1154	387	78	69%
Average Benchmark Suite		**379**	**352**	**126**	**22**	**236**	**204**	**80**	**13**	**38%**

ment compact circuits, in the context of standard reversible synthesis starting from ESOP forms. The aim of the second experimental evaluation, discussed in Section 3.3.2, is to establish whether more advanced quantum compilation methods could also benefit from the decomposition based on D-reducibility.

These two experimental evaluations have been conducted on D-reducible functions taken from the LGSynth'89 benchmark suite [145]. Since D-reducibility is a property of single outputs, we consider single outputs of the benchmark functions.

3.3.1 Standard Reversible Synthesis

Following the strategy depicted in Figure 3.1, we implement a reversible circuit for a D-reducible function $f = \chi_A \cdot f_A$. To better evaluate the quality of the reversible circuits derived for D-redicibile funtions, and to compare them with those derived without exploiting the D-reducibility property, we have measured their size in terms of elementary quantum gates. More precisely, instead of considering the overall number of MPMC Toffoli gates, we have mapped each Toffoli gate into elementary quantum gates, considering the Clifford+T low-level quantum gate library and the algorithm described in [88].

Table 3.1 reports a significant subset of benchmarks as representative indicators of our experiments. The first column reports the name of the benchmark and the number of the output. The following group of 3 columns reports the costs, in terms of elementary quantum gates, of the reversible circuits derived from minimal ESOP expressions of the benchmarks, without exploiting the D-reducibility structural regularity. The last group of columns reports the costs of the reversible circuits derived exploiting the D-reducibily decomposition, as explained in Section 3.2. ESOP minimization of the benchmarks and of their projections onto the associated affine spaces is performed using the EXORCISM-4 heuristic [97]. Due to the heuristic nature of this ESOP minimizer, the synthesis times for the functions and

Table 3.2: Comparison of the number of T gates in quantum circuits compiled without and with the D-reducible decomposition

Benchmark_output	**input**	**T gates without decomposition**	**T gates with decomposition**	**T gain**
b10_3	15	76	96	-26%
dk48_2	15	84	52	38%
dk48_4	15	76	40	47%
gary_2	15	84	96	-14%
gary_4	15	124	96	23%
in0_3	15	64	52	19%
in0_5	15	156	136	13%
in2_5	19	240	184	23%
in2_9	19	228	292	-28%
in5_9	24	152	168	-11%
m181_1	15	24	24	0%
newtpla_0	15	56	48	14%
newtpla_2	15	80	68	15%
rckl_6	32	120	120	0%
spla_21	16	96	56	42%
spla_32	16	324	88	73%
t2_6	17	68	44	35%
vg2_2	25	120	80	33%
vg2_6	25	68	64	6%
vtx1_5	27	184	116	37%
Average Benchmark Suite		**63**	**49**	**21%**

their projections are similar and very short, leading to negligible gain in synthesis time. Finally, the last column reports the gain in the number of T gates, and the last row reports the average costs for all the benchmarks considered in our experiments. The gain obtained synthesizing a reversible circuit exploiting this structural regularity is quite interesting. Indeed, the cost gain for T gates is about 38%, the cost gain for H gates is about 42%, the cost gain for CNOTs is about 37% (including the cost of the uncomputing procedure), and the gain in ancillary qubits is about 42%. Notice that the uncomputing procedure does not introduce new T gates.

3.3.2 XAG-Based Quantum Compilation

We now discuss the experimental results obtained by applying the quantum compilation heuristic proposed in [91]. In particular, we are interested in evaluating experimentally whether this recent technique could benefit from the decomposition of the target function based on the D-reducibility property.

The considered compilation heuristic starts from a XAG representation of a Boolean function f and produces quantum circuits containing elementary quantum gates taken from the Clifford+T gate set. Since, as already observed, the T gate is particularly expensive to be applied, the overall number of T gates is considered a good measure for the cost of the quantum implementation. Therefore, in this experimental evaluation, we only consider the number of T gates in the circuits obtained applying the XAG-based quantum compilation heuristic with and without exploiting the decomposition based on D-reducibility.

A very interesting result shown in [91] is that the number of T gates in a quantum circuit for a Boolean function f can be expressed in terms of the number of AND gates in its XAG representation. This implies that it is possible to provide an upper bound on the number of T gates in a circuit for a function f in terms of its *multiplicative complexity*, i.e., the minimum number of AND gates required to realize f in XAG form. Computing the multiplicative complexity for an arbitrary Boolean function is intractable, however as shown in [23], it is sometimes possible to derive good estimates of this complexity measure exploiting structural regularities of the functions, as for instance the D-reducibility property. We can derive a quantum circuit for a D-reducible function $f = \chi_A \cdot f_A$ combining the quantum circuits obtained applying this heuristic to χ_A and f_A separately. This approach should be convenient since 1) we are able to compute the exact multiplicative complexity of χ_A, and build a XAG with the minimum number of AND gates required to realize χ_A (see Theorem 2 in [23]); 2) f_A is a function that depends on fewer variables, so its XAG representation might contain a reduced number of AND gates (as already experimentally verified in [23]); 3)the two circuit components can be combined by adding a single AND gate, and therefore only 4 T gates in the final quantum circuit. Therefore, this strategy should lead to a reduced number of T gates in the final quantum implementation of the function f. The experimental results confirm this expectation, showing a significant reduction in the number of T gates: compiling a quantum circuit exploiting the decomposition of the target function f as $\chi_A \cdot f_A$, we can obtain a cost gain in T gates of about 21%.

We report in Table 3.2 a subset of all the benchmarks that are considered for our experiments. The first column contains the name and the number of the output of the considered benchmark. The second column reports the number of inputs. The following column reports the cost, in terms of T gates, of the quantum circuit compiled without exploiting the D-reducibility regularity. Finally, the last two columns report the cost of the quantum circuits derived exploiting the D-reducibily decomposition, and the gain in the number of T gates. The last row reports the average results for all the benchmarks considered in our experiments.

From the results reported in Table 3.2, we can notice how some benchmarks highly benefit from the proposed strategy. For example, the benchmark *spla_32* shows a gain of 73%, in T gates. For other benchmarks the gain is much less significant, for example *m181_1* and *vg2_6*. There are also cases where the strategy based on D-reducibility gives circuits with a higher number of T gates (see for instance, *b10_3* and *in2_9*). This fact is due to the heuristic nature of both the XAG minimizer, used to derive the initial representation of the function, and of the quantum compiler itself. In general, we can observe how the overall T cost of the XAG based quantum compiler is much lower than the cost of the circuits derived from standard ESOP forms, both for decomposed and non-decomposed benchmarks.

3.4 Conclusion

In this chapter, we have explored the class of D-reducible functions and demonstrated how this structural regularity can be harnessed to implement more compact reversible circuits. Our approach leverages the inherent properties of D-reducible functions, which are characterized by their minterms being confined to a smaller affine subspace within the Boolean cube. This structural insight allows for the creation of reversible circuits that are optimized for quantum synthesis and compilation.

To validate our method, we conducted extensive experimental evaluations using both standard ESOP (Exclusive Sum of Products) forms and XAG (And-Inverter Graph) representations of the decomposed Boolean expressions. These experiments confirmed that our approach yields significant improvements in circuit compactness. Specifically, we observed notable reductions in the number of elementary quantum gates required, demonstrating the effectiveness of our technique in minimizing circuit size and enhancing efficiency.

Overall, the findings presented in this chapter lay the groundwork for continued advancements in quantum circuit design, with the potential for substantial improvements in both theoretical and practical aspects of quantum computing.

Chapter 4

Quantum Synthesis of Autosymmetric and Dimension Reducible Functions

This chapter introduces a new technique that streamlines the quantum synthesis process by leveraging the inherent regularities of autosymmetric and dimension-reducible Boolean functions. The results of the proposed approach show that it generates significantly more compact quantum circuits, reducing the overall quantum cost. The experimental results also demonstrated that the effectiveness of the proposed approach leads to considerable reduction in terms of cost required for the quantum circuits.

4.1 Introduction

As mentioned in the previous chapters, the synthesis of quantum circuits for Boolean functions is a critical area of research, particularly in the context of optimizing performance and resource utilization. This chapter introduces a novel approach for the quantum synthesis of functions characterized by the autosymmetry and dimension reducibility (D-reducibility) structural properties reviewed in Chapter 2 [13, 27]. The simultaneous presence of both autosymmetry and D-reducibility in Boolean functions opens up a new avenue for designing more efficient quantum circuits, as these functions exhibit both regularity in their algebraic structure and a reduction in their computational complexity.

D-reducible and Autosymmetric functions are defined by their minterms being confined to an affine subspace smaller than the whole Boolean cube and also presenting a structural regularity easily expressed using the XOR operator [19, 20, 87, 17]. These functions are prevalent in classical benchmarks; for instance, approximately 24% of the functions in the classical ESPRESSO benchmark suite [145] exhibit at least one autosymmetric output [19, 20], and 70% exhibit a D-reducible output [17]. Efficiently decomposing these regular functions in polynomial time makes them attractive targets for quantum synthesis, allowing for the development of circuits that are not only compact but also easier to implement.

Our approach leverages both autosymmetry and D-reducibility to design quantum circuits that minimize cost. By structurally decomposing these functions, we can reduce the number of quantum gates required for implementation, leading to circuits that are optimized for quantum compilation. A significant advantage of this method is that the resulting reversible circuits predominantly use CNOT gates, which are less expensive compared to T-gates that require additional input lines. This chapter provides a comprehensive discussion on how the combination of autosymmetry and D-reducibility offers a promising path toward optimizing quantum circuit synthesis.

The quantum compilation phase produces compact quantum circuits as validated by the experimental results that are conducted starting with XAG (XOR-AND-Inverter graphs) representation on the decomposed expressions. Experimental results demonstrate the effectiveness of our synthesis method for functions that are both autosymmetric and D-reducible. When applied to XAG-based quantum compilation [91], our approach results in a 30% reduction in T-gates, highlighting its potential for designing efficient quantum circuits in terms of quantum cost.

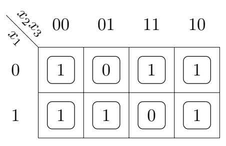
The organization of this chapter is as follows: Section 4.2 explains the completely specified autosymmetric and D-reducible Functions. Section 4.4 details our proposed methodology for constructing reversible circuits for autosymmetric and D-reducible functions. Section 4.5 evaluates the proposed decomposition and presents experimental findings on benchmark sets. Finally, we conclude this work in Section 4.6.

4.2 Completely Specified Autosymmetric and D-Reducible Functions

It is possible to use two principal decomposition strategies in the study of Boolean functions that exhibit both autosymmetry and D-reducibility. The first strategy involves beginning with D-reducibility, followed by applying autosymmetry. Conversely, the second strategy involves starting with autosymmetry and then proceeding to apply D-reducibility. This chapter presents quantum circuit synthesis specifically for these two approaches applied to completely specified functions.

When employing the first strategy, the decomposition starts with D-reducibility. This involves expressing the Boolean function f as $f = \chi_A \cdot f_A$, where A represents an affine space and f_A is the projection of f onto A. Following this, autosymmetry is applied to f_A, leading to the representation $f = \chi_A \cdot f_{Ak}$, where f_A is based on its autosymmetric properties. This sequential approach ensures that the function f is represented in terms of both its affine space and its autosymmetric characteristics.

Example 13 *Consider the function f presented in Figure 2.2, in Chapter 2. We apply D-reducibility first, followed by autosymmetry to the function. As demonstrated in the Example 5, in Chapter 2, f is D-reducible, and its projection is $f_A(x_1, x_2, x_3)$. The set of minterms for f_A is given by $f_A = \{000, 010, 011, 100, 101, 110\}$. Next, we compute the autosymmetry decomposition for f_A. The Karnaugh map of f_A is also depicted on the left side of Figure 4.1, where it is observed that f_A exhibits autosymmetry. The corresponding vector space is $L_{f_A} = \{000, 110\}$, which has dimension $k = \log_2 |L_{f_A}| = 1$, indicating that f_A is 1-autosymmetric. Projecting the minterms onto the Boolean space $\{0,1\}^2$ of the variables y_1 and y_2, we obtain the function $f_{A1}(y_1, y_2) = \{00, 10, 11\}$, which is represented in the Karnaugh map on the right side of Figure 4.1. The corresponding reduction equations are $y_1 = x_1 \oplus x_2$ and $y_2 = x_3$. A Sum-of-Products (SOP) form for f_{A1} is equal to*



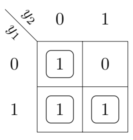


Figure 4.1: Karnaugh maps of the function $f_A(x_1, x_2, x_3)$ (left) and of its corresponding projection $f_{A1}(y_1, y_2)$ (right).

$y_1 + \overline{y_2}$. Substituting the reduction equations, we obtain $y_1 + \overline{y_2} = (x_1 \oplus x_2) + \overline{x_3}$. Recalling that the characteristic function of A is $\chi_A = (x_1 \oplus x_4)(x_2 \oplus x_5)$, the original function f can now be written as

$$f(x_1, \ldots, x_5) = (x_1 \oplus x_4) \cdot (x_2 \oplus x_5) \cdot [(x_1 \oplus x_2) + \overline{x_3})].$$

Conversely, the second strategy begins with autosymmetry. In this approach, we first determine the autosymmetry of the function f and obtain f_k, which is the restriction of f that reflects its autosymmetric property. Afterward, D-reducibility is applied to f_k, leading to the representation $f = \chi_A \cdot f_{kA}$, where A denotes the affine space, and f_{kA} is the projection of f_k onto A.

Example 14 *Let us consider the ongoing function again, but this time applying the decomposition method where autosymmetry is performed first, followed by D-reducibility. Referring to Figure 2.3 and Example 6 in Chapter 2, which demonstrate that f is 1-autosymmetric, and the restriction f_1 corresponds to the set of minterms $f_1(y_1, y_2, y_3, y_4) = \{0011, 1010, 1110\}$ in $\{0,1\}^4$. Next, we perform the D-reducibility decomposition on f_1. The Karnaugh map for f_1 is shown on the left side of Figure 4.2. Projecting f_1 onto A, we derive the Boolean function $f_{1A}(y_1, y_2) = \{00, 10, 11\}$, which is illustrated in the Karnaugh map on the right side of Figure 4.2 . The characteristic function for A is $y_3 \cdot (y_1 \oplus y_4)$. For simplicity, we represent f_{1A} is SOP form as $f_{1A} = (y_1 + \overline{y_2})$. It is worth noting that f_{1A} can be represented in various forms, in particular we will use the XOR-And Graph (XAG) representation in the experimental section. Thus, we have*

$$f_1(y_1, y_2, y_3, y_4) = \chi_A \cdot f_{1A} = (y_3 \cdot (y_1 \oplus y_4)) \cdot (y_1 + \overline{y_2}).$$

To reconstruct the original function f, we substitute the variables y_1, y_2, y_3, y_4 with their corresponding reduction equations as computed in Example 6. This gives us

$$\begin{aligned} f(x_1, \ldots, x_5) &= [(x_1 \oplus x_4) \cdot ((x_1 \oplus x_2) \oplus (x_1 \oplus x_5))] \\ &\quad \cdot [(x_1 \oplus x_2) + \overline{x_3})] . \end{aligned}$$

After simplification, the final expression becomes

$$f(x_1, \ldots, x_5) = (x_1 \oplus x_4) \cdot (x_2 \oplus x_5) \cdot [(x_1 \oplus x_2) + \overline{x_3})].$$

Finally, it is worth noting that this decomposition is identical to the one obtained using the alternate strategy in the previous example. We explore two decomposition strategies for representing the function f and show that they provide the same final representation. First, we decompose f by applying the D-reducibility

$y_1y_2 \backslash y_3y_4$	00	01	11	10
00	0	0	$\boxed{1}$	0
01	0	0	0	0
11	0	0	0	$\boxed{1}$
10	0	0	0	$\boxed{1}$

$y_1 \backslash y_2$	0	1
0	$\boxed{1}$	0
1	$\boxed{1}$	$\boxed{1}$

Figure 4.2: Karnaugh maps of the function $f_1(y_1, y_2, y_3, y_4)$ (left) and of its corresponding projection $f_{1A}(y_1, y_2)$ (right).

property and subsequently leveraging autosymmetry on the projected function f_A, resulting in $f = \chi_A f_{A_k}$. Alternatively, if we initially exploit the autosymmetry property of f and then apply the D-reducibility property to the restricted function f_k, we obtain $f = \chi_A f_{k_A}$. Note that both f_{A_k} and f_{k_A} depend on the same set of $a - k$ variables.

The following theorems, originally stated in [14], establish the required properties of D-reducibility and autosymmetry for both approaches.

Theorem 1 *Let f be a completely specified k-autosymmetric Boolean function depending on n binary variables. If f is D-reducible with an associated affine space A, then the projection f_A of f onto A is k-autosymmetric.*

To establish that these two decomposition strategies provide the same final representation of f, we also need to show that the restriction f_k of an autosymmetric function retains the D-reducibility property, as outlined in the following theorem.

Theorem 2 *Let f be a completely specified D-reducible Boolean function depending on n binary variables, with an associated affine space A. If f is k-autosymmetric, then the restriction f_k of f is also D-reducible with respect to the same affine space A.*

Having established the properties of each decomposition path through Theorems 1 and 2, we can now assert that these two strategies yield an identical representation of f, as captured in the following theorem [14].

Theorem 3 *The two decomposition strategies are equivalent, that is, $f_{A_k} = f_{k_A}$.*

This theorems confirms that regardless of the sequence in which D-reducibility and autosymmetry are applied, the same compact representation of f is obtained.

4.3 Incompletely Specified Autosymmetric and D-reducible Functions

In this section, the situation where an incompletely specified Boolean function f is both D-reducible and autosymmetric. The autosymmetry test for such a function assigns each “don’t care” to either 0 or 1, producing a fully specified function with the highest possible degree of autosymmetry [14]. As a result, after performing the autosymmetry test, the reduced function f_k is completely specified.

At the same time, D-reducibility reduction aims to find the smallest affine space A that encompasses all minterms of f. Points in A that are not minterms of f may be assigned 0 or left as "don't cares," meaning that the projected function f_A remains incompletely specified. Nevertheless, if f is both D-reducible and autosymmetric, the resulting decomposed functions f_{kA} and f_{Ak} are fully specified due to the autosymmetry test.

Building on the findings in [14], if f is incompletely specified, f_{kA} and f_{Ak} may differ based on the order of application and different reduction sequences can lead to distinct final results. Thus, it should be noted that for incompletely specified functions, both approaches should be applied, with the most optimal result selected afterward.

4.4 Quantum Circuit Synthesis for Autosymmetric and D-Reducible Functions

As discussed in the previous chapter, there is a natural correspondence between ESOP expressions and reversible circuits. This arises from the fact that product terms in an ESOP expression can be effectively represented using reversible MPMC Toffoli gates. Given an ESOP expression, one can generate a sequence of MPMC Toffoli gates, where the control lines are associated with the literals in the product terms and the target line represents the output of the function. It should be noted that all gates act on the same target line, thereby implementing the exclusive OR of the product terms.

The primary challenge with this approach lies in the fact that MPMC Toffoli gates are only intermediate representations. To implement these gates in quantum circuits, an additional synthesis step is required to map the reversible MPMC Toffoli gates into functionally equivalent quantum circuits. Unfortunately, this mapping step can become very onerous, particularly when the MPMC Toffoli gates are controlled by many variables, leading to quantum circuits with significant size and depth.

To address these challenges, in this section, we therefore propose a comprehensive approach for designing circuits that implement functions exhibiting both autosymmetry and D-reducibility. These methods are intended to optimize the reversible synthesis process, leading to quantum circuits that are efficient and minimized in terms of size and depth. The methodology is divided into two distinct strategies, depending on the order in which autosymmetry and D-reducibility are considered during the circuit construction.

4.4.1 Autosymmetric-D-Reducible Design

The first approach focuses on functions that are first analyzed for autosymmetry and then subjected to D-reducibility. Initially, an autosymmetry test is conducted to determine the linear space L, identify the characteristic function χ_A, and compute the restricted function f_k, which corresponds to the autosymmetric structure of the original function. Once the autosymmetry analysis is complete, D-reducibility is applied to the restricted function f_k. At this stage, the affine space A is derived based on the restricted function f_k, and provides the foundation for the D-reducibility step. The restricted function f_k is then projected onto the affine space A, resulting in the simplified function f_{kA}. This sequential decomposition approach highlights the interplay between autosymmetric properties and affine projections, where autosymmetry is applied first to reduce the complexity of the function, and D-reducibility is subsequently used to further simplify it by projecting onto an affine space.

The circuit design under this approach involves the integration of three key reversible circuits:

- Circuit for Deriving Reduction Equations: The initial circuit is designed to derive the reduction equations for the linear space L. These equations involve both canonical and non-canonical variables within L. The primary aim of this circuit is to simplify the function by reducing the number of variables, thus preparing the function for subsequent processing.

- Circuit for the Characteristic Function χ_A: The next circuit computes the characteristic function χ_A of the affine space A. This function depends on non-canonical variables within the linear space L, which includes both canonical and non-canonical variables within the affine space A. Computing χ_A is crucial as it provides the necessary foundation for the final projection step.

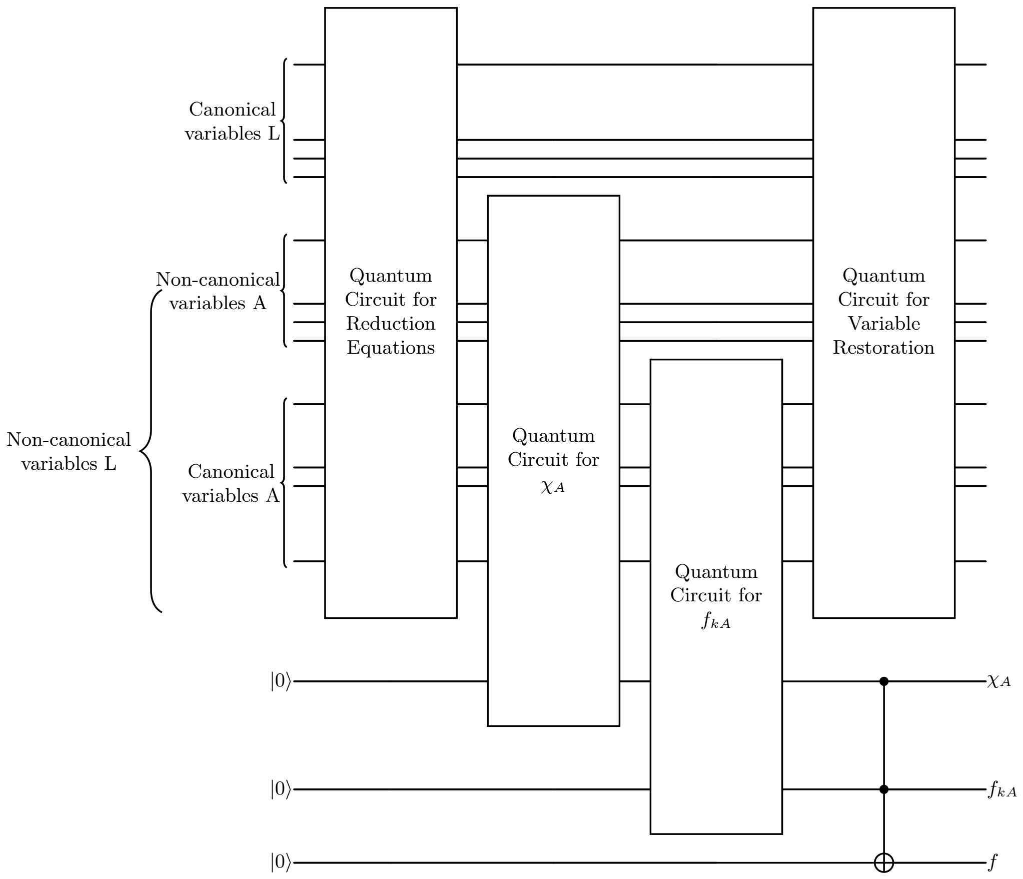
- Circuit for Implementing Function Projection f_{kA}: The next circuit implements the projection of the function onto the restricted function, resulting in f_{kA}. This projection circuit focuses on the non-canonical variables within the affine space A, ensuring that the function is further simplified and optimized for the final quantum synthesis.

- Circuit for Restoring Initial Variables: This stage in the synthesis process focuses on designing a circuit to restore the initial values of canonical and non canonical variables of the linear space L after performing the required computations.

To compute the final function f, a Toffoli gate is utilized to perform an AND operation between the subfunctions χ_A and f_{kA}, with f as the output target. This structured approach ensures that the synthesis of the quantum circuit is both streamlined and efficient, minimizing resource usage while maintaining the integrity of the function.

Figure 4.3 illustrates the overall structure of the circuit designed using this method. As shown, this approach requires three additional lines, corresponding to three extra qubits in the quantum implementation: one for χ_A, one for f_{kA}, and one for f. These additional qubits are essential for handling the intermediate results of the subfunctions before combining them to produce the final output.

Overall, the implementation of the circuit begins with the construction of the XOR layer. This layer, composed of multiple XOR gates, is necessary to realize the reduction equations, which are based on linear combinations of the variables. The XOR layer is a critical component as it enables the reduction process by simplifying the function's dependency on certain variables, making the final quantum circuit more compact and efficient. Then, a reversible circuit for χ_A can be implemented directly from the CEX expression of the affine space A, and a reversible circuit for f_{kA} depends only on the canonical variables of the affine space A. Finally an uncomputing procedure restores all variables to their initial values.



Canonical variables L
Non-canonical variables A
Canonical variables A
Non-canonical variables L
Quantum Circuit for Reduction Equations
Quantum Circuit for χ_A
Quantum Circuit for f_{kA}
Quantum Circuit for Variable Restoration
$|0\rangle$
$|0\rangle$
$|0\rangle$
χ_A
f_{kA}
f

Figure 4.3: Reversible circuit with Autosymmetric-D-Reducible strategy.

Example 15 *Consider the function $f = \{00011, 01010, 01110, 10001, 10101, 11000\}$ in the running example. To compute the reduction equations defining the new variables y_1, y_2, y_3, y_4, we use three CNOT gates to perform the XOR operations $x_1 \oplus x_2$, $x_1 \oplus x_4$ and $x_1 \oplus x_5$. As mentioned in Section 2.1.1, a CEX consists of a conjunction of XOR factors, with each XOR factor being implementable using CNOT gates. In particular, an XOR factor with k literals can be realized using $k - 1$ CNOT gates. In this case, for χ_A we use a CNOT gate to perform $y_1 \oplus y_4$, and a Toffoli gate that combines y_3 with the result of $y_1 \oplus y_4$. The projection f_{kA} can be computed by a Toffoli gate, which is initialized with a qubit in state $|1\rangle$ to compute the NAND between the complement of $\overline{y}_1$ and y_2, to perform $y_1 + \overline{y}_2$ operations. Recall that, to simplify the description of our example, we represent the function f_{kA} in SOP form. Note that f_{kA} can be expressed in various forms, but we will use the XAG representation in the experimental section. Subsequently, the next Toffoli gate computes the AND between χ_A and f_{kA}. Finally four additional gates are used to restore the input variables. As can be observed in Figure 4.4, this quantum implementation requires three Toffoli and eight CNOT gates.*

In this example, four T-gates suffice to represent χ_A, as proved in the following proposition.

Proposition 2 *Let $\chi_A : \{0,1\}^n \rightarrow \{0,1\}$ be the canonical CEX expression representing an affine subspace A of $\{0,1\}^n$, where the number of factors in χ_A is de-*

noted by $|\mathcal{F}|$. Then, a quantum circuit for χ_A can be implemented using $4 \cdot (|\mathcal{F}|-1)$ T-gates.

Proof. The function $\chi_A : \{0,1\}^n \to \{0,1\}$ represents the characteristic function of an affine subspace $A \subseteq \{0,1\}^n$, thus it is a conjunction (AND) of XOR factors. If the number of factors in the expression for χ_A is $|\mathcal{F}|$, then the number of AND gates required to implement χ_A is $|\mathcal{F}| - 1$. This is because each AND gate operates on two inputs, and to compute a conjunction of $|\mathcal{F}|$ terms, we need $|\mathcal{F}| - 1$ binary AND operations. Each AND gate in a quantum circuit can be implemented using a Toffoli gate. It is known that a Toffoli gate can be decomposed into Clifford and T-gates, specifically requiring 4 T-gates. Therefore, each AND gate in the circuit corresponds to 4 T-gates. Overall, there are $|\mathcal{F}| - 1$ AND gates to compute χ_A, and each AND gate corresponds to a Toffoli gate, which in turn requires 4 T-gates. In addition, the XOR gates are not implemented exploiting T-gates. Finally, the total number of T-gates needed to implement χ_A is $4 \cdot (|\mathcal{F}|-1)$. ■

Proposition 3 *Let $\chi_A : \{0,1\}^n \to \{0,1\}$ be the canonical CEX expression representing an affine subspace A of $\{0,1\}^n$, and let f_{kA} be the function projected onto the affine subspace A. Then, reversible circuits for f that is obtained by autosymmetric-D-reducible can be implemented with just two additional ancilla lines.*

Proof. The implementation of χ_A requires a Toffoli gate to realize the necessary AND operations, with each Toffoli gate necessitating one ancilla qubit to store its output. Consequently, a single additional ancilla line is sufficient for the implementation of χ_A. Similarly, the function f_{kA} also relies on a Toffoli gate for its implementation. Each Toffoli gate again requires one ancilla qubit, resulting in the need for just one additional ancilla line for f_{kA}. Thus, both functions can be efficiently implemented with only two additional ancilla lines, ensuring the outputs are preserved for further computation, as shown in Figure 4.3. ■

4.4.2 D-Reducible-Autosymmetric Design

The second approach reverses the order of analysis, starting with D-reducibility before addressing autosymmetry. In this strategy, the process begins with the identification of the characteristic function χ_A and also the projection f_A, which are derived from the D-reducibility test. Once these components are established, the autosymmetry properties of the function are considered and tested directly onto f_A, leading to the reduction equations and restriction function f_{Ak}.

The circuit design for this approach involves the following three reversible circuits:

- Circuit for the Characteristic Function χ_A: The initial step involves designing a circuit to compute the characteristic function χ_A of the affine space A. This function depends on all variables within A, including both canonical and non-canonical variables. Calculating χ_A is crucial for understanding the structure of the affine space and provides the foundation for subsequent steps in the synthesis process.

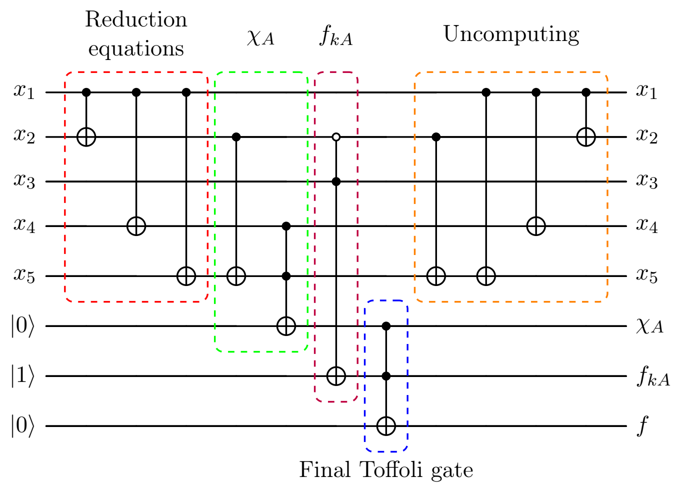


Figure 4.4: Reversible circuit for the function f with Autosymmetric-D-Reducible strategy.

- Circuit for Deriving Reduction Equations: After computing χ_A, the next circuit focuses on deriving the reduction equations for the linear space L. This circuit processes the canonical variables of the affine space A, which includes both non-canonical and canonical variables within L, simplifying the function by reducing the number of variables involved. This reduction is essential for optimizing the function and preparing it for the final projection.

- Circuit for Implementing Function Projection f_{Ak}: The next step involves a circuit that projects the function onto the restricted space, resulting in f_{Ak}. This projection circuit utilizes only the non-canonical variables within the linear space L, ensuring that the function is appropriately reduced and ready for the final synthesis process.

- Circuit for Restoring Initial Variables: This step in the synthesis process involves designing a circuit to restore the initial values of canonical and non canonical variables of the affine space A after performing the required computations.

To compute the final function f, a Toffoli gate is used to perform an AND operation between the subfunctions χ_A and f_{Ak}, with the output f serving as the final result. This structured strategy ensures that the quantum circuit synthesis is both effective and efficient, with careful consideration given to minimizing the circuit's size and depth.

The overall circuit structure, as depicted in Figure 4.5, shows that this approach necessitates three additional lines, corresponding to three extra qubits in the quantum implementation: one for χ_A, one for f_{Ak}, and one for f. These additional qubits are crucial for managing the intermediate computations required during the synthesis process.

In implementing these circuits, the process begins with constructing the reversible circuit for χ_A that is implemented starting from the CEX expression of the affine space A. Then, the XOR layer for the reduction equations is constructed, followed by a reversible subcircuit for computing f_{Ak}. Note that this subcircuit depends only on the non-canonical variables of the linear space L. Finally an uncomputing procedure restores all variables to their initial values.

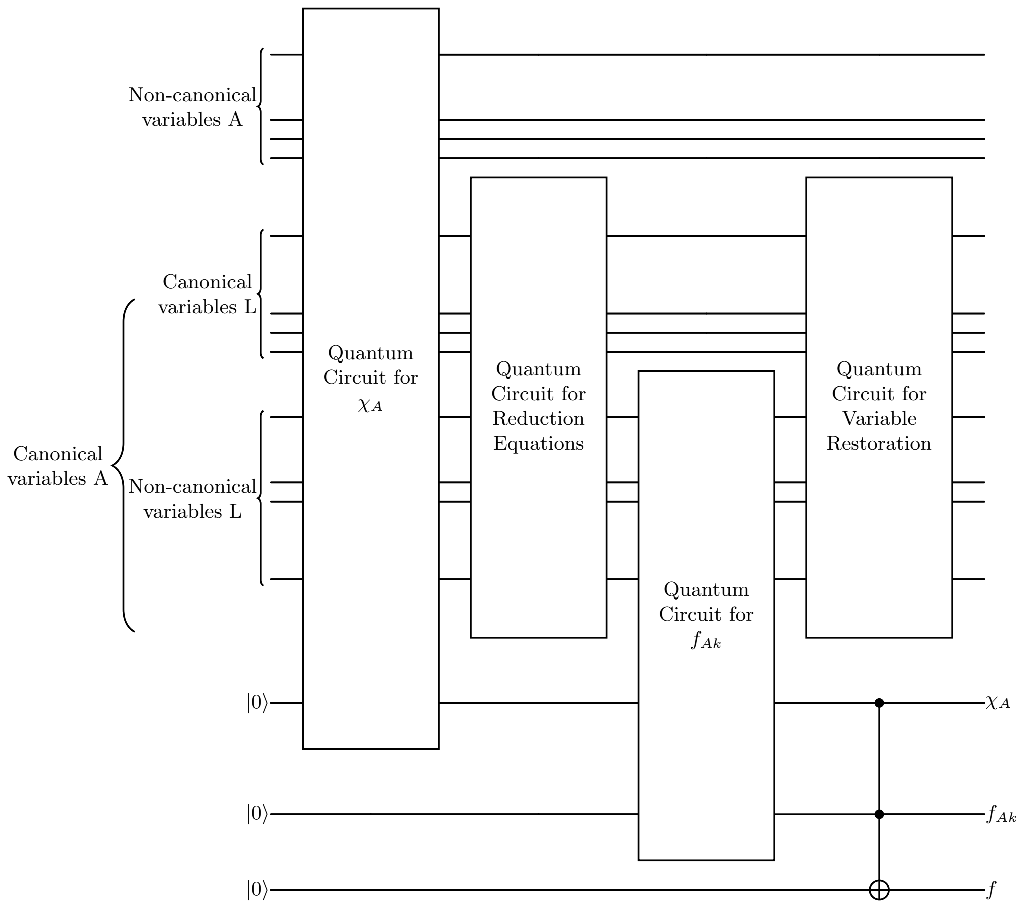

Non-canonical variables A
Canonical variables L
Canonical variables A
Non-canonical variables L
Quantum Circuit for χ_A
Quantum Circuit for Reduction Equations
Quantum Circuit for f_{Ak}
Quantum Circuit for Variable Restoration
$|0\rangle$
$|0\rangle$
$|0\rangle$
χ_A
f_{Ak}
f

Figure 4.5: Reversible circuit with D-Reducible-Autosymmetric strategy.

Example 16 *Consider the function* $f = \{00011, 01010, 01110, 10001, 10101, 11000\}$ *from the running example once again. To compute* χ_A*, the detailed process for deriving* χ_A *is provided in Section 2.1.1, in Chapter 2, we use two CNOT gates to perform* $x_1 \oplus x_4$ *and* $x_1 \oplus x_5$*, and also a Toffoli gate that combines the result of* $x_1 \oplus x_4$ *with the result of* $x_1 \oplus x_5$*. We also use a CNOT gate to perform the reduction equation* y_1 *and* y_2*. To compute* f_{Ak}*, a Toffoli gate, which is initialized with a qubit in state* $|1\rangle$ *to compute the NAND between the complement of* $\overline{y}_1$ *and* y_2*, is used to compute the* $y_1 + \overline{y_2}$ *operation. Then, the next Toffoli gate computes the AND between* χ_A *and* f_{Ak}*. Finally, three additional gates are used to restore the original input variables. As depicted in Figure 4.6, this quantum implementation requires three Toffoli gates and six CNOT gates. It should be noted that, according to proposition 2, the number of T-gates for* χ_A *in this example equals four.*

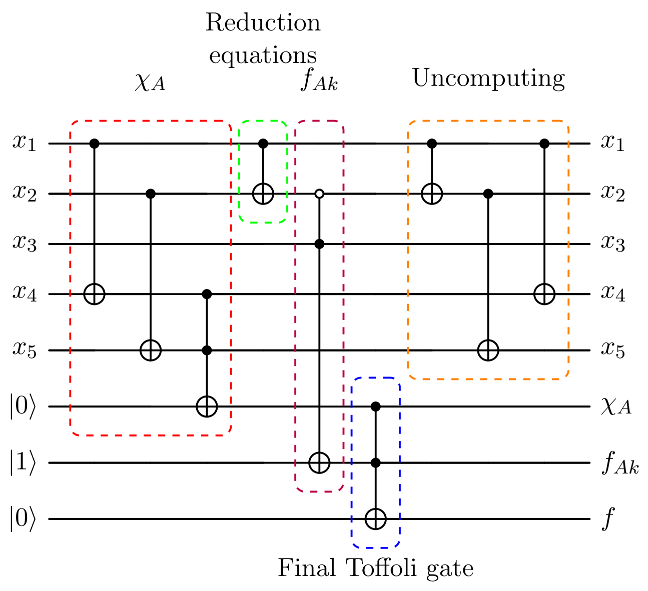


Figure 4.6: Reversible circuit for the function f with D-Reducible-Autosymmetric strategy.

Overall, both approaches provide a systematic and efficient pathway for designing quantum circuits that handle functions with autosymmetry and D-reducibility, ensuring that the final circuits are not only functional but also optimized for quantum implementation.

4.5 Experimental Results

In this section, we assess the effectiveness of the proposed method for quantum synthesis of autosymmetric and dimension reducible functions. We present the computational results obtained by constructing decomposed expressions for Boolean functions and comparing them to their standard quantum synthesis counterparts, which do not utilize these structural regularities. The objective of this experiment is to measure the extent to which the proposed strategy can be exploited to implement more compact circuits.

All experiments were conducted on an Intel i7-8550U CPU running at 1.80GHz with 16GB of RAM. We used classical benchmarks in PLA format, specifically from the Espresso and LGSynth'89 benchmark suites [145]. It should be noted that autosymmetry and D-reducibility are properties of single outputs, which means different outputs from the same benchmark can have different autosymmetry degrees and D-reducibility for each output. As a result, each output is considered as a separate Boolean function for analysis, leading to the examination of 237 (non-degenerate) autosymmetric and D-reducible functions. The given functions, along with their restrictions or projections, were synthesized using the structural decomposition strategies depicted in Figures 4.3 and 4.5, with ESPRESSO [32] employed in heuristic mode for SOP synthesis and then they were synthesized by applying the XAG-based quantum compilation heuristic proposed in [139]. Our major focus was to experimentally determine whether advanced quantum compilation methods could benefit from the decomposition based on D-reducibility and autosymmetry, or they are capable of deriving compact quantum circuits without exploiting these structural regularities.

To evaluate the reversible circuits derived from proposed decomposition strategies and compare them with those derived without decomposition, we measured the cost in terms of elementary quantum T-gates. Specifically, we mapped each MPMC Toffoli gate into elementary quantum gates based on the Clifford+T library and the algorithm outlined in [91]. Since the T-gate is recognized as the most costly gate in the library, the cost of a Toffoli gate is typically expressed according to the number of T-gates required for its implementation. Therefore, in this experimental evaluation, we only consider the number of T- gates in the circuits obtained applying the XAG-based quantum compilation heuristic with and without exploiting the decomposition based on D-reducibility-autosymmetry and autosymmetry-D-reducibility strategies.

A noteworthy result highlighted in [91] indicates that the number of T gates in a quantum circuit for a Boolean function f can be expressed as a function of the number of AND gates in its XAG representation.

Table 4.1 reports a significant subset of benchmarks as representative indicators of our experiments. The first column reports the name and the number of the considered autosymmetric and D-reducible output of each benchmark. The second column reports the costs, in terms of elementary quantum gates, of the reversible circuits derived from minimal SOP expressions of the benchmarks, without exploiting the structural regularities. The following column reports the cost of the reversible circuits derived exploiting the autosymmetry-D-reducibility decomposition, as explained in Section 4.4.1. The next column reports the required costs of the reversible circuits derived exploiting the D-reducibility-autosymmetry decomposition, as described in Section 4.4.2. The last group of 2 columns reports the T-gain of the reversible circuits derived exploiting the autosymmetry-D-reducibility decomposition and D-reducibility-autosymmetry decomposition, respectively. SOP minimization of the benchmarks and of their projections onto the associated affine spaces is performed using the ESPRESSO heuristic [97]. Due to the heuristic nature of this SOP minimizer, the synthesis times for the functions and their projections are similar and very short, leading to negligible gain in synthesis time.

Finally, the last column reports the gain in the number of T gates, and the last row reports the average costs for all the benchmarks considered in our experiments.

As shown in Table 4.1, it is clear that some benchmarks experience significant advantages from decomposed expressions in terms of the number of T-gates compared to standard synthesis. For instance, the benchmarks *exp11* and *rd53* achieve a significant reduction in T-gates when utilizing D-reducibility-autosymmetry and autosymmetry-D-reducibility decomposition, respectively. Specifically, *exp11* shows a 67% reduction in T-gates, while *rd53* shows a 75% reduction in T-gates. However, in some cases, such as with the *b10* benchmark, standard synthesis results in circuits with fewer T-gates compared to synthesis that utilizes structural regularities.

The gain obtained synthesizing a reversible circuit exploiting these structural regularities is quite interesting. The experiments demonstrate that approximately 41% of the functions benefit from autosymmetry and D-reducibility decomposition, resulting in an average reduction of about 30% in the number of T gates.

Table 4.1: Comparison between standard synthesis and the proposed strategies for functions that are both autosymmetric and D-reducible.

	Standard Synthesis	**With Decomposition**		**T-Gain**	
Benchmark	**T-count**	**T-count (A+D)**	**T-count (D+A)**	**A+D**	**D+A**
add6_0	12	**4**	**4**	67%	67%
adr4_4	12	**4**	**4**	67%	67%
apla_3	28	20	**16**	29%	43%
apla_5	28	28	**16**	0%	43%
b10_4	**60**	64	64	-7%	-7%
bench_3	32	**12**	**12**	63%	63%
bench_7	20	20	20	0%	0%
dk17_10	24	**12**	**12**	50%	50%
dk17_4	40	**16**	20	60%	50%
dk27_0	20	12	**8**	40%	60%
exam_4	60	104	**52**	-73%	13%
exep_6	72	**64**	**64**	11%	11%
exp_11	60	24	**20**	60%	67%
p1_0	**0**	12	8	0%	0%
p3_7	**116**	156	128	-34%	-10%
rd53_1	16	**4**	12	75%	25%
spla_22	92	**52**	**52**	43%	43%
t1_19	20	20	**12**	0%	40%
xor5_0	12	**4**	**4**	67%	67%
Z5xp1_8	12	**4**	**4**	67%	67%

Particularly, The autosymmetry-D-reducibility strategy results in a 31% reduction in T-gates, while the D-reducibility-autosymmetry approach achieves a 41% reduction in T-gates.

Overall, we can conclude that the quality of reversible circuits obtained by applying the decomposition scheme with D-reducibility first, followed by autosymmetry, tends to outperform those derived from the scheme where autosymmetry is applied before D-reducibility.

It is worth mentioning that, the running times are significantly influenced by the XAG heuristic [139], which generally varies based on the dimension of the input function. Consequently, when both testing procedures are performed, the overall running times are often reduced, as the input function for the XAG heuristic tends to be smaller. In other words, the time saved in constructing the XAG outweighs the time needed to test the two regularities. Additionally, for completely specified functions, where the results of A+D and D+A are the same, the D+A strategy is more convenient due to its superior running times, and for incompletely specified functions, it is convenient to test both the strategies A+D and D+A in order to find the best result [18].

4.6 Conclusion

In this chapter we have considered the class of D-reducible and autosymmetric functions and have shown how these regularities can be exploited to implement compact reversible circuits at any order. The proposed quantum circuits require just two additional input lines, and the final reversible circuit exhibits an uncomputing part implemented with CNOT gates only, no T-gates are required for this part. Experimental results, based on XAG representations of the decomposed expressions, have validated the proposed approach. The results show that the reversible circuits obtained by first applying the decomposition scheme with D-reducibility, followed by autosymmetry, tend to outperform those derived from the scheme where autosymmetry is applied before D-reducibility. Future work can consider new regularities to improve quantum compilation. Moreover, we plan to investigate the potential of alternative decomposition techniques in deriving more compact quantum circuits.

Chapter 5

Quantum Synthesis for PSOP Based Functions

This chapter explores the application of PSOP decomposition techniques in quantum circuit synthesis, addressing the challenges of optimizing gate count and circuit depth on current quantum hardware. By breaking down complex operations into simpler, more efficient components, these techniques significantly enhance the practicality of quantum circuits with lower cost. The chapter presents a detailed methodology, along with theoretical and experimental validation, demonstrating the effectiveness of these approaches in reducing circuit quantum cost. The findings contribute to the ongoing efforts to optimize quantum circuit design, offering valuable insights for future research and the development of more efficient quantum computing systems.

5.1 Introduction

In recent years, studies like [91, 129, 144] have highlighted efforts to optimize reversible circuits for quantum synthesis. Additionally, several pre-processing methods have emerged to enhance reversible synthesis and quantum compilation. These methods often leverage structural regularities in the input Boolean functions [22, 27], making them primarily useful for specific types of functions, such as autosymmetric and D-reducible functions.

This chapter presents a novel pre-processing approach designed to overcome this limitation by applying the method to any Boolean function, regardless of regularities. The approach involves decomposing the original function f and recomposing it after the quantum compilation phase. The decomposition process operates with linear-time complexity, while the recomposition step is constant in time, ensuring high efficiency.

These characteristics make the proposed strategy an effective and fast pre-processing step before performing reversible synthesis and quantum compilation. The method relies on a structural non-disjoint decomposition of the input function, called the *Projected Sum of Products* (PSOP), which is an EXOR-based decomposition. PSOP forms extend the standard Shannon decomposition, and this choice is advantageous because EXOR-based reconstructions have a very low quantum cost, and the decomposition itself is very fast, with linear-time complexity. Figure 5.1 provides a comparison between standard quantum synthesis techniques and the proposed approach.

The theoretical framework in this chapter explains the pre-processing method and the reconstruction strategy in detail. Specifically, it shows that after performing the PSOP decomposition and synthesizing the quantum components, it is possible to reconstruct the original function f by adding only a few Toffoli gates, typically two or three, equivalent to 8 or 12 T-gates [91].

To validate the effectiveness of this approach, we compare the quantum synthesis of PSOP-decomposed functions against traditional quantum compilation techniques. The experimental results indicate that the proposed pre-processing step improves performance for 61% of the benchmarks, with an average gain of around 22% in terms of T-gates, utilizing the XAG-based quantum compilation described in [91]. The results presented in this chapter were presented at the DSD Conference 2024 and have been invited for a special issue in the WiPiEC Journal-Works in Progress in Embedded Computing Journal [26].

The chapter is organized as follows. Section 5.2 describes our proposed new methodology for construction of quantum circuit based on PSOP decomposition. Section 5.3 shows the evaluation of the proposed decomposition and reports our experiments on a set of benchmarks. Finally, the conclusion of this work is given in Section 5.4.

5.2 Quantum Circuits Synthesis Based on PSOP Decomposition

In this section, we describe how the PSOP decomposition of a Boolean function f can be exploited to ease its quantum compilation. More precisely, we show how to combine quantum circuits for the two projected functions $f_{|x_i=p}$ and $f_{|x_i\neq p}$, the function p, and the remainder r, if present, in order to derive a quantum circuit for the original function f, following the strategies depicted schematically in Figure 5.1. Potential benefits of this approach are a reduced compilation time, and a final quantum circuit of reduced area with respect to the quantum circuit derived compiling directly the function f without leveraging its PSOP decomposed forms.

As already observed, this new quantum compilation strategy can be applied to any Boolean function, after the fast PSOP decomposition step, whose cost is linear in the initial SOP representation of the target function.

Let f be a Boolean function depending on n binary variables, and let PSOP(f) and Pr-SOP(f) denote its PSOP forms, without and with remainder r:

$$\begin{aligned}\text{PSOP}(f) &= (\overline{x}_i \oplus p) f_{|x_i=p} + (x_i \oplus p) f_{|x_i\neq p}\,,\\ \text{Pr-SOP}(f) &= (\overline{x}_i \oplus p) f_{|x_i=p} + (x_i \oplus p) f_{|x_i\neq p} + r\,,\end{aligned}$$

where x_i is one input variable, p is a function that does not depend on x_i, $f_{|x_i=p}$ and $f_{|x_i\neq p}$ are the two projections of the SOP expression of f, and r is the remainder. Recall that both forms can be derived in linear time.

It is worth mentioning that in the Projected Sum of Products (PSOP), inserting remainders avoids cutting through crossing products, thus limiting the overall number of terms. However, the disadvantage arises when a crossing cube is removed from the projection: by excluding these cubes, the opportunity to form larger cubes in the projected part is limited.

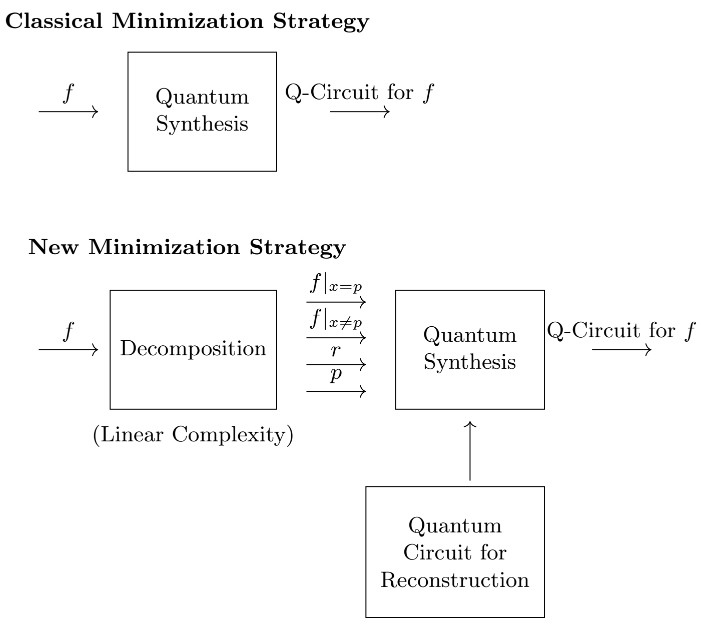


Figure 5.1: Classical and new minimization strategies *without* SOP synthesis.

Before the quantum synthesis step, a heuristic SOP minimization step could be performed to facilitate quantum compilation and possibly derive more compact circuits, as shown in Figure 5.2. This step can be performed by applying polynomial time SOP heuristics on all components of the PSOP(f) and Pr-SOP(f) expressions, which are generally smaller functions, that depend on fewer variables and contain fewer minterms than the target function f. Notice that a similar step in the standard quantum compilation flow, not based on decomposition, would require the more costly heuristic SOP minimization of the whole function f. This preliminary minimization is not mandatory and can be avoided in case of large benchmarks, whose SOP minimization could result too time demanding.

After the optional SOP minimization step, quantum compilation is applied independently onto the subfunctions p, $f_{|x_i=p}$, $f_{|x_i \neq p}$, and the remainder r (if present).

Finally, we derive a quantum circuit for the overall function f using the quantum circuits for p, $f_{|x_i=p}$, $f_{|x_i \neq p}$, and the remainder r as building blocks, as shown in Figures 5.3 and 5.4.

Before describing how to derive a quantum circuit for a function f from PSOP decomposition, we state and prove a proposition that allows to ease the reconstruction strategy.

Proposition 4 *Let f be a Boolean function depending on n binary variables, and let PSOP(f) and Pr-SOP(f) be its PSOP decomposition without and with remainder, respectively. The disjunction between the first two terms in both algebraic expressions can be replaced with an Exclusive Or:*

$$PSOP(f) = (\overline{x}_i \oplus p) f_{|x_i=p} \oplus (x_i \oplus p) f_{|x_i \neq p}\,,$$
$$Pr\text{-}SOP(f) = \big((\overline{x}_i \oplus p) f_{|x_i=p} \oplus (x_i \oplus p) f_{|x_i \neq p}\big) + r\,.$$

Proof. Observe that the first two terms in both PSOP(f) and Pr-SOP(f) represent disjoint sets of points. Indeed, the two subspaces $B_{x_i=p}$ and $B_{x_i \neq p}$ do not

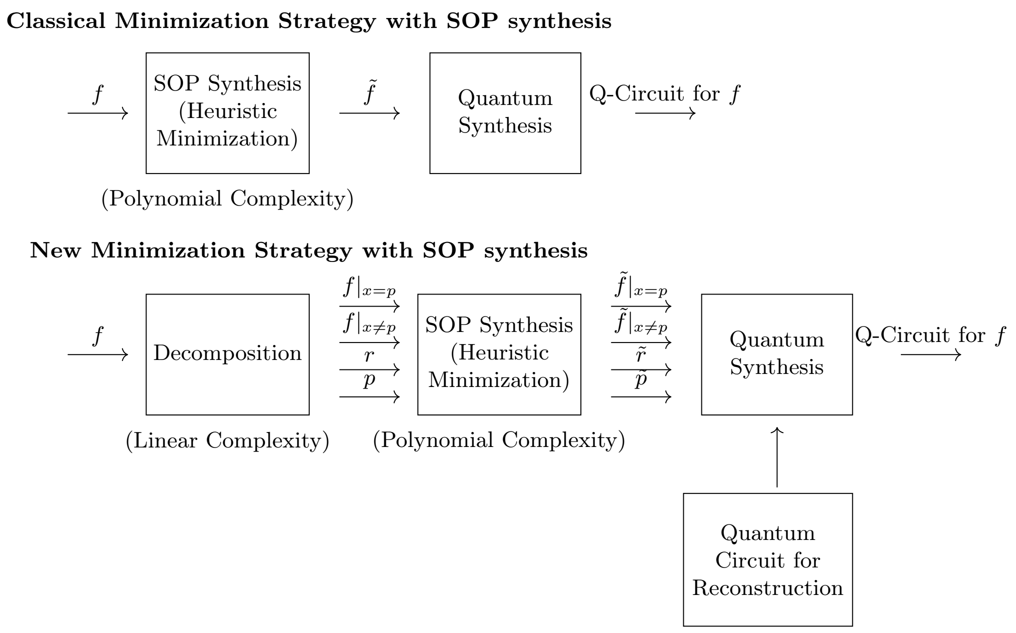


Figure 5.2: Classical and new minimization strategies *with* SOP synthesis.

intersect, and the product of their characteristic functions $(\overline{x}_i \oplus p)$ and $(x_i \oplus p)$ is the zero function. This immediately implies that the disjunction can be replaced with an exclusive OR. ■

This result is important for the reconstruction procedure since an EXOR can be easily implemented in a quantum circuit using a CNOT instead of a Toffoli gate.

We now describe the reconstruction procedure of a quantum circuit for f, considering first the case of PSOP decomposition without remainder.

The overall quantum circuit for f in this case is obtained concatenating the two quantum subcircuits for the projections, that depend on all variables but x_i, with the quantum circuit for $(x_i \oplus p)$, possibly depending on all input variables. The quantum circuit for $(x_i \oplus p)$ can be derived inserting a CNOT, controlled by x_i, on the output line of a quantum circuit for p. Note that four additional lines (and therefore four new qubits) are needed: one for $f_{|x_i=p}$, one for $f_{|x_i\neq p}$, one for $(x_i \oplus p)$ and finally one output line for f. The overall circuit structure is shown in Figure 5.3, where two swap gates are used to bring the qubits for the intermediate results closer to the corresponding subcircuits. Eventually, two Toffoli gates are inserted for computing the AND between the projections and the corresponding subspaces, one described by the subcircuit for $(x_i \oplus p)$ and the other by its complement. Both Toffoli gates act on the output line for f, thanks to the fact that the OR operator in the PSOP expression has been replaced with an EXOR. The overall methodology is summarized in the algorithm in Figure 5.5, and its cost in terms of elementary quantum T-gates is discussed in the following proposition.

Proposition 5 *The number of T-gates required to synthesize the PSOP-based quantum circuit for f is given by the overall number of T-gates occurring in the subcircuits for $f_{|x_i=p}$, $f_{|x_i\neq p}$, and $(x_i \oplus p)$, plus 8 additional T-gates.*

Proof. Observe from Figure 5.3 that the three quantum subcircuits for $f_{|x_i=p}$,

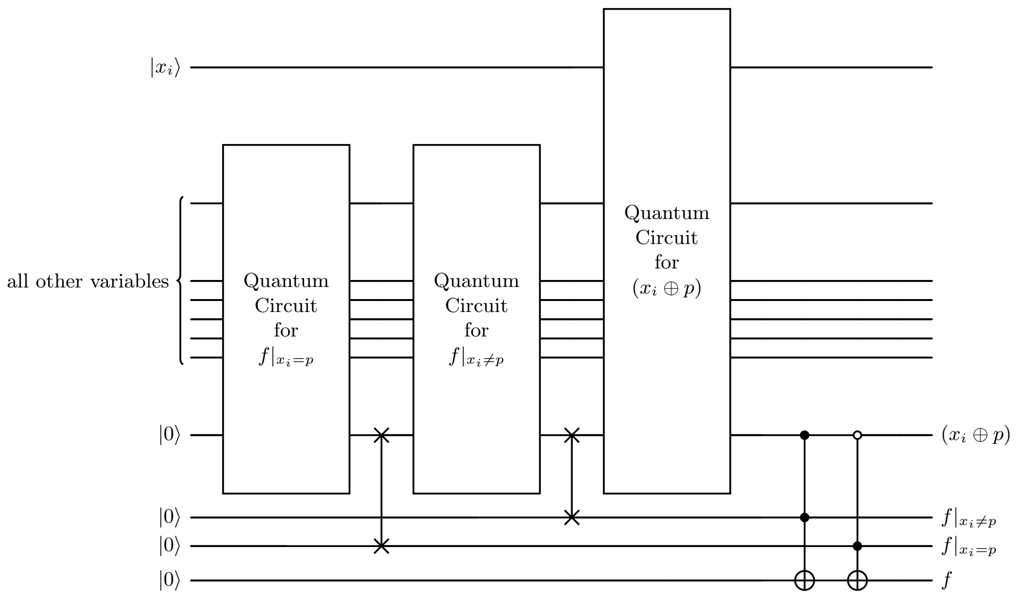


Figure 5.3: Quantum circuit based on PSOP without remainder.

$f_{|x_i \neq p}$, and $(x_i \oplus p)$ are combined using only two additional swap gates and two Toffoli gates. Since swap gates are implemented using CNOTs, only 8 additional T-gates are required, four for each Toffoli gate [91]. ■

Figure 5.4 shows the circuit for f based on the PSOP decomposition with remainder. As already noted in Proposition 4, the first disjunction can be replaced by an EXOR. Moreover, using De Morgan's laws, we can replace the remaining OR with a NAND. Thus the form becomes

$$\text{Pr-SOP}(f) = \overline{\overline{\left((\overline{x}_i \oplus p)f_{|x_i=p} \oplus (x_i \oplus p)f_{|x_i \neq p}\right)} \wedge \overline{r}}$$

The overall quantum circuit for f is thus obtained concatenating the subcircuits for the projections, for the characteristic function $(x_i \oplus p)$ of the projection subspace, and for the remainder r, possibly depending on all input variables.

This time, six additional lines are used: two for $f_{|x_i=p}$ and $f_{|x_i\neq p}$, one for $(x_i \oplus p)$, one for the remainder, one for storing the intermediate result $(\overline{x}_i \oplus p)f_{|x_i=p} \oplus (x_i \oplus p)f_{|x_i\neq p}$, and one as output line for f. As before, swap gates are used to bring the qubits for the intermediate results closer to the corresponding subcircuits.

Two Toffoli gates, both acting on the same line, are then used for computing the EXOR of the products between the projections and the corresponding subspaces. A third Toffoli gate on the output line for f, inizialized with a qubit in state $|1\rangle$, is finally used to compute the NAND between the complement of the EXOR of the two products on the second to last line, and the complement of the remainder r.

The overall methodology, summarized in the algorithm in Figure 5.5, requires a constant number of additional T-gates for combining the four quantum subcircuits, as stated and proved in the following proposition.

Proposition 6 *The number of T-gates required by the quantum circuit based on PSOP decomposition with remainder is given by the overall number of T-gates occurring in the subcircuits for $f_{|x_i=p}$, $f_{|x_i\neq p}$, $(x_i \oplus p)$, and r, plus 12 additional T-gates.*

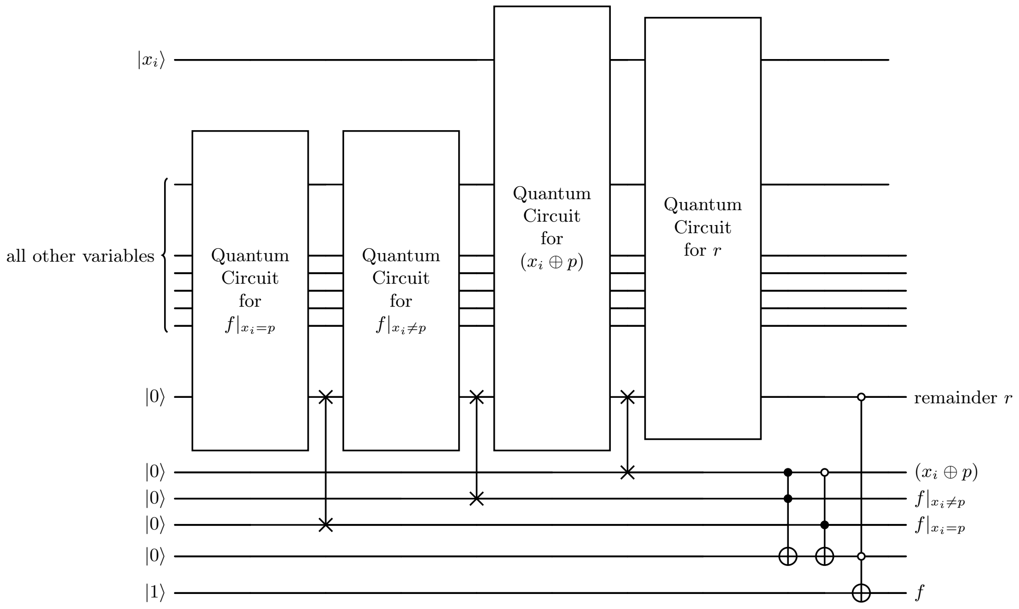


Figure 5.4: Quantum circuit based on PSOP with remainder.

Proof. Observe from Figure 5.4 that the four quantum subcircuits for $f_{|x_i=p}$, $f_{|x_i\neq p}$, $(x_i \oplus p)$, and r are combined using three additional swap gates, implemented using CNOT gates only, and three Toffoli gates. Thus, only 12 additional T-gates are required, four for each Toffoli gate [91]. ■

Algorithm 2 Quantum synthesis based on PSOP decomposition without remainder

```
 1: INPUT:
 2:    $f$      ▷ Function in SOP form depending on $n$ variables $\{x_1, \dots, x_n\}$
 3:    $x_i$                                          ▷ An input variable
 4:    $p$    ▷ Function in SOP form depending on all input variables but $x_i$
 5:    $f_{|x_i\neq p}$                ▷ Projection of $f$ onto the subspace $(x_i \oplus p)$
 6:    $f_{|x_i=p}$                ▷ Projection of $f$ onto the subspace $(\overline{x_i} \oplus p)$
 7:
 8: OUTPUT:
 9:    $Q$                                       ▷ Quantum circuit for $f$
10: OPTIONAL: Heuristic SOP minimization of $p$, $f_{|x_i\neq p}$, $f_{|x_i=p}$
11: $Q_{f_{\neq}} \leftarrow$ QuantumSynthesis($f_{|x_i\neq p}$)
12: $Q_{f_{=}} \leftarrow$ QuantumSynthesis($f_{|x_i=p}$)
13: $Q_p \leftarrow$ QuantumSynthesis($x_i \oplus p$)
14: $Q \leftarrow$ Toffoli($Q_{f_{\neq}}, Q_p$) $\oplus$ Toffoli($Q_{f_{=}}, \overline{Q_p}$)
15: return $Q$
```

Figure 5.5: Quantum synthesis based on PSOP decomposition without remainder.

Algorithm 3 Quantum synthesis based on PSOP decomposition with remainder

```
1: INPUT:
2:     $f$        ▷ Function in SOP form depending on $n$ variables $\{x_1, \ldots, x_n\}$
3:     $x_i$      ▷ An input variable
4:     $p$        ▷ Function in SOP form depending on all input variables but $x_i$
5:     $f_{|x_i \neq p}$   ▷ Projection of non-crossing products of $f$ onto subspace $(x_i \oplus p)$
6:     $f_{|x_i = p}$   ▷ Projection of non-crossing products of $f$ onto subspace $(\overline{x_i} \oplus p)$
7:     $r$        ▷ Sum (OR) of the crossing products of $f$
8: OUTPUT:
9:     $Q$        ▷ Quantum circuit for $f$
10: Optional: Heuristic SOP minimization of $p$, $f_{|x_i \neq p}$, $f_{|x_i = p}$, $r$
11: $Q_{f_{\neq}} \leftarrow$ QuantumSynthesis($f_{|x_i \neq p}$)
12: $Q_{f_{=}} \leftarrow$ QuantumSynthesis($f_{|x_i = p}$)
13: $Q_p \leftarrow$ QuantumSynthesis($x_i \oplus p$)
14: $Q_r \leftarrow$ QuantumSynthesis($r$)
15: $Q_1 \leftarrow$ Toffoli($Q_{f_{\neq}}, Q_p$) $\oplus$ Toffoli($Q_{f_{=}}, \overline{Q_p}$)
16: $Q \leftarrow 1 \oplus$ Toffoli($\overline{Q_r}, \overline{Q_1}$)
17: return $Q$
```

Figure 5.6: Quantum synthesis based on PSOP decomposition with remainder.

The overall computational cost of the proposed approach includes the cost of the projections (linear in the initial SOP of f), the cost of the optional heuristic SOP minimization of $f_{|x_i=p}$, $f_{|x_i \neq p}$, p, and r (polynomial), the cost of their quantum compilation, and the (constant) cost for combining the quantum subcircuits into a quantum circuit for f.

The cost of the standard quantum compilation would include the cost of the optional heuristic SOP minimization of f and the cost of its quantum compilation.

5.3 Experimental Results

In this section we evaluate the effectiveness of the proposed method for the quantum synthesis of PSOP-decomposed functions. We, then, present the computational results achieved by constructing PSOP expressions for Boolean functions, and comparing these expressions to their standard quantum synthesis forms. In order to assess reversible circuits derived from PSOP decomposition and compare them with standard synthesis derived circuits, we have measured their number of qubits and also evaluated their cost in terms of elementary quantum T-gates. Specifically, we mapped each MPMC Toffoli gate into elementary quantum gates. This mapping was performed based on the Clifford+T library and the algorithm detailed in [91]. Since the T-gate is considered the most expensive gate in the library, usually the cost of a Toffoli gate is expressed in the number of T-gates needed for its realization. For this reason we report the number of T-gates in the tables.

Table 5.1: Comparison between the compilation heuristic proposed in [91] (Standard synthesis) applied after ESPRESSO in the heuristic mode, and the proposed strategy with different options of p with and without remainder.

			PSOP with AND				PSOP with variable				PSOP with EXOR			
	Standard synthesis		Without remainder		With remainder		Without remainder		With remainder		Without remainder		With remainder	
Benchmark	T -count	# qubits	T -count	# qubits	T -count	# qubits	T -count	# qubits	T -count	# qubits	T -count	# qubits	T -count	# qubits
addm4	**1680**	429	2048	521	2156	548	2240	569	2244	570	2352	597	2352	597
adr4	**108**	35	264	74	120	38	216	62	216	62	352	96	440	118
amd	1244	325	**1096**	288	1128	296	1124	295	1132	297	1140	299	1140	299
apla	**404**	111	776	204	824	216	724	191	748	197	616	164	632	168
b3	1220	338	1320	363	**1200**	333	1304	359	1340	368	1272	351	1272	351
b10	1520	396	**1424**	372	1428	373	1436	375	1468	383	1552	404	1524	397
b12	280	85	344	101	260	80	260	80	**236**	74	408	117	256	79
bench	**228**	63	280	76	296	80	264	72	260	71	332	89	332	89
br1	504	138	580	157	496	136	**444**	123	504	138	524	143	524	143
br2	**348**	99	**348**	99	388	109	360	102	368	104	432	120	436	121
co14	192	62	244	75	224	70	192	62	192	62	**172**	57	180	59
dc2	**328**	90	424	114	364	100	**328**	90	**328**	91	424	114	424	115
exp	**1132**	292	1280	329	1284	330	1192	307	1208	311	1208	311	1272	327
f51m	454	121	412	111	**400**	108	508	135	508	135	420	113	416	112
fout	**820**	211	932	239	928	238	916	235	916	235	900	231	900	231
gary	1716	444	1792	463	1488	387	1612	418	1592	413	1692	438	**1440**	375
in0	1720	445	**1656**	429	1668	432	1712	443	1708	442	1720	445	1744	451
in2	1352	357	1632	427	1340	354	**1316**	348	1352	357	1328	351	1332	352
in3	1284	356	**1164**	326	1256	349	1224	341	1272	353	1280	355	1240	345
in4	1344	368	1384	378	1288	355	1372	375	1380	377	1344	368	**1320**	363
in5	1216	328	1160	314	1072	293	1052	287	**956**	264	1168	316	1088	297
in7	480	146	536	160	348	114	592	174	332	110	432	134	**316**	106
inc	**364**	98	444	118	444	118	380	102	384	103	388	104	388	104
m3	1024	264	936	242	948	245	1008	260	1052	271	**888**	230	912	236
m4	2128	540	1692	431	**1592**	406	1680	428	1836	467	1944	494	1932	491
max128	1392	356	**1196**	307	1232	316	1220	313	1260	323	1496	382	1496	382
mlp4	**1264**	324	1336	342	1308	335	1388	355	1388	355	1316	337	1316	337
newapla	136	46	196	61	72	31	140	47	**40**	23	200	62	76	32
newcpla1	336	93	464	125	248	72	280	79	320	90	364	100	**260**	75
newcpla2	228	64	216	61	**148**	45	236	66	168	49	240	67	152	46
newxcpla1	508	136	528	141	184	56	368	101	**128**	42	584	155	184	56
p3	632	167	736	193	720	189	**584**	155	592	157	628	166	620	164
p82	**292**	78	328	87	320	85	320	85	328	87	368	97	372	98
rckl	520	162	864	248	568	175	544	168	**296**	107	528	164	568	175
rd73	352	95	344	93	344	93	388	104	312	85	300	82	**180**	52
root	516	137	520	138	524	139	**452**	121	**452**	121	468	125	**452**	121
spla	**2536**	650	2720	696	2828	723	3356	855	3464	882	3308	843	3484	887
sqr6	404	108	408	109	416	111	**392**	105	**392**	105	440	117	428	114
sym10	1420	365	1080	280	1080	280	804	211	804	211	**748**	197	756	199
t1	**568**	163	792	219	792	219	572	164	592	169	816	225	816	225
t3	252	75	244	73	**228**	69	276	81	272	80	320	92	272	80
tms	744	194	828	215	840	218	704	184	744	194	**680**	178	692	181
vg2	444	136	440	135	360	116	512	153	364	116	348	112	**292**	99
x6dn	1112	317	1224	345	1180	334	**1080**	309	1092	312	1248	351	1308	366
x9dn	408	129	428	134	348	115	436	136	348	115	320	107	**256**	92
Z5xp1	456	121	380	102	392	105	512	135	520	137	356	96	**352**	95
Z9sym	828	216	804	210	788	206	692	182	692	182	**680**	179	**680**	179

All computational experiments have been run on a Intel i7-8550U CPU of 1.80GHz with 16GB of RAM. The benchmarks utilized are classical benchmarks in PLA form (classical Espresso and LGSynth'89 benchmark suite [145]). This choice is due to the fact that the computation of the function p described in [24] derives from a statistical analysis of the initial SOP (or PLA) form. The benchmarks in other classical sets (such as EPFL benchmark suite [137, 138]) are, unfortunately, not given in PLA form. We further discuss this point in the concluding section. As representative indicators of our experiments, we report only a significant subset of the functions.

The experiments have been conducted using the SOP minimization as described in the strategy depicted in Figure 5.2, using ESPRESSO [32] in the heuristic mode for the SOP synthesis. The experimental results are obtained by applying the XAG-based quantum compilation heuristic proposed in [91]. In particular, we are interested in evaluating experimentally whether this recent technique could benefit from the PSOP decomposition of the target function.

The decomposition phase is extremely fast, coherently with the linear time complexity of the corresponding algorithm [24]. Therefore, the computational times of the standard minimization and the decomposed one are extremely similar. For this reason the comparison of computational times is not interesting, and we do not report them in the tables.

In Table 5.1, the names of a significant set of benchmarks, included in our experiments, are listed in the first column. The following four columns provide details on the number of T-gates, which determine the cost, and the number of qubits required for the quantum circuits obtained from standard synthesis and PSOP expressions of the benchmarks. As can be seen, we investigate three scenarios for PSOP expressions: the projection of f with respect to p is first explored as an AND of two variables (PSOP with AND), secondly as a simple Boolean variable (PSOP with variable), and lastly as an EXOR of two variables (PSOP with EXOR). In each scenario, we examine PSOP expressions both with and without remainder.

As shown in Table 5.1, it is clear that some benchmarks experience significant advantages from PSOP expressions in terms of the number of T-gates and the number of qubits compared to standard synthesis. For instance, the benchmarks *rd73* and *sym10* achieve a significant reduction in T-gates and qubits when utilizing PSOP with EXOR (with remainder) and PSOP with EXOR (without remainder), respectively. Specifically, *rd73* shows a 49% reduction in T-gates and a 45% reduction in qubits, while *sym10* shows a 47% reduction in T-gates and an 46% reduction in qubits. However, in some cases, standard synthesis results in circuits with fewer T-gates and qubits compared with PSOP-based synthesis. For example, the *addm4* benchmark.

Overall, we can note that the best strategy seems to be the one where p is a single variable (with or without remainder). Moreover, the one that uses p as an AND gate is less useful. This is probably due to the fact that an AND gate has an expensive (in terms of T-gates) quantum representation. Meanwhile, the single variable or the EXOR gates require less expensive quantum gates.

Table 5.2 reports a subset of all the benchmarks used in our experiments. The first column lists the name of each benchmark. The next group of 2 columns detail the cost and the number of qubits for the the quantum circuits derived from standard synthesis and the best PSOP expressions of the benchmarks. Finally, the last column reports the gain in the number of T-gates.

According to the results shown in Table 5.2, it is evident that some benchmarks benefit greatly from the proposed strategy. For instance, the benchmarks *newapla* and *newxcpla1* achieve a 71% and 75% reduction in T-gates, respectively. However, the gain is much less significant for some benchmarks, such as *in0* and *in2*. In some cases, the best PSOP strategy results in circuits with a higher number of T-gates, for example *adr4* and *apla*. Overall, the T cost of the best PSOP-based quantum circuit is significantly lower than that of circuits derived from standard synthesis.

It is also crucial to minimize the number of qubits in quantum circuit design. As can be observed in Table 5.2, it is clear that the best PSOP strategy has a significant effect on some benchmarks in terms of the number of qubits. For example, the benchmark *newapla* and *newxcpla1* experiences more that 50% reduction in qubit numbers. However, the improvement is much smaller for some benchmarks like *b3* and *sqr6*. In some instances, the best PSOP strategy leads to circuits with a higher number of qubits, such as in the case of *adr4*. In general, we can see that the number of qubits in quantum circuits based on the best PSOP is notably fewer compared to the circuits obtained through standard synthesis.

In summary, we have that the proposed strategy gives better results for the 61% of the benchmarks with an average gain of about 22% in terms of T-gates, within the same time limit. Some benchmarks particularly benefit from this strategy, since their cost gain is more than the 70%.

Table 5.2: Comparison between the compilation heuristic proposed in [91] (Standard synthesis) applied after ESPRESSO in the heuristic mode, and the best solution of the proposed strategy.

	Standard synthesis		Best PSOP		Gain
Benchmark	**T -count**	**# qubits**	**T -count**	**# qubits**	**(T-gates)**
addm4	**1680**	429	2048	521	–
adr4	**108**	35	216	62	–
amd	1244	325	**1096**	288	12%
apla	**404**	111	616	164	–
b3	1220	338	**1200**	333	2%
b10	1520	396	**1424**	372	6%
b12	280	85	**236**	74	16%
bench	**228**	63	260	71	–
br1	504	138	**444**	123	12%
br2	**348**	99	**348**	99	–
co14	192	62	**172**	57	10%
dc2	**328**	90	**328**	90	–
exp	**1132**	292	1192	307	–
f51m	454	121	**400**	108	12%
fout	**820**	211	900	231	–
gary	1716	444	**1440**	375	16%
in0	1720	445	**1656**	429	4%
in2	1352	357	**1316**	348	3%
in3	1284	356	**1164**	326	9%
in5	1216	328	**956**	264	21%
in7	480	146	**316**	106	34%
inc	**364**	98	380	102	–
m3	1024	264	**888**	230	13%
m4	2128	540	**1592**	406	25%
max128	1392	356	**1196**	307	14%
mlp4	**1264**	324	1308	335	–
newapla	136	46	**40**	23	71%
newcpla1	336	93	**260**	75	23%
newcpla2	228	64	**148**	45	35%
newxcpla1	508	136	**128**	42	75%
p3	632	167	**584**	155	8%
p82	**292**	78	320	85	–
rckl	520	162	**296**	107	43%
rd73	352	95	**180**	52	49%
root	516	137	**452**	121	12%
spla	**2536**	650	2720	696	–
sqr6	404	108	**392**	105	3%
sym10	1420	365	**748**	197	47%
t1	**568**	163	572	164	–
t3	252	75	**228**	69	10%
tms	744	194	**680**	178	9%
vg2	444	136	**292**	99	34%
x6dn	1112	317	**1080**	309	3%
x9dn	408	129	**256**	92	37%
Z5xp1	456	121	**352**	95	23%
Z9sym	828	216	**680**	179	18%

5.4 Conclusion

This chapter has detailed a pre-processing procedure and a reconstruction method designed to facilitate quantum synthesis. The core of the proposed strategy lies in a PSOP (Projected Sum of Products) decomposition method, which utilizes the expression $x_i \oplus p$ to simplify the synthesis process. This approach addresses the challenges associated with quantum circuit design by leveraging a decomposition technique that refines Boolean functions into more manageable components.

The proposed PSOP decomposition method has been rigorously tested through experimental validation. Specifically, the algorithms were evaluated on decompositions where p varied as a single variable, an AND combination of variables, and an XOR combination of variables. These experiments demonstrated the effectiveness and robustness of the PSOP-based approach in practical scenarios, confirming its utility in simplifying complex Boolean functions and aiding in quantum synthesis.

The synthesis method, based on PSOP decomposition, has yielded promising results. However, it is important to note that this decomposition is specifically applicable to SOP forms. The PSOP decomposition relies on statistical analysis of the variables within the initial SOP representation, necessitating that the input Boolean function be expressed in a PLA format or another two-level logic representation. This requirement highlights a limitation of the current approach, as it may not directly apply to Boolean functions represented in alternative formats.

Part II

Multivalued Reversible Designs for Quantum Circuits

Chapter 6

Preliminaries on Multivalued Reversible Logic

This chapter explains the fundamental concepts of multi-valued reversible logic, with a focus on ternary and quaternary logic systems. Additionally, an overview of ternary and quaternary Galois fields, as well as reversible gates, is provided, setting the stage for their application in subsequent discussions. This foundational knowledge is crucial for understanding the advanced topics explored in the following chapters.

6.1 The Concept of Ternary Reversible Logic

Quantum ternary logic is an essential concept within the broader field of quantum multi-valued logic, gaining significant attention due to its potential applications in next-generation computing technologies. Unlike traditional binary logic, which operates on two states (0 and 1), ternary logic utilizes three distinct states, making it a prime candidate for advanced computational systems, such as ternary quantum computers and digital filtering technologies. This shift from binary to ternary logic offers several notable advantages, including reduced power consumption, enhanced fault tolerance, and increased security, especially in the field of quantum cryptography [10, 30, 133]. These advantages make ternary logic highly attractive for future quantum information systems and secure communications, where classical binary methods are less efficient and robust.

In multi-valued systems, ternary logic is distinguished by its interesting noise immunity [46], and it is considered the most economical radix, as determined by examining the width and depth of number representation [63, 80]. Hurst [63] demonstrated that circuit complexity (C) associated with processing specific quantitative data (N) can be expressed as:

$$C = K(R \cdot d), \tag{6.1}$$

where R is the number of basis states, d is the number of signals, and K is the proportional coefficient. Additionally, $d = \log_R N$. It can be concluded that C reaches its lowest point when R is approximately equal to $e \approx 2.718$. Considering that R must be an integer, 3 is the nearest integer to e, thus confirming ternary logic's efficiency. Table 6.1 provides the relationship between R and the associated circuit complexity C.

R	$C = K(R \cdot d)$
2	100.0
3	94.6
4	100.0
5	107.7
10	150.5

Table 6.1: The relationship between values of R and C

In comparison to binary circuits, where $R = 2$, the complexity of ternary circuits ($R = 3$) is lower. The complexity C can be further refined using the following equation:

$$K = \frac{50}{\log_2 N}, \tag{6.2}$$

and for cases where $R \neq 2$, the equation becomes:

$$C = 50R\frac{\ln(2)}{\ln(R)}. \tag{6.3}$$

These equations indicate that $R = 3$ is the optimal radix for reducing circuit complexity [42].

At the core of quantum ternary logic is the concept of the "qutrit," analogous to the qubit in binary quantum computing, serving as the basic unit of quantum information in a ternary system. A qutrit can exist in three possible states, denoted as $|0\rangle$, $|1\rangle$, and $|2\rangle$, mathematically represented as three-dimensional vectors:

$$|0\rangle = \begin{bmatrix}1\\0\\0\end{bmatrix}, \quad |1\rangle = \begin{bmatrix}0\\1\\0\end{bmatrix}, \quad |2\rangle = \begin{bmatrix}0\\0\\1\end{bmatrix}.$$

A qutrit can also exist in a superposition of these basis states, expressed as:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle + \gamma|2\rangle,$$

where α, β, and γ are complex probability amplitudes associated with each state. The normalization condition ensures that the total probability of measuring the system in any state sums to one:

$$|\alpha|^2 + |\beta|^2 + |\gamma|^2 = 1.$$

Quantum computers based on ternary logic are found to be 37% more compact than binary logic systems [61]. Additional advantages of quantum ternary logic include increased security [10, 30], improved quantum information processing efficiency [55], greater density of stored information [33], enhanced error tolerance [81], simplified interconnection complexity, and reduced power consumption [33, 96].

In the design of ternary systems, researchers have also explored reversible ternary logic, which holds promise for quantum computing structures [106]. Reversible ternary logic offers numerous advantages, such as the ability to design efficient quantum circuits with minimized power loss. Balanced and unbalanced ternary arithmetic functions are often used in such systems, where the balanced representation (with values $\hat{1}$, 0, and 1) proves advantageous for efficient arithmetic operations [46]. Balanced ternary representations simplify operations like inversion, partial product generation, and carry ripple structures, making them highly suitable for arithmetic circuit design.

Moreover, in quantum ternary circuits, metrics like quantum cost, garbage outputs, and constant inputs are critical factors in the design and performance optimization of these circuits [9, 99], which are explained as follows:

- Quantum cost refers to the number of ternary primitive gates, which are 1-qutrit Shift gates and 2-qutrit Muthukrishnan-Stroud gates, required to realize the circuit.
- The number of garbage outputs indicates the outputs which are generated to preserve one-to-one mappings but have unimportant values.
- The number of constant inputs signifies the number of inputs which must be maintained constant in order to synthesize the specified logic function.

Minimization of the above mentioned features increases the efficiency of ternary quantum reversible logic design [9, 99].

In summary, quantum ternary logic provides a more compact, efficient, and secure framework for quantum computing compared to traditional binary logic. The optimal complexity of ternary circuits and their favorable characteristics in noise immunity, arithmetic operations, and quantum cryptography underscore their growing importance in the development of future quantum technologies.

6.1.1 Ternary Galois Field Logic

Ternary logic can be represented using the Galois Field GF3. This field can be applied in both unbalanced and balanced ternary systems. In unbalanced ternary logic, the Galois Field GF3 consists of the elements $\{0, 1, 2\}$, with operations defined modulo 3 [70, 101]. The addition and multiplication tables for this system are provided in Tables 6.2 and 6.3.

Table 6.2: Truth table of GF3 addition operation (unbalanced).

$\oplus$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

Table 6.3: Truth table of GF3 multiplication operation (unbalanced).

$\odot$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

In balanced ternary logic, the Galois Field GF3 consists of the elements $T = \{\hat{1}, 0, 1\}$, where $\hat{1}$ represents -1. This balanced representation offers various advantages in computation. The addition and multiplication tables for balanced ternary logic are shown Tables 6.4 and 6.5.

Table 6.4: Truth table of GF3 addition operation (balanced).

$\bigoplus$	$\hat{1}$	0	1
$\hat{1}$	1	$\hat{1}$	0
0	$\hat{1}$	0	1
1	0	1	$\hat{1}$

Table 6.5: Truth table of GF3 multiplication operation (balanced).

$\bigodot$	$\hat{1}$	0	1
$\hat{1}$	1	0	$\hat{1}$
0	0	0	0
1	$\hat{1}$	0	1

The following axioms rules are also derived according to the tables provided above, which demonstrate the behavior of addition and multiplication operations in the algebraic structure [34].This field can be applied in both unbalanced and balanced ternary systems.

Addition:

(A1) Associative Law: $a \oplus (b \oplus c) = (a \oplus b) \oplus c$

(A2) Commutative Law: $a \oplus b = b \oplus a$

(A3) Identity Element: There is an element 0 such that $a \oplus 0 = a$ for all a

(A4) Additive Inverse: For any a, there is an element $(-a)$ such that $a \oplus (-a) = 0$

Multiplication:

(M1) Associative Law: $a \odot (b \odot c) = (a \odot b) \odot c$

(M2) Commutative Law: $a \odot b = b \odot a$

(M3) Identity Element: There is an element 1 (not equal to 0) such that $a \odot 1 = a$ for all a

(M4) Multiplicative Inverse: For any $a \neq 0$, there is an element a^{-1} such that $a \odot a^{-1} = 1$

Distribution:

(D) Distributive Law: $a \odot (b \oplus c) = (a \odot b) \oplus (a \odot c)$

6.1.2 Ternary Shift Gates

In ternary reversible logic, any transformation of the qutrit is expressed using a 3×3 unitary matrix, as demonstrated in Figure 6.1. Each column or row in these matrices contains three elements which are 0 or 1. The transformations are used to shift or exchange elements within the matrices [68, 61]. $Z(+0)$ is an initial state and each column or row of the matrix shows 0, 1 or 2 values. $Z(+1)$ shifts the qutrit state by 1 and $Z(+2)$ shifts the qutrit state by 2. $Z(12)$ exchanges the qutrit states 1 and 2, $Z(01)$ exchanges the qutrit states 0 and 1, and $Z(02)$ exchanges the qutrit states 0 and 2.

Each of these unitary matrices can be realized as a 1-qutrit Shift gate. These gates are uncontrollable, it means that there is no controlling input. The symbolic

$$Z(0) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad Z(+1) = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad Z(+2) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

$$Z(12) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad Z(01) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad Z(02) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Figure 6.1: Unbalanced representation of 1-qutrit permutative transforms.

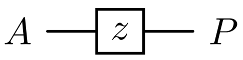


Figure 6.2: The symbolic representation of ternary shift gates.

representation of ternary shift gates can be observed in Figure 6.2. As depicted, A is the input to the gate, and P is the output. P corresponds to the Z transform of A. The quantum cost associated with these gates is 1 [68]. Table 6.6 shows the truth table of ternary Shift gates, named Identity, Single Shift, Dual Shift, Self Shift, Self Single Shift and Self Dual Shift, respectively. As shown in Table 6.7, these gates have unitary inverse gates that return the inputs to their original values [68, 61]. As can be observed in this table, Z(01), Z(02) and Z(12) are self-inverse gates.

Table 6.6: Truth table of ternary Shift gates.

Input	Z(0)	Z(+1)	Z(+2)	Z(12)	Z(01)	Z(02)
0	0	1	2	0	1	2
1	1	2	0	2	0	1
2	2	0	1	1	2	0

Table 6.7: Inverse gates of ternary Shift gates.

Gates	Z(+1)	Z(+2)	Z(12)	Z(01)	Z(02)
Inverse gates	Z(+2)	Z(+1)	Z(12)	Z(01)	Z(02)

6.1.3 Ternary Muthukrishnan and Stroud Gate

Muthukrishnan and Stroud introduced a family of 2-qudit ternary gates that can be theoretically implemented using quantum technologies, such as liquid ion trap technology, as an elementary gate [106]. The symbolic representation of balanced ternary Muthukrishnan and Stroud gate can be observed in Figure 6.3. As depicted, this gate is controllable, A and B are the inputs to the gate, A is controlling input and B is controlled input. P and Q are the outputs. The output

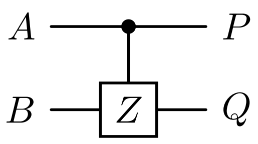


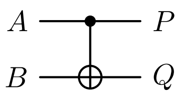
Figure 6.3: The symbolic representation of ternary Muthukrishnan and Stroud gate.

P corresponds to the input A and the output Q corresponds to the Z transform of B where $Z \in \{0, +1, +2, 12, 01, 02\}$, but only when the input A is equal to 2, otherwise the output P is equal to the input B. The quantum cost associated with this gate is 1 [101].

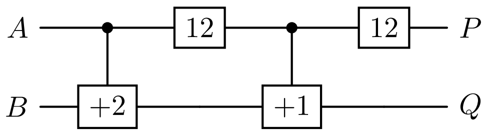
6.1.4 Ternary Controlled Feynman Gate

Figure 6.4a shows the symbolic representation of the ternary Feynman gate. In this gate, the inputs are A and B, and the outputs are P and Q, where $P = A$ and $Q = A \oplus B$. This gate can be implemented using Muthukrishnan-Stroud gates and 1-qutrit permutation gates, as depicted in Figure 6.4b. The evaluation matrix of the ternary Feynman gate can be expressed as discussed in [2, 100].

H.A. Khan in [68] designed a ternary Controlled Feynman gate and demonstrated its realization using 2-qutrit Muthukrishnan-Stroud gates. Figure 6.5a illustrates the symbolic representation of the ternary Controlled Feynman gate. This gate has inputs A, B, and C, and outputs P, Q, and R. For this gate, $P = A$, $Q = B$, and $R = B \oplus C$ when $A \neq 2$; otherwise, $R = C$. This gate can be realized using Muthukrishnan-Stroud gates and 1-qutrit permutation gates, as shown in Figure 6.5b. The quantum cost for this gate is 4. However, if input B is not required at the output Q, the fourth 2-qutrit Muthukrishnan-Stroud gate can be removed, reducing the quantum cost to 3.

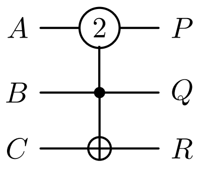


(a) Symbol.

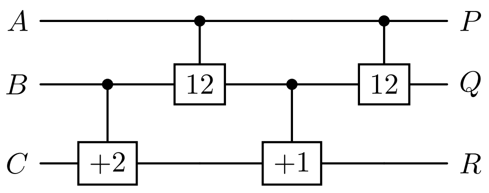


(b) The realization using M-S and Shift gates.

Figure 6.4: The ternary reversible Feynman gate.



(a) Symbol.



(b) The realization using M-S and Shift gates.

Figure 6.5: The ternary reversible controlled Feynman gate.

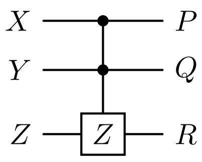


Figure 6.6: The graphical representation of Toffoli gate.

6.1.5 Ternary Toffoli Gate

Khan and Perkowski in [79] present a 3-qutrit generalized Toffoli gate along with its realization in ion-trap technology. The symbolic representation of this gate can be observed in Figure 6.6. In this gate, X, Y and Z are the inputs and P, Q and R are the outputs. The outputs P and Q are equal to the input X and Y, respectively. The output R, which is the target output, is equal to the Z transform of the input Z when X and Y are equal 2, where Z can be $+1, +2, 01, 02, 12$. Otherwise, the output R is equal to the input Z. The quantum cost associated with this gate is $5 + (2 \times \{\text{number of controlling values which are not } 2\})$.

6.2 The Concept of Quaternary Reversible Logic

Quaternary quantum logic is also a type of multiple-valued quantum logic that has been advocated by many researchers. In quaternary quantum systems, the unit of memory (information) is termed a *qudit* (quantum digit) and can be denoted by $|0\rangle$, $|1\rangle$, $|2\rangle$, or $|3\rangle$, represented by the following 4×1 vectors [136, 102]:

$$|0\rangle = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \quad |1\rangle = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \quad |2\rangle = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \quad |3\rangle = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

Qudits exist in a linear superposition of basis states. This technique is called *superposition*. In quaternary quantum logic, the superposition is denoted as:

$$\psi = \alpha|0\rangle + \beta|1\rangle + \gamma|2\rangle + \delta|3\rangle$$

This can also be written as a vector:

$$\psi = (\alpha, \beta, \gamma, \delta)$$

where α, β, γ, and δ are complex numbers. The probability of measuring state $|0\rangle$ is $|\alpha|^2$, state $|1\rangle$ is $|\beta|^2$, state $|2\rangle$ is $|\gamma|^2$, and state $|3\rangle$ is $|\delta|^2$. The sum of these probabilities must satisfy the normalization condition:

$$|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$$

In general, an N-qudit system represents 4^N distinct computational basis states. These basis states can be depicted as:

$$|000\cdots0\rangle, |000\cdots1\rangle, \ldots, |333\cdots3\rangle$$

All possible states of the N-qudit system are indicated by the tensor product of N qudits [109].

In quaternary reversible circuit design, several important figures of merit determine the efficiency of the circuit. Minimizing these parameters is an important consideration in the design of quaternary quantum reversible logic circuits.

- Quantum cost refers to the cost of a circuit, calculated by the number of quaternary 1-qudit gates.
- The number of constant inputs refers to the number of inputs assigned constant values such as 0, 1, 2, or 3 in the synthesis of a given logic function.
- The number of garbage outputs refers to the number of unused outputs in the synthesis of the given logic function.

6.2.1 Quaternary Galois Field Logic

The quaternary Galois field, denoted as GF(4), consists of the set of elements $T = \{0, 1, 2, 3\}$ along with two binary operations: addition ($\oplus$) and multiplication ($\odot$) [75]. The operations within GF(4) follow the rules outlined in Tables 6.8 and 6.9. These operations satisfy the following axioms [24]:

Addition:

(A1) Associative law: $a \oplus (b \oplus c) = (a \oplus b) \oplus c$

(A2) Commutative law: $a \oplus b = b \oplus a$ (A3) Identity element: $a \oplus 0 = a$ for all a

(A4) Inverse element: For any a, there exists an element $(-a)$ such that $a \oplus (-a) = 0$

Multiplication:

(M1) Associative law: $a \odot (b \odot c) = (a \odot b) \odot c$

(M2) Commutative law: $a \odot b = b \odot a$

(M3) Identity element: $a \odot 1 = a$ for all a

(M4) Inverses: For any $a \neq 0$, there exists an element a^{-1} such that $a \odot a^{-1} = 1$

Distributive Law:

(D) Distributive law: $a \odot (b \oplus c) = (a \odot b) \oplus (a \odot c)$

Both addition and multiplication operations are commutative and associative. Furthermore, the multiplication operation is distributive over addition. The truth tables for addition ($\oplus$) and multiplication ($\odot$) in GF(4) are given in Tables 6.8 and 6.9.

Table 6.8: Truth table of GF(4) addition operation.

$\oplus$	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

Table 6.9: Truth table of GF(4) multiplication operation.

$\odot$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

6.2.2 Quaternary Shift Gates

In quaternary systems, there are $4! = 24$ possible permutations of the set $\{0, 1, 2, 3\}$, each represented by a 4×4 unitary matrix. These quaternary 1-qudit unitary permutative transformations are illustrated in Figure 6.7 [76]. These transformations can be implemented as quaternary 1-qudit gates using quantum technology [106].

The symbolic representation of quaternary 1-qudit gates is shown in Figure 6.8. In this representation, the input to the gate is A, and the output P is obtained by applying a permutation Z-transform to A. The corresponding truth table for quaternary 1-qudit gates is provided in Table 6.10 [75].

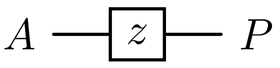


Figure 6.8: The symbolic representation of quaternary shift gates.

$$Z(+0)=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\quad Z(+1)=\begin{bmatrix}0&1&0&0\\1&0&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}\quad Z(+2)=\begin{bmatrix}0&0&1&0\\0&0&0&1\\1&0&0&0\\0&1&0&0\end{bmatrix}\quad Z(+3)=\begin{bmatrix}0&0&0&1\\0&0&1&0\\0&1&0&0\\1&0&0&0\end{bmatrix}$$

$$Z(123)=\begin{bmatrix}1&0&0&0\\0&0&0&1\\0&1&0&0\\0&0&1&0\end{bmatrix}\quad Z(013)=\begin{bmatrix}0&0&0&1\\1&0&0&0\\0&0&1&0\\0&1&0&0\end{bmatrix}\quad Z(021)=\begin{bmatrix}0&1&0&0\\0&0&1&0\\1&0&0&0\\0&0&0&1\end{bmatrix}\quad Z(032)=\begin{bmatrix}0&0&1&0\\0&1&0&0\\0&0&0&1\\1&0&0&0\end{bmatrix}$$

$$Z(132)=\begin{bmatrix}1&0&0&0\\0&0&1&0\\0&0&0&1\\0&1&0&0\end{bmatrix}\quad Z(012)=\begin{bmatrix}0&0&1&0\\1&0&0&0\\0&1&0&0\\0&0&0&1\end{bmatrix}\quad Z(023)=\begin{bmatrix}0&0&0&1\\0&1&0&0\\1&0&0&0\\0&0&1&0\end{bmatrix}\quad Z(031)=\begin{bmatrix}0&1&0&0\\0&0&0&1\\0&0&1&0\\1&0&0&0\end{bmatrix}$$

$$Z(23)=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}\quad Z(01)=\begin{bmatrix}0&1&0&0\\1&0&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\quad Z(0213)=\begin{bmatrix}0&0&0&1\\0&0&1&0\\1&0&0&0\\0&1&0&0\end{bmatrix}\quad Z(0312)=\begin{bmatrix}0&0&1&0\\0&0&0&1\\0&1&0&0\\1&0&0&0\end{bmatrix}$$

$$Z(12)=\begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix}\quad Z(0132)=\begin{bmatrix}0&0&1&0\\1&0&0&0\\0&0&0&1\\0&1&0&0\end{bmatrix}\quad Z(0231)=\begin{bmatrix}0&1&0&0\\0&0&0&1\\1&0&0&0\\0&0&1&0\end{bmatrix}\quad Z(03)=\begin{bmatrix}0&0&0&1\\0&1&0&0\\0&0&1&0\\1&0&0&0\end{bmatrix}$$

$$Z(13)=\begin{bmatrix}1&0&0&0\\0&0&0&1\\0&0&1&0\\0&1&0&0\end{bmatrix}\quad Z(0123)=\begin{bmatrix}0&0&0&1\\1&0&0&0\\0&1&0&0\\0&0&1&0\end{bmatrix}\quad Z(02)=\begin{bmatrix}0&0&1&0\\0&1&0&0\\1&0&0&0\\0&0&0&1\end{bmatrix}\quad Z(0321)=\begin{bmatrix}0&1&0&0\\0&0&1&0\\0&0&0&1\\1&0&0&0\end{bmatrix}$$

Figure 6.7: Representation of 1-qudit permutative transforms.

Table 6.10: Truth table of quaternary 1-qudit gates.

Input	Z(+0)	Z(+1)	Z(+2)	Z(+3)	Z(123)	Z(013)
0	0	1	2	3	0	1
1	1	0	3	2	2	3
2	2	3	0	1	3	2
3	3	2	1	0	1	0
Input	Z(021)	Z(032)	Z(132)	Z(012)	Z(023)	Z(031)
0	2	3	0	1	2	3
1	0	1	3	2	1	0
2	1	0	1	0	3	2
3	3	2	2	3	0	1
Input	Z(23)	Z(01)	Z(0213)	Z(0312)	Z(12)	Z(0132)
0	0	1	2	3	0	1
1	1	0	3	2	2	3
2	3	2	1	0	1	0
3	2	3	0	1	3	2
Input	Z(0231)	Z(03)	Z(13)	Z(0123)	Z(02)	Z(0321)
0	2	3	0	1	2	3
1	0	1	3	2	1	0
2	3	2	2	3	0	1
3	1	0	1	0	3	2

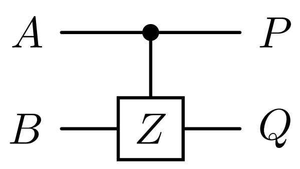


Figure 6.9: The symbolic representation of quaternary Muthukrishnan and Stroud gate.

Each quaternary 1-qudit gate has a corresponding unitary inverse gate, which can restore the input signal for reuse in the circuit. If two quaternary 1-qudit gates, x and y, are cascaded in such a way that the output of gate y restores the input signal of gate x, then gate y is considered the inverse of gate x. Among the quaternary 1-qudit gates, the gates $Z(+1)$, $Z(+2)$, $Z(+3)$, $Z(23)$, $Z(01)$, $Z(12)$, $Z(03)$, $Z(13)$, and $Z(02)$ are self-inverse.

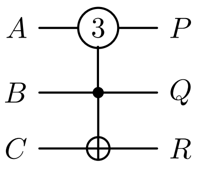
6.2.3 Quaternary Muthukrishnan–Stroud Gate

Muthukrishnan and Stroud proposed a family of 2-qudit multiple-valued gates, known as Muthukrishnan-Stroud (M-S) gates, which can be realized using quantum technologies like liquid ion traps [106]. The symbolic representation of quaternary 2-qudit M-S gates is shown in Figure 6.9. In this gate, the inputs are A (control input) and B (target input). The outputs are P and Q, where $P = A$ and Q is the result of applying a Z-transform to B if $A = 3$.

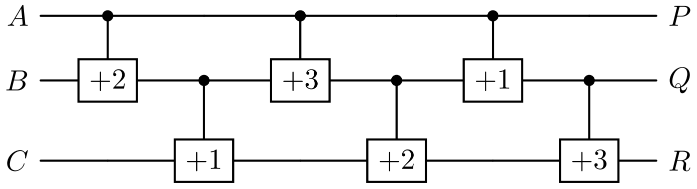
6.2.4 Quaternary Controlled Feynman Gate

The quaternary Controlled Feynman gate is a 3-input, 3-output gate that maps the inputs (A, B, C) to the outputs (P=A, Q=B, $R = B \oplus C$ if A = 3; otherwise R = C). The inputs are A, B, and C, while the outputs are P, Q, and R [76].

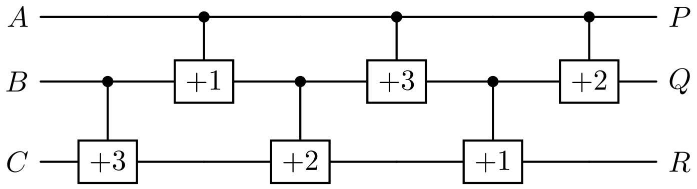
Figure 6.10a shows the graphical representation of the quaternary Controlled Feynman gate. Figures 6.10b and 6.10c present different implementations of this gate using M-S gates. This gate has a quantum cost of 6. Based on the second implementation in Figure 6.10c, the last 2-qutrit M-S gate (+3) can be removed if the input B is not required at the output Q. In this case, the quantum cost is reduced to 5, and the output Q becomes equal to B+2 if A=3 [76].



(a) Symbol.



(b) The first realization using M-S and Shift gates.



(c) The second realization using M-S and Shift gates.

Figure 6.10: The quaternary reversible Controlled Feynman gate.

Chapter 7

Novel Qutrit Circuit Design for Multiplexer, Demultiplexer, and Decoder

This chapter addresses ternary logic, which reduces circuit width, offering a potential solution to current limitations in quantum technologies. In this chapter, we propose two approaches for quantum ternary decoder circuits, followed by the development of quantum ternary multiplexer and demultiplexer circuits that leverage the ternary decoder design. The chapter emphasizes techniques for lowering quantum cost and compares the proposed circuits to existing designs, highlighting significant improvements. These designs are realizable using macro-level ternary gates, specifically ion-trap-based ternary 2-qutrit Muthukrishnan–Stroud and 1-qutrit permutation gates, showcasing their potential in advancing quantum computing.

7.1 Introduction

In the realm of digital logic design, the implementation of combinational logic circuits is a well-established practice, traditionally achieved using binary systems with multiplexers and basic logic gates. However, this capability extends to ternary logic systems as well [65]. Ternary logic, which utilizes three distinct states instead of two, offers a broader range of operational possibilities and can be effectively implemented using ternary multiplexers and gates. This chapter delves into the synthesis and optimization of quantum ternary circuits, focusing specifically on ternary quantum decoders, multiplexers, and demultiplexers.

The literature provides a comprehensive overview of various quantum ternary circuit implementations for different computational units within quantum systems. These include fundamental components such as full adders, half adders, parallel adders/subtractors, subtractors, multipliers, decoders, encoders, demultiplexers, and multiplexers [61, 7, 90, 113, 114, 105, 101, 6, 104, 136, 64]. Among these, decoders, multiplexers, and demultiplexers are particularly crucial as they serve as core sub-circuits for constructing ternary quantum oracles and other ternary system designs, including communication systems, computer memory, and arithmetic logic units [69].

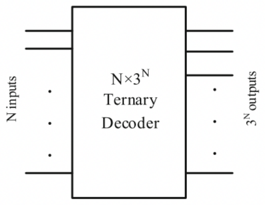


Figure 7.1: Block diagram of $N \times 3^N$ ternary decoder circuit.

Recent studies have addressed the implementation of quantum ternary decoders, multiplexers, and demultiplexers, providing valuable insights into their quantum cost and efficiency. Recall that quantum cost refers to the number of ternary primitive gates, which are 1-qutrit Shift gates and 2-qutrit Muthukrishnan-Stroud gates, required to realize the circuit. A quantum ternary decoder with a quantum cost of 57 is detailed in [78]. Similarly, the work presented in [68] introduces quantum ternary multiplexers and demultiplexers with a quantum cost of 102. Despite these contributions, there remains significant scope for enhancing the efficiency of these circuits.

This chapter focuses on the development and optimization of more efficient quantum ternary decoders, multiplexers, and demultiplexers compared to the existing designs [78, 68]. Our approach emphasizes minimizing critical parameters such as quantum cost, circuit depth, number of garbage outputs, and number of constant inputs. By optimizing these parameters, we aim to achieve superior efficiency in quantum ternary logic design, aligning with the objectives of recent advancements in the field [112, 9, 99]. The results presented in this chapter were published in the Quantum Information Processing journal [135].

The structure of this chapter is organized as follows: Section 7.2 details the realization of our proposed quantum reversible ternary decoder, multiplexer, and demultiplexer circuits. In Section 7.3, we assess the performance of these circuits based on the aforementioned parameters. Finally, Section 7.4 offers concluding remarks and summarizes the contributions of this study.

7.2 The Proposed Quantum Ternary Circuit

In this section, we first propose two quantum ternary decoders. The quantum ternary multiplexer and demultiplexer circuits are then presented. The following subsections describe these designs in detail.

7.2.1 Ternary Reversible Decoder

One of the important ternary combinational logic circuits is the ternary decoder, which converts ternary information from the N inputs to 3^N unique outputs. Figure 7.1 illustrates the block diagram of the $N \times 3^N$ ternary decoder circuit. Table 7.1 shows the truth table of a 2×9 ternary decoder with active-2 output. In this table, A and B are the input variables, whereas $D_0, D_1, D_2, D_3, D_4, D_5, D_6, D_7$, and D_8 are the output variables.

For each combination of inputs, only one output line will be activated. In this case, the circuit is active-2, meaning that if the output line is 2, the line is ON;

Table 7.1: 2×9 ternary decoder truth table (x denotes the result when the output line is off)

Inputs	**Outputs**								
AB	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8
00	2	x	x	x	x	x	x	x	x
01	x	2	x	x	x	x	x	x	x
02	x	x	2	x	x	x	x	x	x
10	x	x	x	2	x	x	x	x	x
11	x	x	x	x	2	x	x	x	x
12	x	x	x	x	x	2	x	x	x
20	x	x	x	x	x	x	2	x	x
21	x	x	x	x	x	x	x	2	x
22	x	x	x	x	x	x	x	x	2

otherwise, it is OFF. According to the truth table, only one of the outputs will be equal to 2 for any given input combination. The outputs are described as follows:

$$\begin{aligned} D_0 &= A_0B_0, & D_1 &= A_0B_1, & D_2 &= A_0B_2, \\ D_3 &= A_1B_0, & D_4 &= A_1B_1, & D_5 &= A_1B_2, \\ D_6 &= A_2B_0, & D_7 &= A_2B_1, & D_8 &= A_2B_2. \end{aligned}$$

The outputs x in Table 7.1 can be either 0 or 1. In both cases, the corresponding output line will remain inactive.

In this chapter, we present two approaches to constructing the proposed quantum ternary decoder.

The Primary Quantum Ternary Decoder Design Approach

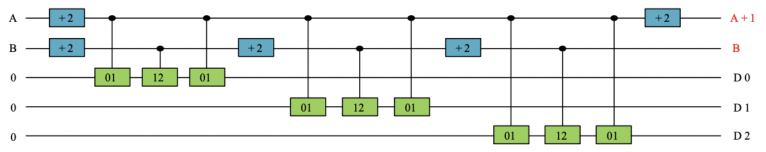
In the Primary Quantum Ternary Decoder Design Approach (PQTDA), only one of the outputs will be equal to 2 for a given input combination, while the remaining outputs will be equal to 0 (i.e., $x = 0$). To enhance comprehension, the operation of the quantum ternary decoder is first demonstrated in three parts, as shown in Fig. 7.2a, Fig. 7.2b, and Fig. 7.2c, followed by the complete design in Fig. 7.2d.

The realization of the first part is depicted in Fig. 7.2a, consisting of five 1-qutrit permutation gates and nine 2-qutrit Muthukrishnan–Stroud gates. In this stage, if the inputs A and B are equal to 00, only output D_0 will be equal to 2. Similarly, when A and B are 01, only output D_1 will be 2, and when A and B are 02, output D_2 will be 2.

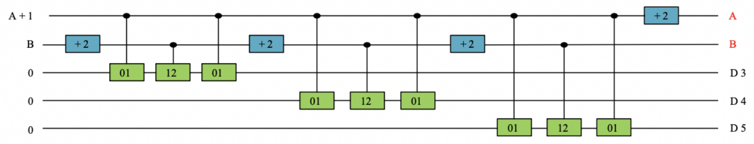
The second part of the decoder is shown in Fig. 7.2b, utilizing five 1-qutrit permutation gates and nine 2-qutrit Muthukrishnan–Stroud gates. In this stage, when A and B are equal to 10, only output D_3 will be equal to 2. If A and B are 11, only output D_4 will be 2, and when A and B are 12, output D_5 will be 2.

The realization of the final part is shown in Fig. 7.2c. In this stage, when inputs A and B are equal to 20, only output D_6 will be equal to 2. Similarly, when A and B are 21, output D_7 will be 2, and if A and B are 22, output D_8 will be 2.

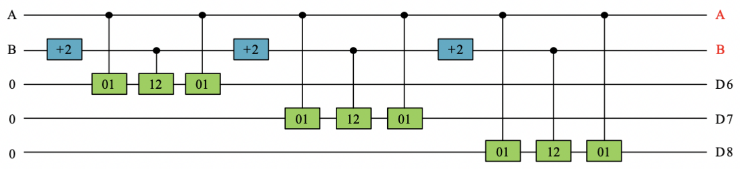
The complete realization of the proposed quantum ternary decoder is shown in Fig. 7.2d. The proposed design uses twelve 1-qutrit permutation gates and twenty-seven 2-qutrit Muthukrishnan–Stroud gates, resulting in a quantum cost of 39. The outputs for various input combinations are shown in Table 7.2. The depth of the proposed quantum ternary decoder circuit, as shown in Fig. 7.2d, is 30.



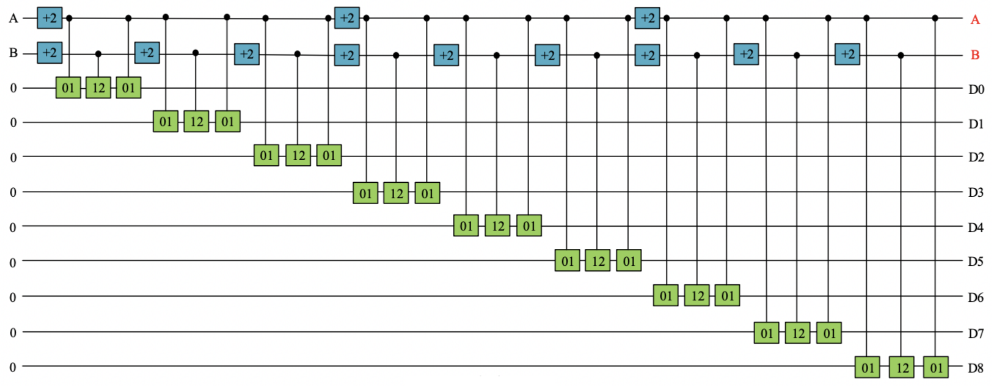
(a) The realization of the first part.



(b) The realization of the second part.



(c) The realization of the third part.



(d) Complete realization of PQTDA using 2-qutrit M–S and 1-qutrit permutation gates.

Figure 7.2: The proposed PQTDA design.

Optimized Quantum Ternary Decoder Design Approach

In the Optimized Quantum Ternary Decoder Design Approach (OQTDA), we propose a ternary decoder with a lower quantum cost. In this optimized design, all unselected outputs are allowed to take on the value of 1 or 0, while only the selected output is set to 2. To implement this circuit, we use 1-qutrit permutation gates and 2-qutrit Muthukrishnan–Stroud gates. The realization of the proposed circuit is illustrated in Fig. 7.3.

As shown in Fig. 7.3, the circuit utilizes nine 1-qutrit permutation gates and

Table 7.2: PQTDA truth table.

Inputs	**Outputs**								
AB	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8
00	2	0	0	0	0	0	0	0	0
01	0	2	0	0	0	0	0	0	0
02	0	0	2	0	0	0	0	0	0
10	0	0	0	2	0	0	0	0	0
11	0	0	0	0	2	0	0	0	0
12	0	0	0	0	0	2	0	0	0
20	0	0	0	0	0	0	2	0	0
21	0	0	0	0	0	0	0	2	0
22	0	0	0	0	0	0	0	0	2

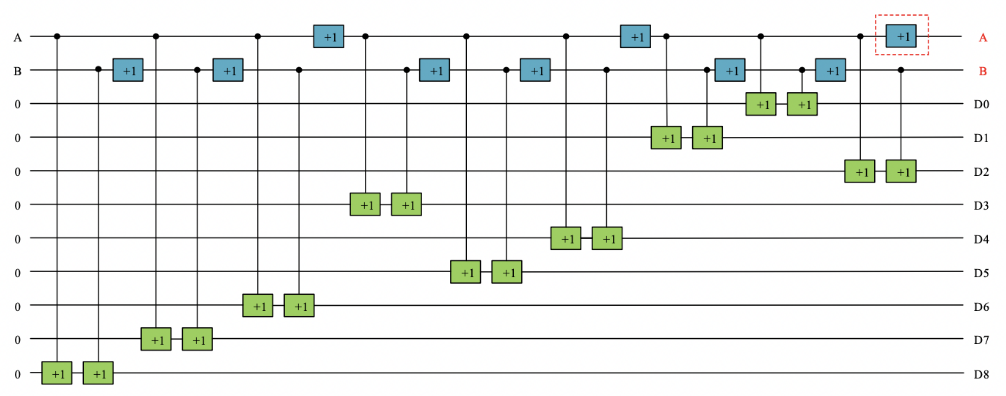


Figure 7.3: OQTDA realization

eighteen 2-qutrit Muthukrishnan–Stroud gates. The quantum cost of this optimized design is 27, representing a significant improvement in terms of both implementation cost and circuit depth compared to the PQTDA. The outputs for different input combinations are provided in Table 7.3. The depth of the proposed quantum ternary decoder circuit, as shown in Fig. 7.3, is 18.

Table 7.3: OQTDA truth table.

Inputs	**Outputs**								
AB	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8
00	2	1	1	1	0	0	1	0	0
01	1	2	1	0	1	0	0	1	0
02	1	1	2	0	0	1	0	0	0
10	1	0	0	2	1	1	1	0	0
11	0	1	0	1	2	1	0	1	0
12	0	0	1	1	1	2	0	0	1
20	1	0	0	1	0	0	2	1	1
21	0	1	0	0	1	0	1	2	1
22	0	0	1	0	0	1	1	1	2

It is worth noting that in OQTDA, the first output line is restored to input A. If restoring this input at the output is not necessary, we can remove the final $+1$ gate, which is marked by a red dashed line in Fig. 7.3. By doing so, the quantum cost is further reduced to 26, and the output on the first line becomes $A + 2$.

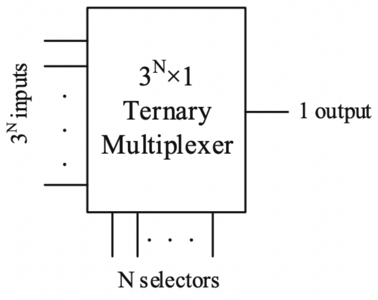


Figure 7.4: Block diagram of $3^N \times 1$ ternary multiplexer circuit.

7.2.2 Ternary Reversible Multiplexer

A multiplexer circuit has one output, multiple inputs, and selectors. The selectors specify which inputs are gated to the output. A ternary multiplexer takes N ternary numbers as selectors and 3^N ternary numbers as input, providing the output as a number. The block diagram of the $3^N \times 1$ ternary multiplexer circuit is illustrated in Fig. 7.4. Table 7.4 shows the truth table of a 9×1 ternary multiplexer, where A and B are selectors, $I_0, I_1, I_2, I_3, I_4, I_5, I_6, I_7$, and I_8 are the inputs, and the output is depicted as O. For a given selector combination, only the selected input will be gated to the output.

Table 7.4: 9×1 ternary multiplexer truth table.

Selectors (AB)	Output (O)
00	I_0
01	I_1
02	I_2
10	I_3
11	I_4
12	I_5
20	I_6
21	I_7
22	I_8

According to Table 7.4, the output can be expressed as:

$$O = (A_0B_0)I_0 + (A_0B_1)I_1 + (A_0B_2)I_2 + (A_1B_0)I_3 + (A_1B_1)I_4 + (A_1B_2)I_5$$
$$+(A_2B_0)I_6 + (A_2B_1)I_7 + (A_2B_2)I_8.$$

To construct the ternary multiplexer circuit, we use the proposed quantum ternary decoder and ternary controlled Feynman gates. The realization of the proposed multiplexer is shown in Fig. 7.5. In this circuit, input restoration is required, where PQTDA, nine ternary controlled Feynman gates, and sixty-three 2-qutrit Muthukrishnan–Stroud gates are used. In this implementation, the outputs preserve their corresponding inputs by utilizing nine Muthukrishnan–Stroud gates, highlighted in red in the figure, leading to a total quantum cost of 75.

The quantum cost can be further reduced to 53 if input restoration is not necessary, which can be achieved by using OQTDA and removing the 1-qutrit permutation gates shown in red in Fig. 7.5.

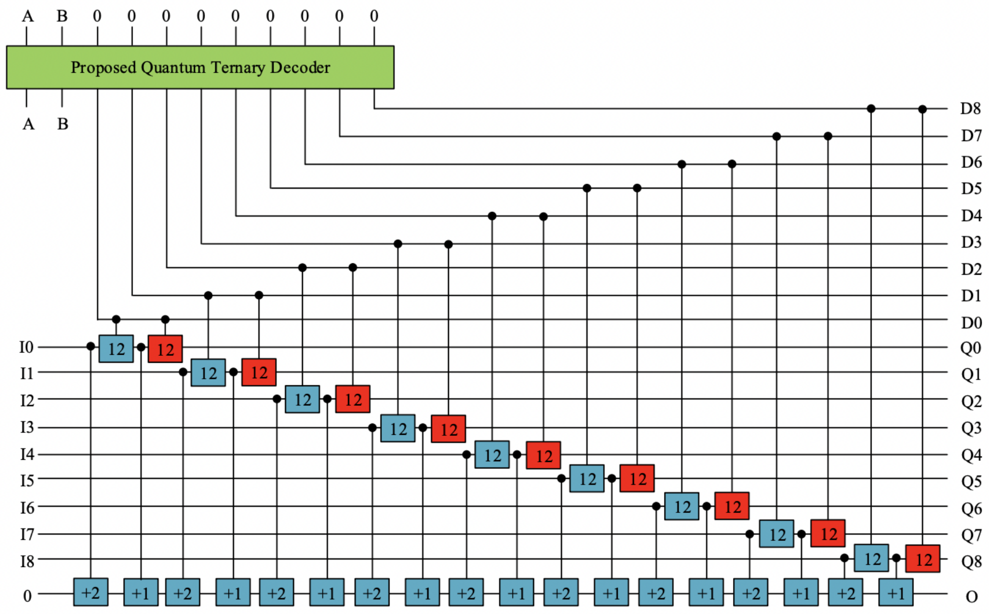


Figure 7.5: Realization of the proposed quantum ternary multiplexer circuit.

7.2.3 Ternary Reversible Demultiplexer

A demultiplexer circuit has one input, multiple outputs, and selectors. The selectors specify how the input is routed to the specified output. A ternary demultiplexer takes N ternary numbers as selectors and one ternary number as input, producing 3^N outputs. The block diagram of a 1×3^N ternary demultiplexer is shown in Fig. 7.6. Table 7.5 shows the truth table of a 1×9 ternary demultiplexer. The input is represented as I, the selectors are A and B, and the outputs are denoted as O. For a given selector combination, only one output will be equal to the input I, while the remaining outputs will be 0.

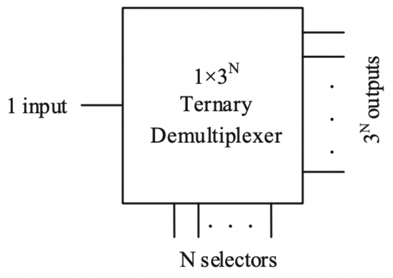


Figure 7.6: Block diagram of 1×3^N ternary demultiplexer circuit.

The proposed decoder and ternary-controlled Feynman gates are utilized to construct the ternary demultiplexer circuit. The realization of the proposed quantum ternary demultiplexer is shown in Fig. 7.7. In this circuit, input restoration is required, and the PQTDA, along with nine ternary controlled-Feynman gates, is employed. By using the red-colored Muthukrishnan–Stroud gates, input restoration can be achieved, resulting in a total quantum cost of 75.

Table 7.5: 1×9 ternary demultiplexer truth table.

Selectors	**Outputs**								
AB	O_0	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8
00	I	0	0	0	0	0	0	0	0
01	0	I	0	0	0	0	0	0	0
02	0	0	I	0	0	0	0	0	0
10	0	0	0	I	0	0	0	0	0
11	0	0	0	0	I	0	0	0	0
12	0	0	0	0	0	I	0	0	0
20	0	0	0	0	0	0	I	0	0
21	0	0	0	0	0	0	0	I	0
22	0	0	0	0	0	0	0	0	I

However, if input restoration is not necessary, the OQTDA design can be implemented. In this case, the red-colored gates can be removed, reducing the quantum cost to 53.

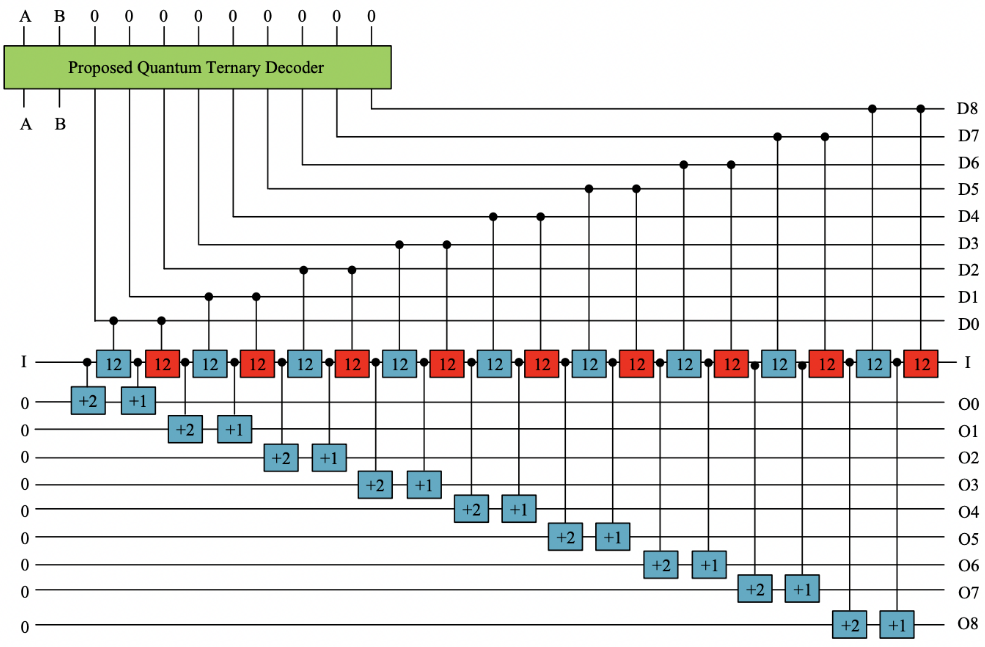


Figure 7.7: Block diagram of 1×3^N ternary demultiplexer circuit.

7.3 Evaluation Results

The proposed quantum ternary decoders, multiplexer, and demultiplexer are analyzed with respect to their quantum cost, depth, number of garbage outputs, and constant inputs. A thorough examination of these metrics provides a comprehensive evaluation of the efficiency and performance of the quantum circuits.

Table 7.6 compares our proposed quantum ternary decoder circuit with its counterpart in [78]. As shown, the proposed decoder (OQTDA) has lower quantum cost, depth, fewer garbage outputs, and constant inputs than the design in [78], eliminating 27 M-S gates and four permutation gates, and reducing both constant inputs and garbage outputs by one.

Table 7.6: Comparison between ternary reversible decoder designs.

	Proposed design	[78]	**Improvement percentage**
Quantum cost	26	57	54%
Depth	18	54	66%
Constant input number	9	10	10%
Garbage output number	2	3	33%
Input restoration capability	No	Yes	-

According to Table 7.7, the proposed multiplexer significantly improves quantum cost, depth, garbage outputs, and constant inputs when compared to the similar circuit suggested in [68]. Specifically, the improvements in quantum cost, depth, garbage outputs, and constant inputs are 48%, 41%, 4%, and 9%, respectively.

Table 7.7: Comparison between ternary reversible multiplexer designs.

	Proposed design	[68]	**Improvement percentage**
Quantum cost	53	102	48%
Depth	44	75	41%
Constant input number	10	11	9%
Garbage output number	20	21	4%
Input restoration capability	No	Yes	-

The comparison in Table 7.8 clearly demonstrates that the proposed design of the demultiplexer leads to substantial improvements in terms of quantum cost (48%), depth (41%), garbage outputs (7%), and constant inputs (5%) compared to its counterpart in [68].

Table 7.8: Comparison between ternary reversible demultiplexer designs.

	Proposed design	[68]	**Improvement percentage**
Quantum cost	53	102	48%
Depth	44	75	41%
Constant input number	18	19	5%
Garbage output number	12	13	7%
Input restoration capability	No	Yes	-

It should be noted that although all the proposed designs outperform the circuits suggested in [78, 68], the comparison has been made using the best-proposed designs, which do not restore inputs.

7.4 Conclusion

In this chapter, we present two advanced realizations of quantum ternary decoders, utilizing 1-qutrit permutation gates and 2-qutrit Muthukrishnan–Stroud gates. These novel decoder designs have been leveraged to develop associated quantum ternary multiplexers and demultiplexers. Our proposed circuits achieve significant improvements in several key performance metrics, including reductions in quantum cost, circuit depth, garbage outputs, and constant inputs. These advancements result in more efficient quantum ternary circuit designs compared to existing approaches. The enhanced efficiency of our designs not only advances the state of quantum ternary circuits but also opens new possibilities for optimizing the construction of various components in ternary quantum computers and other complex computational systems. The implications of these improvements extend to a broader range of applications, potentially fostering more effective and practical implementations of quantum technology.

Chapter 8

Quantum Reversible Ternary Circuit Design for Quantum Image

This chapter examines the gaps in existing binary quantum image processing circuits related to the development of a quantum ternary image processing circuit. The study proposes a novel design that integrates ternary Shift gates and ternary Muthukrishnan–Stroud gates, building on the qutrit representation of quantum images. This approach offers significant improvements in quantum cost, reduces the number of constant inputs, and lowers garbage outputs, which are critical for optimizing quantum circuit design.

8.1 Introduction

Quantum computing utilizes the unique principles of quantum mechanics, including superposition and entanglement, to process information in ways that significantly surpass the capabilities of classical systems. In recent times, significant advances have been made in both the theoretical and experimental dimensions of the research domain. Groundbreaking algorithms, such as Shor's integer factoring algorithm [131] and Grover's search algorithm [57], demonstrate the potential of quantum computing to revolutionize different domains of computing.

Digital image processing is one of the many fields poised to benefit from quantum computing. With image data sets expanding in size, conventional image processing algorithms face increasing difficulties in terms of time and storage efficiency, prompting the need for more advanced techniques [53]. Quantum image processing, a developing domain that merges quantum computing with digital imaging, aims to overcome these challenges by exploiting the computational power of quantum systems.

A key aspect of quantum image processing is the efficient representation of images. Although various quantum image representation models have been proposed [36, 40, 56, 62, 84, 124, 132, 141, 140, 149], many depend on binary logic, which often results in considerable storage overhead due to the large volume of image data. This chapter addresses the gaps in binary-based quantum image representation by examining multi-valued quantum systems, with a focus on qutrit-based representations. Multi-valued quantum systems not only reduce the number of qubits needed but also improve both security and storage efficiency in image processing applications [8, 106, 86].

Building on these advantages, this study focuses on the qutrit representation of quantum images (QTRQ), which has the potential to expand computational capabilities and optimize resource utilization. In [42], a compression circuit for quantum images utilizing qutrits was presented. In this chapter, we build upon that work by developing a more efficient quantum ternary circuit that employs qutrit-based Muthukrishnan-Stroud gates [49] and Shift gates. Our proposed design seeks to reduce quantum cost, minimize garbage outputs, and lower the number of constant inputs—key metrics in optimizing quantum circuits.

This chapter presents two significant contributions: first, we introduce a novel quantum ternary circuit architecture for QTRQ that achieves significant reductions in quantum cost, constant inputs, and garbage outputs; second, we demonstrate the circuit's effectiveness in quantum image processing. Specifically, our design results in a 20.56% reduction in quantum cost, an 11% reduction in constant inputs, and a complete elimination (100%)of garbage outputs compared to the existing design presented in [42]. The outcomes presented in this chapter were published in the Quantum Information Processing journal [134].

This chapter is organized as follows: Section 8.2 discusses the qutrit representation of quantum images. Section 8.3 presents the proposed implementation of the ternary gate. Section 8.4 provides a detailed overview of the architecture for the proposed quantum ternary gate and circuit. Section 8.5 assesses the performance of the circuit and benchmarks it with previous works. Finally, Section 8.6 concludes the chapter by summarizing the findings and discussing their implications for quantum image processes.

8.2 Qutrit Representation of Quantum Image

This section provides a brief review of existing qutrit image representation to facilitate a clear comparison between our proposed circuit and the existing approaches.

8.2.1 Quantum Image Representation

[42] introduced a novel qutrit-based method for representing grayscale images, where pixel locations and values are encoded using pairs of entangled qutrits. This technique utilizes six qutrits to represent 256 shades of grey, leading to some energy levels being redundant. Such redundancy can be advantageous for detecting and correcting errors during data transmission. The expression representing a $3^n \times 3^n$ image is as follows:

$$|I\rangle = \frac{1}{3^n} \sum_{Y=0}^{3^n-1} \sum_{X=0}^{3^n-1} |f(X,Y)\rangle |YX\rangle = \frac{1}{3^n} \sum_{Y=0}^{3^n-1} \sum_{X=0}^{3^n-1} \bigotimes_{i=0}^{q-1} |C^i_{YX}\rangle |YX\rangle \tag{8.1}$$

where $C^0_{YX} C^1_{YX} \ldots C^{q-1}_{YX}$ and $|Y\rangle|X\rangle$ encode the grayscale information and the corresponding position in the image, respectively. To construct the quantum image model for a $3^n \times 3^n$ image with 3^q shades of gray, $q+2n$ qutrits are needed. For example, a 3×3 dimensional image is considered in Figure 8.1, and its corresponding quantum image state is represented in Equation 8.2.

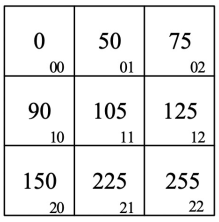


Figure 8.1: Example of a 3 × 3 dimensional image.

$$
\begin{aligned}
|I\rangle = \frac{1}{3}(&|0\rangle \otimes |00\rangle + |50\rangle \otimes |01\rangle + |75\rangle \otimes |02\rangle + |90\rangle \otimes |10\rangle + |105\rangle \otimes |11\rangle \\
&+ |125\rangle \otimes |12\rangle + |150\rangle \otimes |20\rangle + |225\rangle \otimes |21\rangle + |255\rangle \otimes |22\rangle) \\
= \frac{1}{3}(&|000000\rangle \otimes |00\rangle + |001212\rangle \otimes |01\rangle + |002210\rangle \otimes |02\rangle + |010100\rangle \otimes |10\rangle \\
&+ |010220\rangle \otimes |11\rangle \quad + |011122\rangle \otimes |12\rangle + |012120\rangle \otimes |20\rangle + |022100\rangle \otimes |21\rangle \\
&+ |100110\rangle \otimes |22\rangle)
\end{aligned}
\tag{8.2}
$$

8.2.2 Quantum Image Preparation

It is necessary to first store the image information in a quantum state before using quantum mechanics to process it. According to the preparation procedure for the novel qutrit representation of grayscale image [42], preparation of $q+2n$ qutrits and setting them all to 0 is the first step. The initial quantum state can be represented as follows:

$$|\Psi_0\rangle = |0\rangle^{q+2n} \tag{8.3}$$

The quantum circuit for the quantum image preparation process is depicted in Figure 8.2. In this figure, $|C_0\rangle$, $|C_1\rangle$, and $|C_{q-1}\rangle$ represent $|C^1_{YX}\rangle$, $|C^2_{YX}\rangle$, and $|C^{q-1}_{YX}\rangle$, respectively. There are two steps to the preparation; the position information must be prepared as a first step. This is achieved through the utilization of single-qutrit I and H gates. These gates are applied to transition the initial state $|\Psi_0\rangle$ into the intermediate state $|\Psi_1\rangle$, representing a superposition encompassing all the pixels of an empty image. This process is interpreted by Equation 8.4.

$$|\Psi_1\rangle \triangleq \frac{1}{3^n}\sum_{Y=0}^{3^n-1}\sum_{X=0}^{3^n-1}|0\rangle^{\otimes q}|YX\rangle \tag{8.4}$$

In the second step, the transition from the intermediate state $|\Psi_1\rangle$ to the final state $|\Psi_2\rangle$ results in the creation of the quantum representation of the QTRQ, representing the final quantum image. This process is explained by Equation 8.5.

$$|\Psi_2\rangle \triangleq \frac{1}{3^n}\sum_{Y=0}^{3^n-1}\sum_{X=0}^{3^n-1}|f(X,Y)\rangle|YX\rangle \tag{8.5}$$

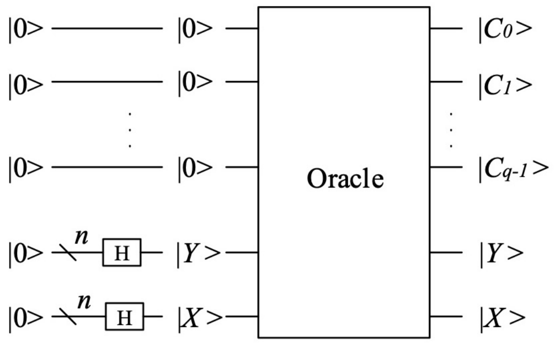


Figure 8.2: The quantum image preparation circuit.

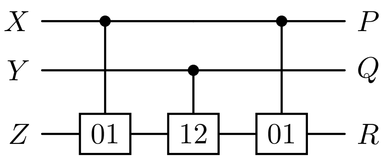


Figure 8.3: The realization of the Toffoli gate in a specific state.

8.3 Proposed Realization of Ternary Toffoli Gate in Specific State

Section 6.1, in Chapter 6 provides an explanation of the Ternary Toffoli gate. One of the most usable states for this gate occurs when the input Z is set to 0, and the transformation for that is equal to +2. Based on the values of control inputs, the quantum cost of the gate in this scenario can be equal to 5, 7, or 9. Here, we focus on the state when the transformation is equal to +2, a novel realization of the Ternary Toffoli gate in this situation is shown in Figure 8.3. In this realization, when the controlling inputs are equal to 0 and 1 no transformation will be applied on constant input 0. If both controlling inputs are equal to 2, $Z(01)$, $Z(12)$ and $Z(01)$ will be applied on constant input 0, respectively. But if only one of the controlling inputs equals 2, the target output restores the constant input, which is 0. Based on our suggested realization of the ternary Toffoli gate in the mentioned state, the quantum cost can be 3, 5, or 7, depending on the values of the inputs. If both inputs are 2, the quantum cost is 3. If one of them is 2, the quantum cost is 5. If neither of them is 2, the quantum cost is 7. In summary, the quantum cost can be expressed as $3 + (2 \times \text{number of controlling values that are not } 2)$.

8.4 Proposed Design for Qutrit Representation of Quantum Image

The truth table of the qutrit representation of the grayscale image (QTRQ) discussed above is given in table 8.1. In this table, X and Y are the input variables, while C_0, C_1, C_2, C_3, C_4, and C_5 represent the output variables. These outputs correspond to the following:

Table 8.1: Truth table of the novel qutrit representation of the grayscale image (QTRQ).

X	Y	C_0	C_1	C_2	C_3	C_4	C_5
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	2	0	1	2	1	2	0
1	0	0	0	1	2	1	2
1	1	0	1	0	2	2	0
1	2	0	2	2	1	0	0
2	0	0	0	2	2	1	0
2	1	0	1	1	1	2	2
2	2	1	0	0	1	1	0

$$C_0 = Z(+1)(Y^2X^2) \tag{8.6}$$

$$C_1 = Z(+1)(Y^1) + Z(+1)(Y^2X^0) + Z(+2)(Y^2X^1) \tag{8.7}$$

$$C_2 = Z(+1)(Y^0X^1)+Z(+1)(Y^1X^2)+Z(+2)(Y^0X^2)+Z(+1)(Y^2X^2)+Z(+2)(Y^2) \tag{8.8}$$

$$C_3 = Z(+2)(Y^1X^1)+Z(+1)(Y^1)+Z(+1)(Y^2)+Z(+1)(Y^0X^0)+Z(+2)(Y^0)+Z(+2)(Y^1X^1) \tag{8.9}$$

$$C_4 = Z(+2)(Y^0X^0)+Z(+1)(Y^0)+Z(+1)(Y^2X^2)+Z(+1)(Y^1X^0)+Z(+2)(Y^1)+Z(+2)(Y^2X^0) \tag{8.10}$$

$$C_5 = Z(+2)(Y^0X^1) + Z(+2)(Y^1X^2) \tag{8.11}$$

In order to design our proposed new quantum ternary circuit based on the qutrit representation of the grayscale image (QTRQ) in [42] and enhance understanding, we initially present some operations of the circuit aligned with equations 21 to 26 .Subsequently, the complete design is illustrated in Figure 8.4g. The realization of the first part (C_0) is depicted in Figure 8.4a, including three 2-qutrit Muthukrishnan–Stroud gates. In this part, if inputs X and Y both are equal to 22, only output C_0 equals 1, which aligns with Equation 8.6. It requires 1 constant input, which is 0.

The realization of the second part (C_1) is depicted in Figure 8.4b, employing six 1-qutrit shift gates and seven 2-qutrit Muthukrishnan–Stroud gates. In this part, when inputs X and Y correspond to 02 or 12, the output C_1 equals 1 or 2, respectively. Moreover, if input Y is 1, output C_1 equals 1, consistent with Equation 8.7. It requires one constant input, denoted as 0. In this part two Z(+1) gates can be merged into a Z(+2) gate, resulting in a quantum cost of 12.

The third part (C_2) is illustrated in Figure 8.4c. As can be seen, we employed thirteen Muthukrishnan–Stroud gates and eight shift gates, resulting in a quantum cost of 21. However, by merging two Z(+1) gates into a Z(+2) gate and similarly merging two Z(+2) gates into a Z(+1) gate , we can reduce the cost to 19.

The fourth part (C_3) is shown in Figure 8.4d. As can be observed, we utilized a total of 28 gates, including twelve Muthukrishnan–Stroud gates and sixteen shift gates. It should be noted that in this realization, each pair of double gates within the blue boxes can be merged into one shift gate, and the gates within the red boxes can be omitted. Consequently, the overall quantum cost is reduced to 21.

Figure 8.4e shows the realization of the fifth part (C_4). We used fourteen Muthukrishnan–Stroud gates and fourteen shift gates to construct the circuit for Equation 8.10. The cost of this part is 28, but the gates within the red boxes can be eliminated altogether. Thus, the total quantum cost is decreased to 22.

The realization of the last part of the circuit (C_5), which aligns with Equation 8.11, is shown in Figure 8.4f. As can be observed, we used six 1-qutrit shift gates and six 2-qutrit Muthukrishnan–Stroud gates. In this part, when inputs X and Y correspond to 10 or 21, the output C_5 equals 2. The gates enclosed in the blue boxes can be merged, resulting in a total quantum cost of 11.

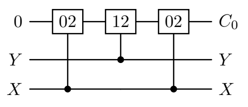
The aforementioned parts were integrated to create the complete design of the proposed quantum ternary circuit for the qutrit representation of grayscale images (QTRQ), as illustrated in Figure 8.4g. This integration reduced the overall quantum cost from 88 to 83 by consolidating and eliminating certain gates, as highlighted in the orange and green boxes, respectively. The first step in the quantum image preparation involves six ternary Identity gates and two ternary Hadamard gates, which transform the initial state $|\Psi_0\rangle$ into the intermediate state $|\Psi_1\rangle$. This preparation step establishes the foundation for the next step.

The core of the circuit design is found in the second step, where the quantum image representation is refined to its final state, $|\Psi_1\rangle$. In this step, 28 quantum ternary Shift gates and 55 quantum ternary M-S gates are employed to transform the intermediate state $|\Psi_1\rangle$ into the final state $|\Psi_2\rangle$. This two-step conversion process ensures a gradual and precise development of the final quantum image representation. Each output state corresponds to specific elements of the final quantum image.

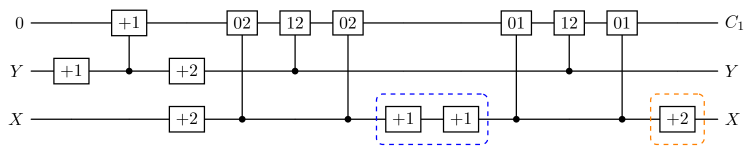
The circuit produces six distinct quantum states, denoted by $|C_0\rangle$ through $|C_5\rangle$. These quantum output states are represented as $|C^0_{YX}\rangle$, $|C^1_{YX}\rangle$, $|C^2_{YX}\rangle$, $|C^3_{YX}\rangle$, $|C^4_{YX}\rangle$, and $|C^5_{YX}\rangle$. These states collectively form a complete qutrit-based quantum representation of the grayscale image. This multiplicity of outputs provides a thorough encoding of the image information.

The quantum cost of the quantum image preparation procedure is determined by the number of gates used in the circuit. The quantum cost for Shift and M-S gates is 1 each. Therefore, the second stage of the circuit, which employs 28 Shift gates and 55 M-S gates, has a quantum cost of 83. Including the two Hadamard gates from the first stage, the total quantum cost of the circuit amounts to 85.

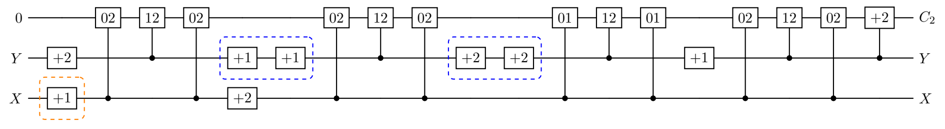
In addition, this design requires 8 constant inputs and generates 0 garbage outputs. This optimization highlights the efficiency and effectiveness of the quantum circuit design.



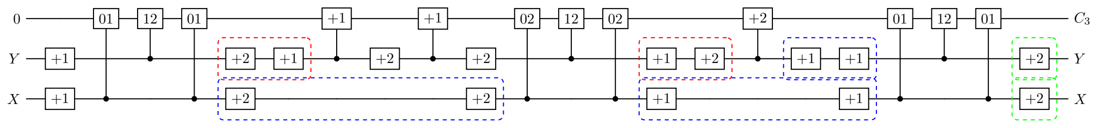
(a) The realization of C_0.



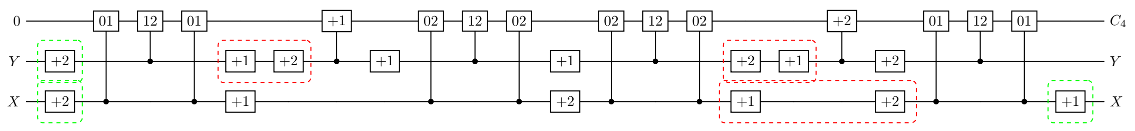
(b) The realization of C_1.



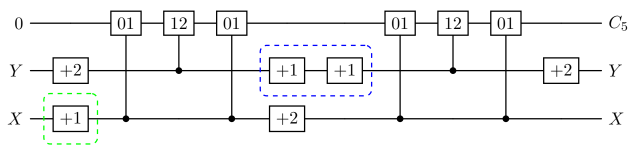
(c) The realization of C_2.



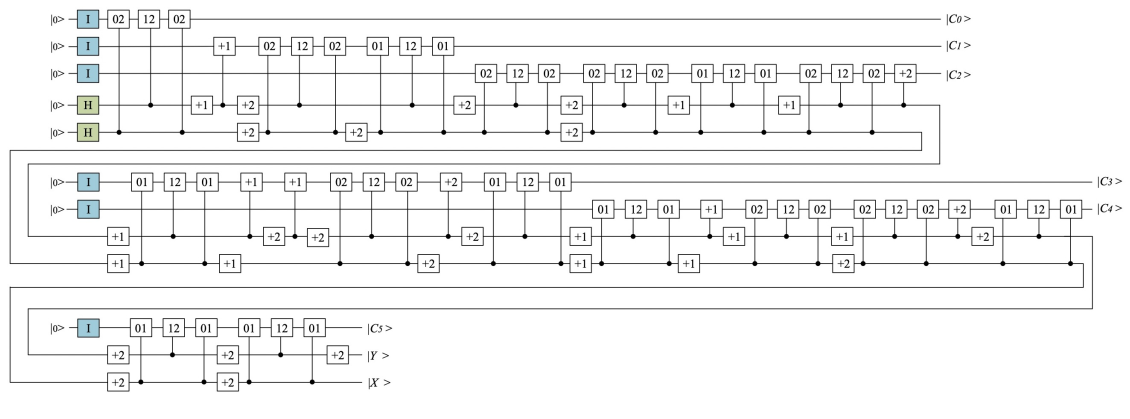
(d) The realization of C_3.



(e) The realization of C_4.



(f) The realization of C_5.



(g) The optimized realization.

Figure 8.4: The realization of the proposed quantum ternary circuit for QTRQ.

8.5 Evaluation Results

In current literature, Novel-enhanced Quantum Representation of Images (NEQR) is frequently used to encode grayscale pixel values for binary and grayscale images [149]. Inspired by NEQR, a qutrit representation for ternary and grayscale images has been introduced, offering a greater capacity for information storage compared to NEQR [42]. To the best of our knowledge, this is the only qutrit approach specifically focused on grayscale images. Our objective is to minimize key parameters in designing a quantum ternary circuit for the qutrit representation of quantum images. The evaluation of these quantum circuits includes an analysis of their quantum cost, the number of constant inputs, and garbage outputs. These metrics are crucial for assessing circuit efficiency and making significant advancements in quantum image processing. Table 8.2 compares our newly proposed quantum ternary circuit for the novel qutrit representation of grayscale images with the design presented in reference [42]. It is clear that our design demonstrates lower quantum cost, fewer garbage outputs, and reduced constant inputs compared to the design from [42]. Specifically, our design eliminates 22 Muthukrishnan and Stroud gates and reduces constant inputs and garbage outputs by one each, resulting in 8 constant inputs and 0 garbage outputs. This improvement translates to a 20.56% reduction in quantum cost, an 11% reduction in constant inputs, and a 100% reduction in garbage outputs.

Table 8.2: Evaluation of quantum ternary circuits for QTRQ

	Quantum Cost	Constant Input	Garbage Output
Design in [42]	107	9	1
Our Proposed Design	85	8	0
Improvement Percentage	20.56%	11%	100%

8.6 Conclusion

Integrating quantum mechanics into conventional image processing offers a promising strategy to address the demanding real-time computational requirements. However, in the current era of Noisy Intermediate-Scale Quantum (NISQ) devices, optimizing quantum gate counts and circuit depths is crucial to mitigate noise effects. Efficient quantum image storage is also vital, which can be achieved by employing multi-valued quantum systems in quantum image processing. These systems enhance information encryption security while requiring fewer qubits and less storage space. Our research aims to develop a novel quantum ternary circuit for the novel qutrit representation of grayscale images. The results demonstrate that the proposed design outperformed existing approaches in terms of quantum cost, the number of constant inputs, and garbage outputs, which are essential parameters in quantum circuit design. Minimizing these factors can significantly advance quantum image processing. Quantum image processing is still evolving, and research on ternary quantum image models is limited. Although the current qutrit representation circuit is optimized and compressed, it may not be the most effective. Future research could benefit from investigating more effective compression and optimization strategies.

Chapter 9

Balanced Ternary Reversible Comparator for Qutrit Quantum Circuits

This chapter is dedicated to two key objectives: first, the synthesis of quantum reversible 1-qutrit comparator circuit through the utilization of balanced ternary logic, and second, the design of generalized reversible n-qutrit comparator circuit. The suggested balanced ternary 1-qutrit comparator has demonstrated notable improvements in terms of quantum cost (65%), the number of constant inputs (50%) and garbage outputs (33%) compared to existing unbalanced comparator designs.

9.1 Introduction

Ternary arithmetic functions can be represented in two primary forms: balanced and unbalanced. In the unbalanced system, a qutrit is represented by the values 0, 1, or 2. On the other hand, balanced ternary representation employs the symbols $\hat{1}$, 0, and 1, where $\hat{1}$ indicates a negative one. Balanced ternary is generally more efficient for implementing arithmetic functions compared to its unbalanced counterpart. This is because balanced ternary simplifies several operations, including ternary inversion, and removes the need for an additional sign digit. It also ensures that operations such as rounding to the nearest integer and truncation are straightforward, facilitates the generation of partial products, and lowers the likelihood of carry generation. Furthermore, it features shorter carry ripple paths, making it advantageous for designing efficient arithmetic circuits [46].

Quantum ternary comparator circuits play a crucial role in quantum computing. These circuits are primarily used to support quantum algorithms and operations that involve ternary logic or quantum data encoded in ternary states. This includes applications such as ternary arithmetic, error correction, search algorithms, machine learning, and state discrimination. To date, all existing quantum ternary reversible comparator circuits have been developed using unbalanced ternary representation [41, 69, 103, 147]. In this chapter, we introduce a novel approach by employing balanced ternary representation for constructing quantum ternary comparator circuits. Additionally, we evaluate our proposed circuits based on several key metrics, including quantum cost, the number of garbage outputs, and the number of constant inputs.

The chapter is structured as follows: Section 9.2 presents the realization of balanced ternary comparator circuits, detailing the proposed approaches. Section 9.3 discusses the evaluation of the proposed circuits, comparing them with existing solutions and analyzing performance metrics. Section 9.4 concludes the chapter with a summary of the findings.

9.2 Proposed Design of Balanced Ternary Comparator

In this section, we present a *1-qutrit comparator* based on balanced ternary reversible logic. We use this comparator design to construct *n-qutrit balanced ternary reversible comparator*. The proposed designs in this study utilizes balanced Shift gates and Muthukrishnan and Stroud gate.

9.2.1 Balanced Ternary 1-qutrit Comparator Circuit

Ternary comparators play a critical role in balanced ternary reversible networks and are an integral part of complex devices. Table 9.1 presents the truth table of a *1-qutrit comparator* circuit. This table shows that A and B are the input variables, and F represents the output variable. When the inputs A and B is equal, the output F is 0. When A is greater than B, the output F is 1. Conversely, if A is less than B, the output F is equal to $\hat{1}$. Generally, the function F is defined in Equation 9.1:

$$F = \begin{cases} \hat{1} & \text{if } A < B \\ 0 & \text{if } A = B \\ 1 & \text{if } A > B \end{cases} \tag{9.1}$$

Table 9.1: Truth table of a *1-qutrit comparator* circuit.

A	B	F
$\hat{1}$	$\hat{1}$	0
$\hat{1}$	0	$\hat{1}$
$\hat{1}$	1	$\hat{1}$
0	$\hat{1}$	1
0	0	0
0	1	$\hat{1}$
1	$\hat{1}$	1
1	0	1
1	1	0

In order to construct balanced ternary *1-qutrit comparator*, we used ternary balanced Self Shift gate and ternary balanced Muthukrishnan and Stroud gate. The proposed balanced ternary comparator can be observed in Figure 9.1. This circuit takes two single qutrit numbers as inputs, and assigns a value to the output based on relationship between the inputs value. If the first input is equal to the second one, the output is set to 0. If the first input is greater than the second one, the output is set to 1. If the first input is less than the second input, the output is equal to $\hat{1}$. As can be seen in Figure 9.1, in our realization, there are three inputs A, B and 0. P,Q and F represent the output of the circuit. The output F,

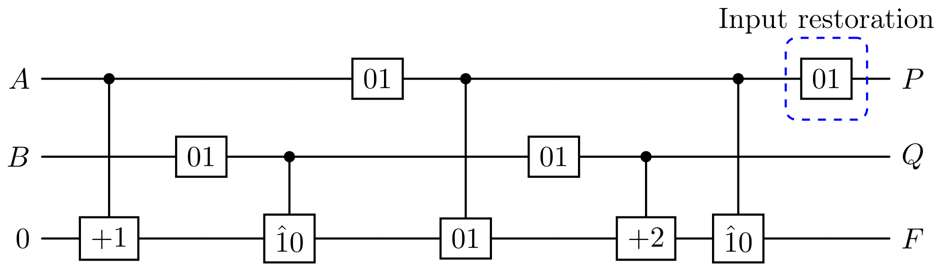


Figure 9.1: The proposed balanced ternary reversible *1-qutrit comparator* circuit.

which is the target output, represent the result of the inputs A and B comparison. This realization requires one *constant input*, which is 0, and produces two *garbage outputs*, which are P and Q. To design this circuit, we used 5 Muthukrishnan and Stroud gates and 4 shift gates, resulting in a *quantum cost* of 9 for the proposed realization. However, if input restoration is not needed, the last Shift gate can be removed, and the quantum cost decreases to 8.

9.2.2 Balanced Ternary N-Qutrit Comparator Circuit

In a balanced ternary n-qutrit comparator, two n-qutrit numbers are being compared. At the beginning, each qutrit of these numbers is compared using n separated 1-qutrit comparators simultaneously. In the next step, the results from these 1-qutrit comparators are compared, beginning with the least significant qutrits. In the second step, we use n-1 *sub comparator circuit* for comparison of the results from the first step.

The proposed realization of the sub-comparator circuit is illustrated in Figure 9.2. This circuit has three inputs: F_0, F_1, and $\hat{1}$, where F_0 and F_1 are the outputs from the previous step, and $\hat{1}$ is a constant input. The outputs of the circuit are labeled R_0, R_1, and F. Here, R_0 and R_1 correspond to F_0 and F_1, respectively. The output F, which represents the result of the comparison between F_0 and F_1, is the target output of the circuit. This realization requires one constant input, $\hat{1}$, and produces two garbage outputs, R_0 and R_1. The circuit consists of 6 Shift gates and 5 Muthukrishnan-Stroud gates, resulting in a total quantum cost of 11. However, if input restoration is not necessary, the Shift and Muthukrishnan-Stroud gates in the box can be eliminated, reducing the quantum cost to 8. Table 9.2 presents the truth table for the sub-comparator circuit.

Table 9.2: Truth table of the sub comparator circuit.

F_0	F_1	F
$\hat{1}$	$\hat{1}$	$\hat{1}$
$\hat{1}$	0	$\hat{1}$
$\hat{1}$	1	1
0	$\hat{1}$	$\hat{1}$
0	0	0
0	1	1
1	$\hat{1}$	$\hat{1}$
1	0	1
1	1	1

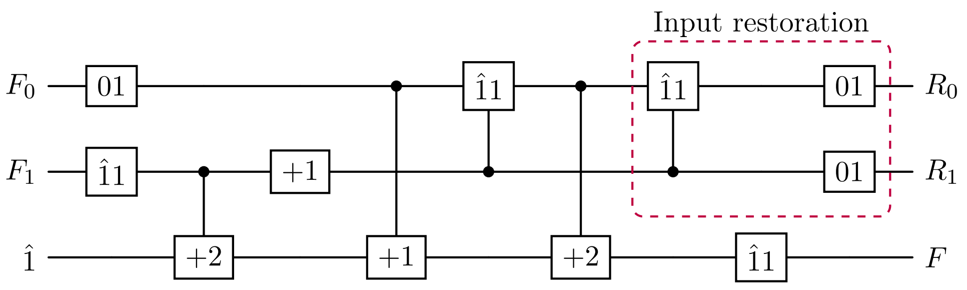


Figure 9.2: The proposed balanced ternary reversible *sub comparator* circuit.

As can be seen in this table, when F_0 and F_1 are equal to 0 (indicating that the first number is equal to the second), the output is set to 0. When F_1 is 1, or when F_0 and F_1 are equal to 1 and 0, respectively (indicating that the first number is greater than the second), the output is set to 1. When F_1 is $\hat{1}$, or when F_0 and F_1 are equal to $\hat{1}$ and 0, respectively (indicating that the first number is less than the second), the output is equal to $\hat{1}$.

The realization of our proposed *balanced ternary n-qutrit comparator* circuit is shown in Figure 9.3. To construct this circuit, we used N 1-qutrit comparator circuits and N-1 sub comparator circuits, which denoted by *1-qutrit Comp* and *Sub Comp*, respectively. The quantum cost, the number of constant inputs, and garbage outputs can be determined according to Equations 9.2, 9.3 and 9.4, respectively.

$$\text{Quantum Cost } = 8N + 8(N-1) = 16N - 8 \tag{9.2}$$

$$\text{The number of constant inputs } = N + N - 1 = 2N - 1 \tag{9.3}$$

$$\text{The number of garbage outputs } = 2N + 2(N-1) = 4N - 2 \tag{9.4}$$

As an example, consider two balanced ternary 2-qutrit numbers, A and B. For the comparison between A and B, in the first step, A_0 and A_1 should be compared using our proposed 1-qutrit comparator with B_0 and B_1, respectively. In the next step the results of 1-qutrit comparator circuits should be compared using the proposed sub comparator circuit. The final result is in accordance with Table 9.2, Equations 9.5, 9.6 and 9.7:

$$A_1A_0 = B_1B_0 \text{ if } A_1 = B_1 \text{ and } A_0 = B_0 \tag{9.5}$$

$$A_1A_0 > B_1B_0 \text{ if } A_1 > B_1 \text{ or } A_1 = B_1 \text{ and } A_0 > B_0 \tag{9.6}$$

$$A_1A_0 < B_1B_0 \text{ if } A_1 < B_1 \text{ or } A_1 = B_1 \text{ and } A_0 < B_0 \tag{9.7}$$

According to Equation 9.5, the final result is set to 0 when both results in the first step are equal to 0. Equation 9.6 shows that the final result is set to 1, if the result of second comparator is 1 or the result of the first and the second are 1 and 0, respectively. Equation 9.7 also means that the final output is set to $\hat{1}$, if the

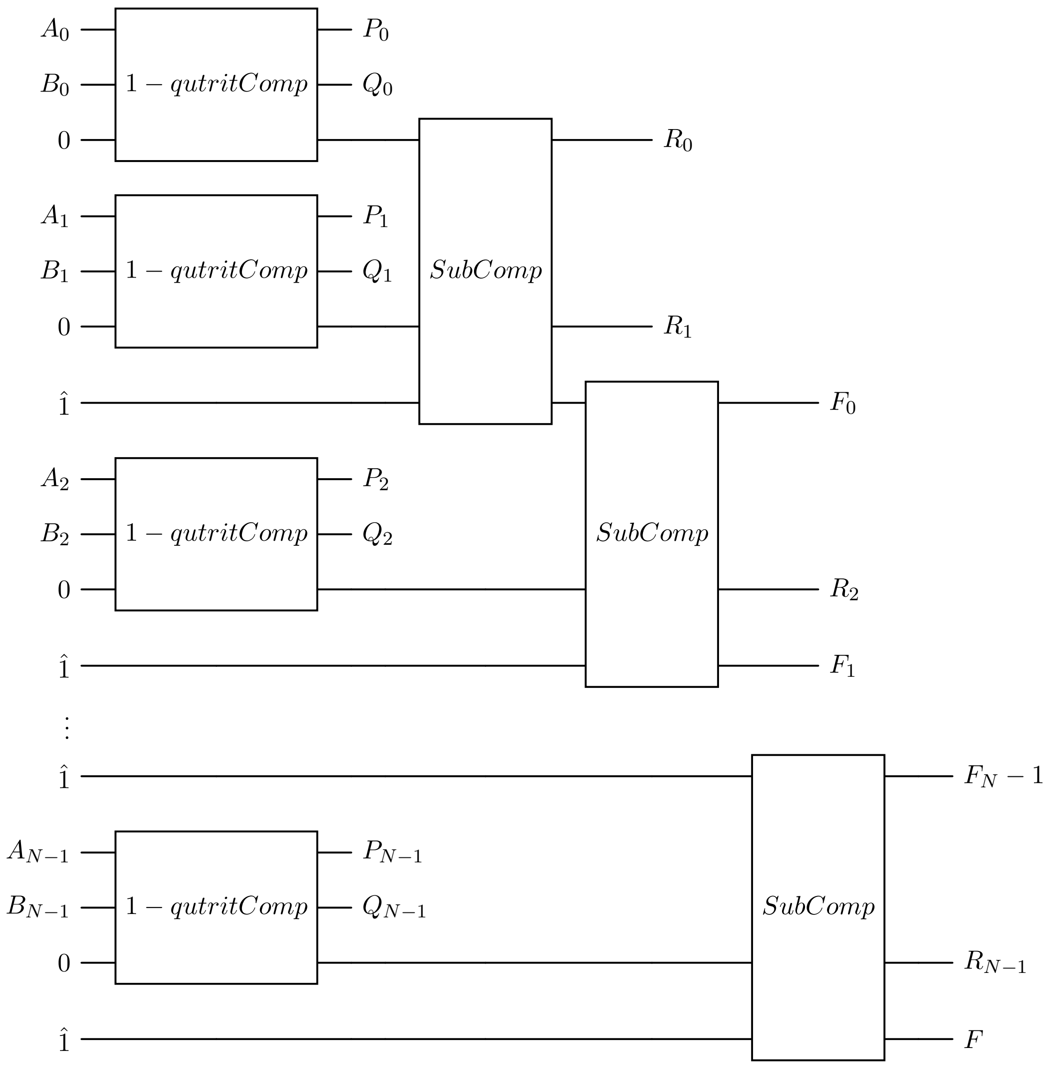


Figure 9.3: The proposed balanced ternary reversible *n-qutrit comparator* circuit.

result of second comparator is $\hat{1}$ or the result of the first and the second are $\hat{1}$ and 0, respectively. The realization of balanced ternary *2-qutrit comparator* circuit is illustrated in Figure 9.4. As you can seen the inputs of the first comparator circuit are A_0 and B_0, and for the second comparator the inputs are shown by A_1 and B_1. The target outputs of these circuits are the inputs of the sub comparator circuit. This comparison requires 3 constant inputs and produces 6 garbage outputs which align with the Equations 9.3 and 9.4 respectively. The quantum cost of this circuit is 24 which is in accordance with Equation 9.2.

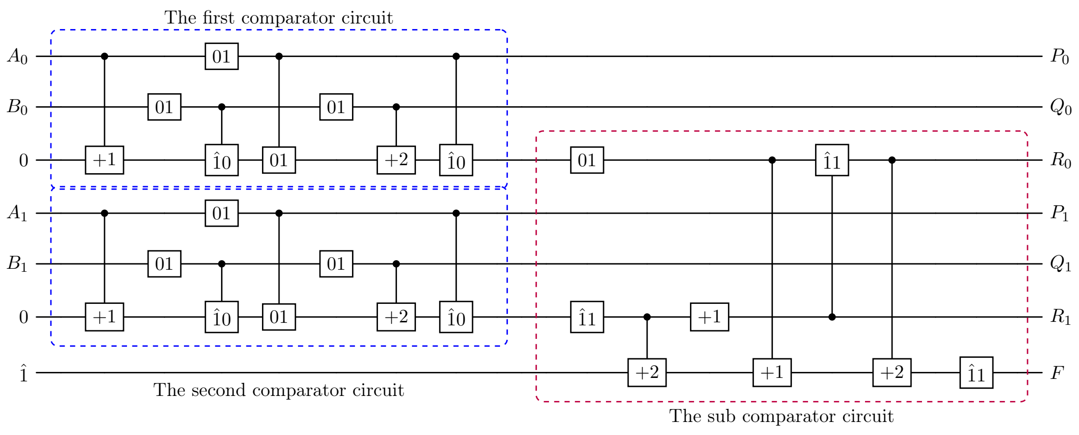

The first comparator circuit
A_0
B_0
0
A_1
B_1
0
$\hat{1}$
The second comparator circuit
The sub comparator circuit
P_0
Q_0
R_0
P_1
Q_1
R_1
F

Figure 9.4: The proposed balanced ternary reversible *2-qutrit comparator* circuit.

9.3 Evaluation Results

In this work, we introduced the use of balanced representation logic for ternary comparator circuits for the first time, leading to a substantial improvement in overall performance. This section involves two separate assessments where we compare our proposed comparator circuits with the existing unbalanced ternary counterparts to assess their relative performance and efficiency. Firstly, we examine the quantum cost of the proposed 1-qutrit comparator alongside the number of constant inputs and garbage outputs. These factors are thoroughly examined, analyzed and compared with its unbalanced counterparts [103, 147]. Secondly, the proposed n-qutrit comparator is evaluated objectively and comprehensively with other existing ones [41, 69, 103, 147]. It worth noting that in first part of assessments, we determine the improvement rate concerning the best results in the literature.

9.3.1 Evaluation of Ternary Reversible 1-Qutrit Comparator Circuits

In Table 9.3, we provide a comparative analysis for ternary reversible 1-qutrit comparator with respect to quantum cost, the number of constant inputs, the number of garbage outputs. Notably, this comparison demonstrates that the quantum cost of the our proposed design is 65% lower than the unbalanced version [103]. The proposed circuit has decreased the number of garbage outputs from 3 to 2, representing a 33% reduction. The table also shows improved efficiency in terms of the number of constant inputs which reduced from 2 to 1, resulting to 50% improvement. It can be concluded that, our design is more efficient than its unbalanced counterparts in the literature [103, 147].

Table 9.3: Comparison between proposed 1-qutrit comparator and unbalanced designs.

	Proposed comparator	**Improvement percentage**	[103]	[147]
Quantum cost	8	65%	23	55
Constant input number	1	50%	2	6
Garbage output number	2	33%	3	7

9.3.2 Evaluation of Ternary Reversible N-Qutrit Comparator Circuits

Table 9.4 provides a comprehensive comparison between our proposed balanced ternary n-qutrit comparator circuit and existing alternatives [41, 69, 103, 147]. This comparison clearly illustrates that the proposed comparator, in balanced representation, surpasses others in [41, 69, 103, 147]. Mainly because it has less quantum cost and fewer constant input number and garbage output number used in the design, leading to considerable improvements. It should be noted that, the number of garbage output in our design is $4N - 2$ and in [69, 103, 147] it is unreported. However, the undeniable fact is that, due to the considerably higher number of constant inputs in comparison to our design which has $2N - 1$ constant inputs, the number of garbage output is significantly greater. It is also important to note that, since the lower quantum cost leads to better efficiency in quantum circuit design, despite having the same number of constant inputs as the proposed design in [41] with the value of $2N - 1$, the lower quantum cost in our design which is $16N - 8$ allows us to assert its greater efficiency.

Table 9.4: Comparison between proposed n-qutrit comparator and unbalanced designs.

	Proposed comparator	[41]	[69]	[103]	[147]
Quantum cost	$16N - 8$	$18N - 8$	$236N - 169$	$43N - 20$	$56N + 6$
Constant input number	$2N - 1$	$2N - 1$	$13N - 7$	$4N - 2$	$5N + 3$
Garbage output number	$4N - 2$	Not reported	Not reported	Not reported	Not reported

9.4 Conclusion

In this chapter, we have presented a balanced ternary 1-qutrit comparator circuit utilizing Shift gates and Muthukrishnan and Stroud gate. Scalability is of paramount importance in circuit design and is essential when an arbitrary number of inputs is required. In order to achieve scalability, we have introduced the balanced ternary reversible sub-comparator circuit. Then, the realization of a balanced ternary n-qutrit comparator has been proposed. While the realization of balanced ternary comparator circuits is represented for the first time, we have made efforts to introduce optimal designs, and the final assessments for our proposed balanced ternary comparator circuits compared to the unbalanced version showed considerable improvements in terms of quantum cost, the number of constant inputs and the number of garbage outputs.

Chapter 10

Quaternary Reversible Circuit Optimization for Scalable Multiplexer and Demultiplexer

This chapter introduces a new approach for designing quaternary reversible multiplexer and demultiplexer circuits, which are critical components in ALUs and significantly influence processor performance. We propose scalable realizations of quaternary reversible 4×1 multiplexers and 1×4 demultiplexers using specialized quantum gates, and further extend these designs to generalized n×1 and 1×n configurations. Comparative analysis with existing designs shows that our circuits are more efficient in terms of quantum cost, the number of garbage outputs, and constant inputs, offering a promising solution for improving system performance.

10.1 Introduction

Recently, multiple-valued logic (MVL) has garnered significant attention as binary logic faces severe limitations due to thermal and reliability challenges [123]. Reversible multiple-valued logic (RMVL) is particularly well-suited for quantum applications due to its security advantages in quantum cryptography and its effectiveness in quantum information processing [10, 133, 30]. As mentioned earlier, ternary logic, one form of MVL, has shown promise, but it struggles to represent conventional binary logic functions [93, 6, 7, 5, 114, 52, 61, 64, 90, 105, 35]. By contrast, quaternary logic offers a more versatile alternative. In quaternary logic, two binary bits can be grouped into quaternary values to better represent binary logic functions [77]. In quantum quaternary systems, the basic unit of information is a qudit, which can exist in four possible states: $|0\rangle$, $|1\rangle$, $|2\rangle$, and $|3\rangle$.

Many essential circuits have been presented based on quaternary reversible logic, such as comparators, parallel adders, full adders, half adders, subtractors, and decoders [73, 136, 60, 71, 44, 43, 59, 118, 107]. This chapter focuses on the design and synthesis of quaternary reversible multiplexer and demultiplexer circuits, which are essential components for a wide range of systems, including computers, communication devices, and arithmetic logic units [59]. The goal of these designs is to enhance efficiency by reducing key metrics such as quantum cost, garbage outputs, and constant inputs, thus offering improvements over existing designs in [59, 74, 72]. The results presented in this chapter were published in the

IEEE Access journal [102].

This chapter is organized as follows: Section 10.2 presents the design and implementation of the proposed quaternary reversible multiplexer and demultiplexer circuits, detailing the circuit architecture and logic gates employed. Section 10.3 evaluates the performance of the proposed circuits, focusing on quantum cost, garbage outputs, and constant inputs, and compares these results with those of existing circuits. Finally, Section 10.4 concludes the chapter by summarizing the key findings and suggesting directions for future research.

10.2 Proposed Quaternary Reversible Circuits

In this section, we propose a scalable quaternary reversible 4×1 multiplexer, and we use it to design the quaternary reversible 16×1 and $n \times 1$ multiplexers. Moreover, we introduce the new scalable quaternary reversible 1×4 demultiplexer to design 1×16 and $1 \times n$ demultiplexers. We use quaternary 1-qudit Shift and 3-qudit Controlled Feynman gates. The aim is to reduce the overall quantum cost, the number of constant inputs, and the number of garbage outputs.

10.2.1 Proposed Quaternary Reversible Multiplexer Circuit

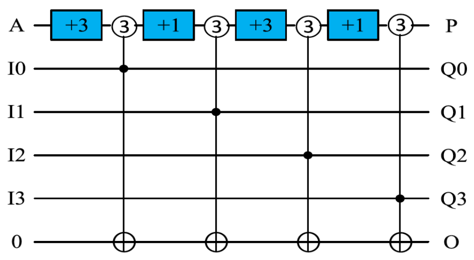
Before discussing our proposed quaternary reversible multiplexer circuit, we provide the basic definitions and properties of the quaternary multiplexer. A quaternary multiplexer with 4^m inputs has m select lines to select which input should be sent to the output. Let A be a selector equal to 0, 1, 2, or 3. In a 4×1 multiplexer, when A is equal to 0, 1, 2, or 3, the output equals I_0, I_1, I_2, or I_3, respectively. Table 10.1 shows the truth table of the quaternary 4×1 multiplexer.

Table 10.1: The truth table of quaternary 4×1 multiplexer.

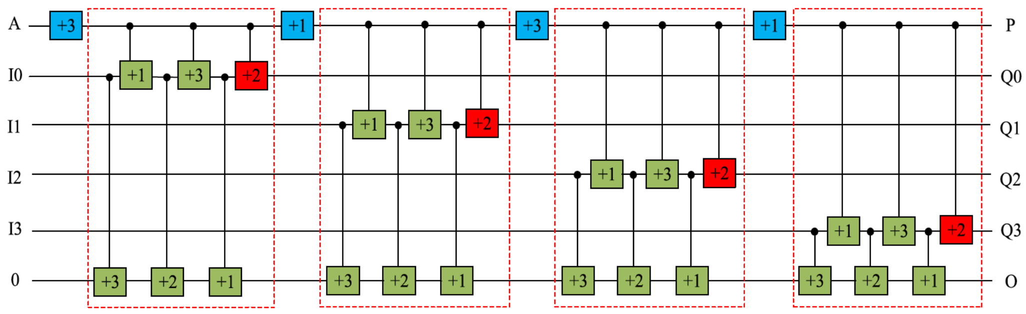
Selector	**Output**
A	**O**
0	I_0
1	I_1
2	I_2
3	I_3

The realization of our proposed quaternary reversible 4×1 multiplexer circuit is illustrated in Figure 10.1. As shown in the figure, we used four quaternary 1-qudit Shift gates and four quaternary 3-qudit Controlled Feynman gates. In this realization, the main inputs are I_0 to I_3, and one constant input of 0 is required. The selector is A, and the main output is O. The circuit produces five garbage outputs that are Q_0 to Q_3 and P. The output P is equal to the selector A, and the outputs Q_0 to Q_3 are equal to the inputs I_0 to I_3, respectively. In this circuit, when the selector A is equal to 0, the controlling value of the first Controlled Feynman gate is 3, and the output O is equal to I_0. If A is equal to 1, the second Controlled Feynman gate is 3, and the output O is I_1. Moreover, when the selector is equal to 2 and 3, the output O is equal to I_2 and I_3, respectively.

The realization of this circuit using quaternary Shift and Muthukrishnan-Stroud gates is shown in Figure 10.1b. In this figure, red boxes depict quaternary



(a) Symbol.



(b) The realization using M-S and Shift gates.

Figure 10.1: The proposed quaternary reversible 4×1 multiplexer circuit.

Controlled Feynman gates. Generally, four quaternary Shift gates and twenty-four quaternary Muthukrishnan-Stroud gates were used. Therefore, the quantum cost of the proposed quaternary reversible 4×1 multiplexer circuit is 28. It is worth mentioning that, in a multiplexer circuit, it is not necessary to restore the input I at the output Q. So, we can remove the red Muthukrishnan-Stroud gates in this realization. The quantum cost can be decreased by 24. In both suggested ways, the number of constant inputs is 1, and the number of garbage outputs is 5.

Our proposed quaternary reversible 4×1 multiplexer can be used to construct a 16×1 multiplexer. For designing this multiplexer, 16 inputs, two selectors, and one output are necessary. The truth table of this circuit is shown in Table 10.2. Only the selected input is gated to the output O for a given selector combination of A and B.

Figure 10.2a shows the logical architecture of the proposed quaternary 16×1 multiplexer using 4×1 multiplexers. As shown, five 4×1 quaternary multiplexers are required. In this design, the first inputs of the first-row multiplexers are activated when input B is equal to 0. Activation of the second inputs of multiplexers occurs when input B is equal to 1. If B is equal to 2 and 3, the third and fourth inputs of multiplexers are activated, respectively. Moreover, the output of the first multiplexer is gated on the main output O when the selector A is equal to 0. If A is equal to 1, the main input is sent to the main output by the second multiplexer. When A is equal to 2 and 3, the output of the third and the fourth multiplexers are gated on the output O, respectively.

Figure 10.2b illustrates the realization of the proposed quaternary reversible 16×1 multiplexer using a 4×1 multiplexer. The red boxes indicate our proposed quaternary reversible 4×1 multiplexer. In this circuit, there are five constant inputs, which are 0, and sixteen main inputs, which are shown by I_0 to I_{15}. The

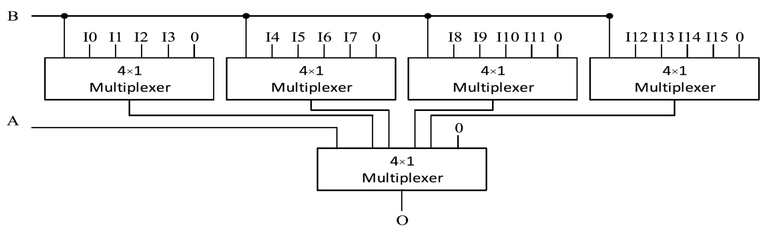
Table 10.2: The truth table of a quaternary 16×1 multiplexer.

Selectors	**Output**
AB	**O**
00	I_0
01	I_1
02	I_2
03	I_3
10	I_4
11	I_5
12	I_6
13	I_7
20	I_8
21	I_9
22	I_{10}
23	I_{11}
30	I_{12}
31	I_{13}
32	I_{14}
33	I_{15}

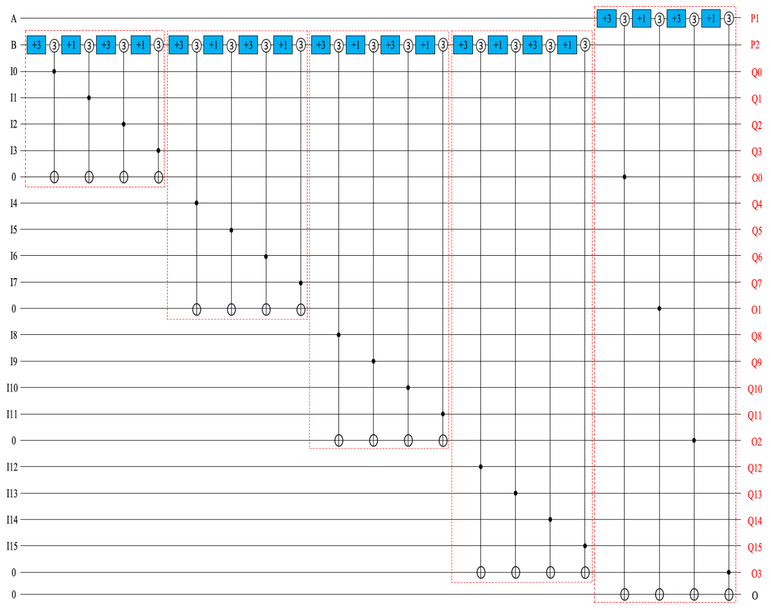
selectors are A and B. The main output is O, and the garbage outputs are P_1, P_2, O_0 to O_3, and Q_0 to Q_{15}. The outputs P_1 and P_2 are equal to A and B, respectively. Generally, the first realization of quaternary 3-qudit Controlled Feynman gates is used when inputs need to be restored. In this case, 20 quaternary Shift gates and 120 quaternary Muthukrishnan-Stroud gates are inserted in the circuit. Therefore, the quantum cost is 140. However, in multiplexer circuits, the inputs I_0 to I_{15} are unnecessary as outputs, so it is possible to use the second realization of quaternary-controlled Feynman gates. Therefore, the second realization of quaternary-controlled Feynman gates can be used, and the quantum cost is 120.

We could also combine some gates in designing a quaternary reversible 16×1 multiplexer and present a circuit with a lower quantum cost. As shown in Figure 10.2, an optimized multiplexer circuit can be realized. Eight quaternary Shift gates are used along with twenty quaternary Controlled Feynman gates. Due to the use of the second realization of Feynman gates, eight quaternary Shift gates and 100 quaternary Muthukrishnan-Stroud gates were used in total. This results in a quantum cost of 108. This innovative combination provides improvement over the first realization regarding the quantum cost. Moreover, in both realizations, the number of constant inputs is five, and the number of garbage outputs is 22.

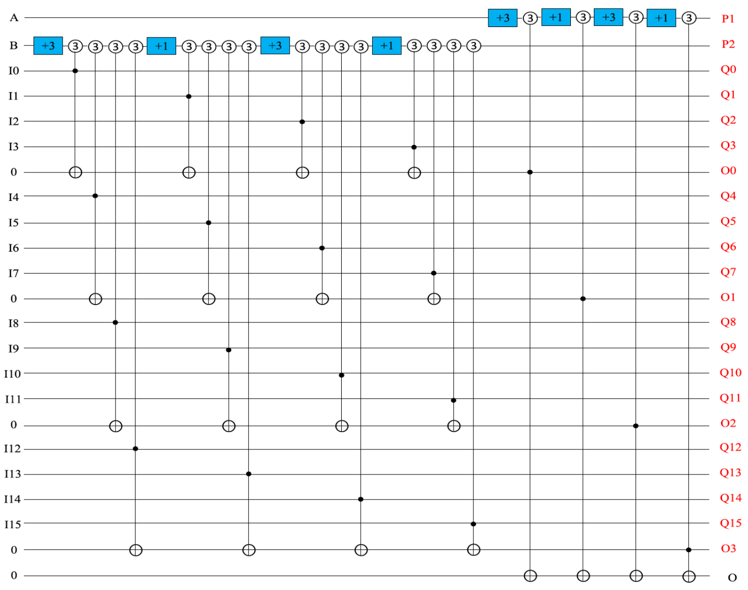
Based on our proposed quaternary reversible 4×1 multiplexer, we propose a generalized quaternary reversible $n \times 1$ multiplexer circuit, shown in Figure 10.3. Hence, our design is scalable. A quaternary $n \times 1$ multiplexer circuit consists of $n = 4^m$ inputs, m selectors, and only one output. In this circuit, m rows of 4×1 multiplexers are needed. The first row requires 4^{m-1} multiplexers, the second row requires 4^{m-2} multiplexers, and the mth row requires one multiplexer. Therefore, we can determine the number of 4×1 multiplexers needed to design our proposed $n \times 1$ multiplexer using geometric series formulas. The number of multiplexers is shown by P in equation (2):



(a) The logical architecture.



(b) The primary realization.



(c) The realization using M-S and Shift gates.

Figure 10.2: The proposed quaternary reversible 16×1 multiplexer circuit.

$$P = \sum_{i=0}^{m-1} 4^i = \frac{4^m - 1}{3} = \frac{n-1}{3} \tag{10.1}$$

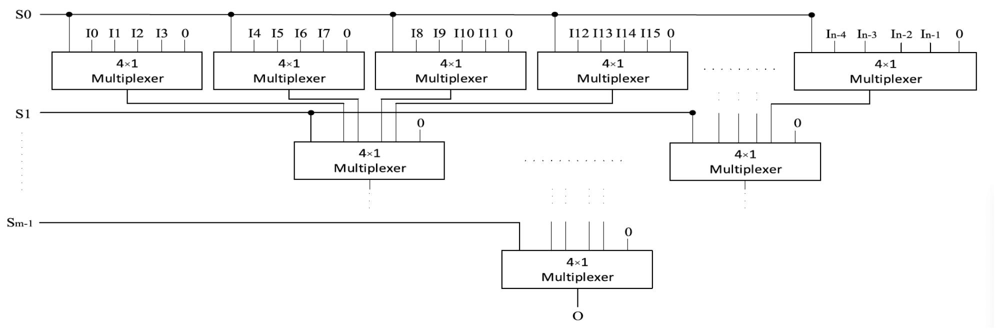


Figure 10.3: The logical architecture of the proposed quaternary reversible $n \times 1$ multiplexer circuit.

The quantum cost of a quaternary reversible $n \times 1$ multiplexer is $24\left(\frac{n-1}{3}\right)$, and it requires $\frac{n-1}{3}$ constant inputs and produces $\frac{3m+4n-4}{3}$ garbage outputs. We can combine the quaternary 1-qudit Shift gates in each row according to the mentioned optimization approach in the last part. In this way, we have four 1-qudit Shift gates in each row. We also have 4^{m-1} and 4^{m-2} Controlled Feynman gates in the first and the second rows, respectively. Moreover, in the last row, four Controlled Feynman gates are needed. Therefore, it can be concluded that in the proposed quaternary reversible $n \times 1$ multiplexer, $4\left(\frac{n-1}{3}\right)$ Controlled Feynman gates and $4m$ 1-qudit Shift gates are required, where n is the number of inputs and m is the number of selectors. Since we used the second realization of the Controlled Feynman gate, the total quantum cost of this optimized circuit is $20\left(\frac{n-1}{3}\right) + 4m$.

10.2.2 Proposed Quaternary Reversible Demultiplexer Circuit

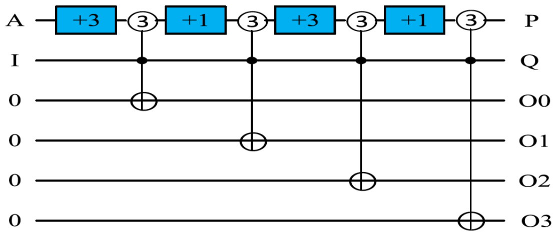
A demultiplexer performs the opposite function of a multiplexer. A quaternary demultiplexer with 4^m outputs has m select lines to send the input to the output. In a 1×4 demultiplexer, when the selector A is equal to 0, 1, 2, or 3, the output O_0, O_1, O_2, or O_3 is equal to I, respectively. Table 10.3 shows the truth table of the 1×4 quaternary demultiplexer.

Table 10.3: The truth table of a quaternary 1×4 demultiplexer.

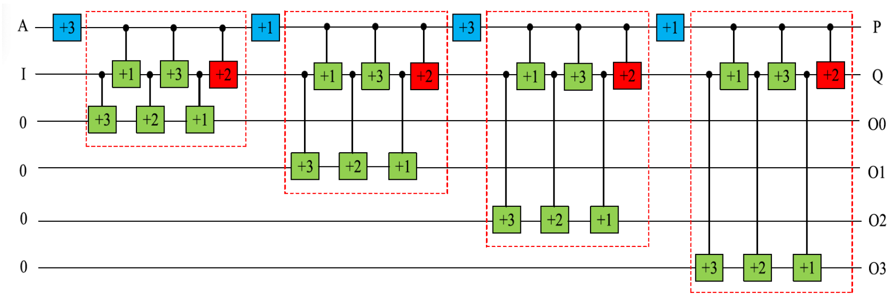
Selector	**Outputs**			
A	O_0	O_1	O_2	O_3
0	I	0	0	0
1	0	I	0	0
2	0	0	I	0
3	0	0	0	I

In Figure 10.4, we show the realization of our quaternary reversible demultiplexer circuit. Four quaternary 1-qudit Shift gates and four quaternary 3-qudit Controlled Feynman gates are utilized in this design. The main input is I, which requires four constant inputs, all of which are 0. The selector is A. O_0 to O_3 are the main outputs, and P and Q are the garbage outputs, which are equal to A and I, respectively. The first Controlled Feynman gate with a controlling value of 1 is applied when the selector is equal to 0, and the input I is sent to O_0. This circuit applies the controlling value of the second Controlled Feynman gate when the selector A is equal to 1, and the input I is sent to O_1. If the selector is equal to 2 or 3, the outputs O_2 and O_3 are equal to the input I, respectively.

Figure 10.4b shows how the proposed circuit is realized using quaternary Shift and Muthukrishnan-Stroud (M-S) gates. In this design, quaternary Controlled Feynman gates are shown by red boxes. Four quaternary 1-qudit Shift gates and twenty-four quaternary 2-qudit Muthukrishnan-Stroud gates are generally used. As a result, the quantum cost of the proposed quaternary reversible 1×4 demultiplexer circuit is 28. Considering that, in the multiplexer circuit, the input I does not need to be restored at the output Q, the red boxes can be removed, and the quantum cost is decreased by 24. In both cases, the number of constant inputs is four, and the number of garbage outputs is two.



(a) Symbol.



(b) The realization using M-S and Shift gates.

Figure 10.4: The proposed quaternary reversible 1×4 demultiplexer circuit.

We can also use our proposed quaternary 1×4 demultiplexer to construct a 1×16 demultiplexer. In this kind of demultiplexer, one input, two selectors, and 16 outputs are needed. The truth table of this circuit is shown in Table 10.4. The input is gated to the selected output based on a given combination of selectors A and B.

The logical architecture of the proposed quaternary reversible 1×16 demultiplexer, using 1×4 demultiplexers, is shown in Figure 10.5a. As can be seen, it requires five quaternary 1×4 demultiplexers. In this design, when the selector A is equal to 0, the main input is gated to one of the outputs in the first demultiplexer. One output of the second multiplexer is gated when selector A is equal to 1. In the third and fourth multiplexers, one output is gated if selector A is equal to 2 and

Table 10.4: The truth table of a quaternary 1×16 demultiplexer.

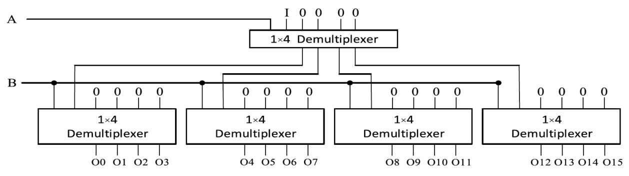
Selectors	**Outputs**															
AB	O_0	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8	O_9	O_{10}	O_{11}	O_{12}	O_{13}	O_{14}	O_{15}
00	I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01	0	I	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02	0	0	I	0	0	0	0	0	0	0	0	0	0	0	0	0
03	0	0	0	I	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	I	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	I	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	I	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	I	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	I	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	I	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	I	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	I	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	I	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	I	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	I	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	I

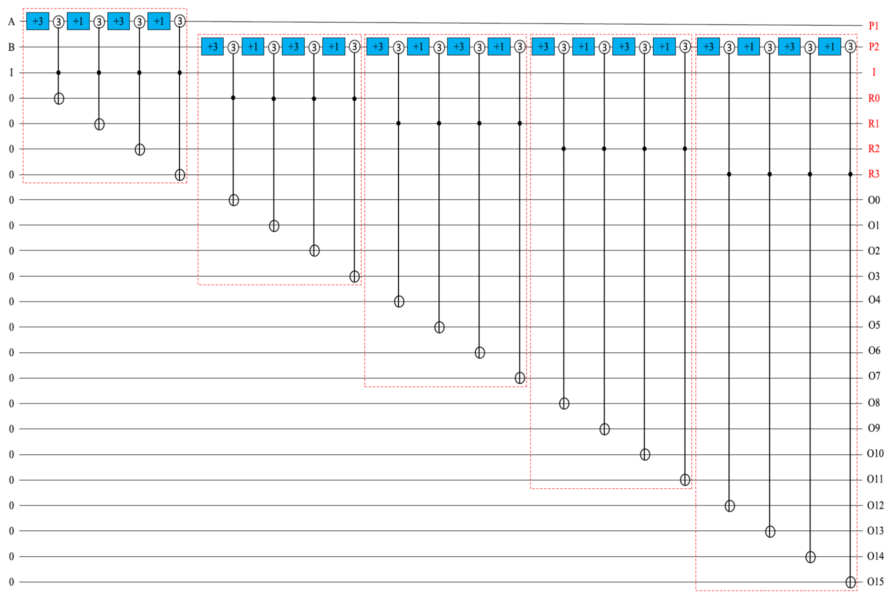
3, respectively. When the input B is equal to 0, the first input of the second row demultiplexers is activated. If the input B is equal to 1, then the second input of the demultiplexers is activated. Moreover, when B is equal to 2 and 3, the third and fourth inputs of the demultiplexers are activated, respectively.

The realization of the proposed quaternary reversible 1×16 demultiplexer using 1×4 demultiplexers is shown in Figure 10.5b. In the figure, red boxes show our proposed quaternary reversible 1×4 demultiplexer. The main input is I, and it requires twenty constant inputs, which are 0. The selectors are A and B. The main outputs are O_0 to O_{15}, and it produces seven garbage outputs that are P_1, P_2, I, and R_0 to R_3. The outputs P_1 and P_2 are equal to the selectors A and B, respectively. Generally, since input restoration is not necessary, the second realization of quaternary Controlled Feynman gates can be used. In this way, the proposed circuit includes 20 quaternary Shift gates and 100 quaternary Muthukrishnan-Stroud gates, resulting in a quantum cost of 120.

We could also use a lower number of gates for designing the quaternary reversible 1×16 demultiplexer and present a circuit with a lower quantum cost. The realization of the proposed optimized circuit is shown in Figure 10.5c. As can be seen, twenty quaternary Controlled Feynman gates and eight 1-qudit Shift gates are used. Since input restoration is not necessary, the second realization of quaternary Controlled Feynman gates is used, resulting in eight quaternary Shift gates and 100 quaternary Muthukrishnan-Stroud gates in the proposed design. The quantum cost is 108. Compared to our first proposed quaternary 1×16 demultiplexer, we improved the quantum cost using this innovative combination. The number of constant inputs and garbage outputs for both realizations is 20 and 7, respectively.

In addition, our proposed quaternary demultiplexer is scalable. A generalized quaternary reversible $1\times n$ demultiplexer circuit, based on the quaternary reversible 1×4 demultiplexer, is suggested. In a quaternary $1 \times n$ demultiplexer circuit, there is one input, m selectors, and $n = 4^m$ outputs. Generally, m rows of 1×4 demultiplexers are needed. It is necessary to use one demultiplexer in the first row, four demultiplexers in the second row, and 4^{m-1} demultiplexers in the last


(a) The logical architecture.


(b) The primary realization.

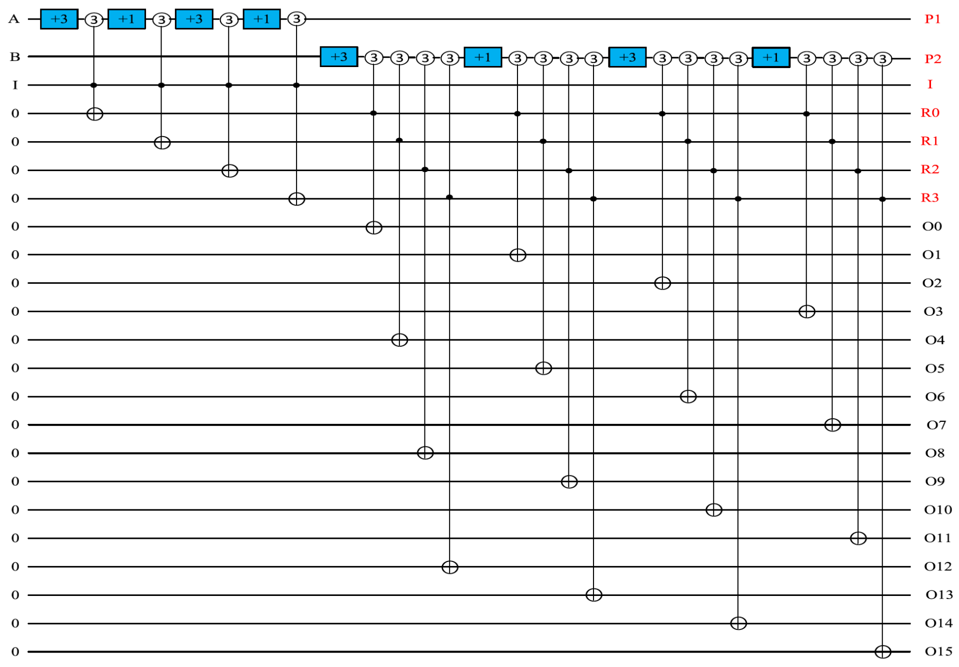

(c) The realization using M-S and Shift gates.

Figure 10.5: The proposed quaternary reversible 1 × 16 demultiplexer circuit.

row. Figure 10.6 shows the logical structure of the proposed $1 \times n$ demultiplexer. We can also use the geometric series formula to determine the number of 1×4 demultiplexers needed to design our proposed $1 \times n$ demultiplexer. Using equation (3), we can determine the number of demultiplexers, represented by Q:

$$Q = \sum_{i=0}^{m-1} 4^i = \frac{4^m - 1}{3} = \frac{n-1}{3} \tag{10.2}$$

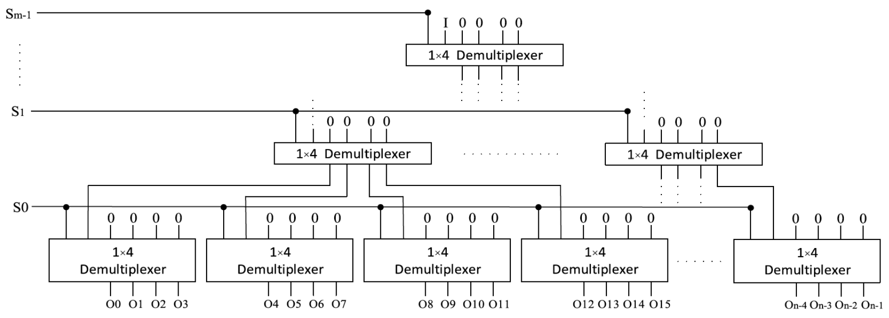


Figure 10.6: The logical architecture of the proposed quaternary reversible $1 \times n$ demultiplexer circuit.

The proposed quaternary reversible $1 \times n$ demultiplexer circuit requires $4\left(\frac{n-1}{3}\right)$ constant inputs and produces $\frac{n+3m-1}{3}$ garbage outputs, with a quantum cost of $24\left(\frac{n-1}{3}\right)$. Based on the optimization approach discussed in the previous section, the quaternary 1-qudit Shift gates in each row can be combined. As a result, each row contains four 1-qudit Shift gates. There are four and sixteen Controlled Feynman gates in the first and second rows, respectively, and 4^{m-1} Controlled Feynman gates in the last row. Therefore, it can be concluded that there are $4\left(\frac{n-1}{3}\right)$ Controlled Feynman gates and $4m$ 1-qudit Shift gates in the proposed quaternary reversible $1 \times n$ demultiplexer, with n outputs and m selectors. Since we use the second realization of the Controlled Feynman gate, this optimized circuit has a total quantum cost of $20\left(\frac{n-1}{3}\right) + 4m$.

10.3 Evaluation Results

In this section, we analyze our proposed realizations of quaternary reversible multiplexer and demultiplexer circuits and calculate the improvement rate with respect to the best results in the literature. We also compare the proposed circuits with the existing designs in [59], [74], and [72] in terms of quantum cost, number of garbage outputs, and number of constant inputs.

The following parts present the first comparison for our proposed quaternary reversible multiplexer and the second comparison for our proposed quaternary reversible demultiplexer.

10.3.1 Evaluation of Quaternary Reversible Multiplexer Circuits

According to Table 10.5, whereas both designs of quaternary reversible 4×1 multiplexer circuits have the same number of garbage outputs and constant inputs, the proposed circuit outperforms the existing design presented in [74] in terms of quantum cost due to its lower values for this parameter. Table 10.6 also illustrates that our proposed quaternary reversible 16×1 multiplexer circuit shows significant improvement in terms of quantum cost, number of garbage outputs, and number of constant inputs compared with its counterparts in [59], [74], and [72]. Therefore, it can be concluded that our proposed design of the 16×1 multiplexer in this chapter is much more efficient than the previous designs in [59], [74], and [72].

Table 10.5: Comparison between quaternary reversible 4×1 multiplexer designs.

	Proposed 4×1 **multiplexer**	**Improvement percentage**	[74]
Quantum cost	24	65%	70
Constant input number	1	0%	1
Garbage output number	5	0%	5

Table 10.6: Comparison between quaternary reversible 16×1 multiplexer designs.

	Proposed 16×1 **multiplexer**	**Improvement percentage**	[59]	[74]	[72]
Quantum cost	108	37%	174	368	580
Constant input number	5	37%	17	8	17
Garbage output number	22	12%	33	25	24

10.3.2 Evaluation of Quaternary Reversible Demultiplexer Circuits

Table 10.7 shows the comparison between our proposed quaternary reversible 1×4 demultiplexer and its counterpart in [74]. As can be seen, although both 1×4 demultiplexer circuits require four constant inputs and produce two garbage outputs, our proposed design has a quantum cost of 24, while the demultiplexer realization in [74] has a quantum cost of 58. Due to its lower quantum cost, our proposed quaternary reversible 1×4 demultiplexer is more efficient than the existing design in [74]. The results given in Table 10.8 show that our proposed quaternary reversible 1×16 demultiplexer has 20 constant inputs, even garbage outputs, and a quantum cost of 108. It is evident from Table 10.8 that our proposed design has lower quantum cost, garbage outputs, and constant inputs compared to the previous designs in [59], [74], and [72]. Since reversible circuits are more efficient when these parameters are minimized, the quaternary reversible 1×16 demultiplexer in this study is more efficient than its counterparts in [59], [74], and [72].

Table 10.7: Comparison between quaternary reversible 1×4 demultiplexer designs.

	Proposed 1×4 **demultiplexer**	**Improvement percentage**	[74]
Quantum cost	24	58%	58
Constant input number	4	0%	4
Garbage output number	2	0%	2

Table 10.8: Comparison between quaternary reversible 1×16 demultiplexer designs.

	Proposed 1×16 **demultiplexer**	**Improvement percentage**	[59]	[74]	[72]
Quantum cost	108	37%	174	308	580
Constant input number	20	13%	33	23	32
Garbage output number	7	30%	20	10	19

10.4 Conclusion

A new quaternary reversible 4×1 multiplexer circuit, based on quaternary 1-qudit Shift gates, 2-qudit Muthukrishnan–Stroud gates, and 3-qudit Controlled Feynman gates, has been presented in this chapter. The proposed 4×1 multiplexer has been utilized to design a quaternary reversible 16×1 multiplexer circuit. The proposed design is scalable for $n \times 1$ multiplexers. Moreover, we have introduced a new scalable realization of a 1×4 demultiplexer to design our proposed quaternary reversible 1×16 and $1 \times n$ demultiplexers. The proposed quaternary reversible circuits in the present study significantly decrease quantum cost, the number of constant inputs, and the number of garbage outputs. Since designing a reversible circuit with lower values of these parameters leads to increased efficiency, it can be concluded that our proposed multiplexer and demultiplexer circuits are more efficient compared to their existing counterparts.

Chapter 11

Conclusions

11.1 Concluding Remarks

This thesis provides a detailed examination of quantum circuit optimization using two interconnected strategies: leveraging functional regularities and decomposition methods, along with introducing multivalued reversible designs. These approaches effectively address critical challenges in quantum computing, particularly related to circuit size, quantum cost, and scalability of quantum architectures. By focusing on function-based synthesis and multivalued logic systems, this work establishes a foundation for more efficient, scalable, and practical quantum circuit designs, creating new opportunities for future advancements in the field.

In **Part I: Exploiting Function Regularities and Decomposition for Quantum Synthesis**, the focus is on reducing the cost of quantum circuits by utilizing inherent regularities in Boolean functions and applying decomposition techniques. Traditional quantum circuits, often created without fully considering the underlying structure of Boolean functions, tend to be less compact, resulting in higher quantum costs. This research focused on *dimension-reducible* and *autosymmetric functions*, showing how exploiting regularities in these functions can lead to the design of more efficient circuits, reducing both gate count and depth—two critical factors in quantum circuit performance.

A key innovation presented in this section was the introduction of the *Projected Sum of Product (PSOP)* technique, an algebraic approach for decomposing complex Boolean functions into simpler components. The PSOP decomposition method provided an effective strategy for simplifying quantum circuits, enhancing their viability on current quantum hardware. Theoretical models and experimental validation demonstrated that PSOP-based circuits achieve significant reductions in gate count and overall quantum cost, highlighting this approach as a valuable advancement in quantum circuit optimization

Essentially, **Part I** established the foundation for reducing the cost of quantum circuits by leveraging the mathematical properties of functions. This approach enables the development of more efficient and compact designs, which are essential as quantum systems expand and advance toward more complex applications. The reduction in quantum resources, particularly concerning gates and circuit depth, addresses a significant limitation of current quantum circuits and enhances the prospects for practical quantum computing applications.

Building upon the foundations established in **Part I**, **Part II: Multivalued**

Reversible Designs for Quantum Circuits shifted its focus to the investigation of *multivalued reversible logic systems*, particularly ternary (qutrit) and quaternary (qudit) logic circuits. While classical binary logic has served as the foundation for both classical and quantum computing, it has certain limitations, especially as quantum circuits become more complex. By transitioning to multivalued logic systems, this research illustrated a more efficient utilization of quantum resources, leading to further reductions in circuit size and complexity.

This section started with an overview of multivalued logic, laying the foundation for advanced circuit designs that take advantage of the additional states present in ternary and quaternary systems. The thesis presented novel designs for *qutrit-based multiplexers, demultiplexers, and decoders*, which achieved significant reductions in quantum cost, as well as in the number of garbage outputs and constant inputs, when compared to existing alternatives. The capacity of qutrits to store more information was leveraged in these circuits, resulting in more compact and efficient designs.

A significant highlight in **Part II** was the use of ternary logic in *quantum image processing.* Existing binary quantum circuits often face challenges with resource-intensive tasks like image manipulation, where ternary logic offers a considerable advantage. The integration of *ternary Shift gates* and *Muthukrishnan–Stroud gates* into quantum image circuits resulted in substantial reductions in quantum cost, constant inputs, and garbage outputs—key factors in enhancing circuit performance. This innovation paves the way for new applications of quantum computing, especially in fields that require complex data manipulation, such as quantum image processing.

Additional progress in *balanced ternary reversible comparator circuits* demonstrated the potential of balanced ternary logic in minimizing quantum costs. Comparators are essential elements in computational tasks, and the balanced ternary design introduced in this thesis accomplished remarkable reductions in key performance metrics, achieving a 65% decrease in quantum cost, 50% fewer constant inputs, and 33% fewer garbage outputs. These enhancements highlight the effectiveness of multivalued logic in optimizing crucial operations within quantum systems.

The final chapter of **Part II** examined *quaternary reversible circuits* for multiplexers and demultiplexers, further enhancing the advantages of multivalued logic systems for scalable designs. Quaternary logic, represented by qudits, enables even greater reductions in circuit complexity compared to ternary logic. The proposed scalable quaternary multiplexers and demultiplexers showed marked improvements in quantum cost, as well as in the number of garbage outputs and constant inputs, relative to existing designs. These findings indicate that as quantum systems increase in size and complexity, quaternary logic will become increasingly essential for maintaining scalable and efficient circuit designs.

In conclusion, this thesis has significantly advanced the field of quantum circuit design by integrating two complementary approaches: *exploiting function regularities* and *adopting multivalued logic systems.* The implementation of function decomposition techniques, especially the PSOP method, has demonstrated a notable reduction in quantum circuit complexity and cost. Meanwhile, the incorporation of ternary and quaternary logic systems has created new opportunities for developing more efficient and scalable quantum circuit designs.

These innovations present significant potential for both theoretical progress and practical applications as quantum technologies advance. The proposed methods provide avenues for addressing critical challenges in quantum computing, such as the substantial resource demands of current circuits and the constraints of classical binary logic within quantum systems. As the field approaches the development of viable quantum computers, the insights and designs outlined in this thesis will form a solid foundation for future innovations in quantum hardware and algorithm development.

In conclusion, this thesis has established a comprehensive framework for enhancing the efficiency and scalability of quantum circuits. It addresses critical challenges related to quantum cost, circuit complexity, and resource optimization, equipping researchers and engineers with valuable tools to advance the field of quantum computing. The novel methods and designs presented here are sure to play a significant role in the ongoing effort to make quantum computing a practical and transformative technology for addressing some of the most complex computational problems faced globally.

11.2 Future Directions

The research presented in this thesis has significantly enhanced our understanding of quantum circuit optimization through innovative approaches such as PSOP decomposition, D-reducible and Autosymmetric functions, and multivalued reversible logic. The findings indicate substantial improvements in circuit efficiency and cost reduction, paving the way for several promising avenues for future research and development.

One key avenue for future work involves further refining and expanding decomposition techniques. While this thesis has established effective methods for simplifying quantum circuits using PSOP forms, subsequent research could investigate deriving new decomposed forms from alternative representations, such as AND-inverter graphs (AIGs) or Reduced Ordered Binary Decision Diagrams (ROBDDs). This exploration could extend the applicability of PSOP decomposition and enhance its effectiveness across a broader spectrum of Boolean functions. Additionally, examining Projected Exclusive Sum of Products (PESOP) forms as alternatives to traditional Exclusive Sum of Products (ESOP) expressions may yield new insights and methodologies for reversible logic synthesis. Combining these decomposition techniques with quantum error correction methods could further enhance the fault tolerance and robustness of quantum circuits.

At the same time, the investigation of D-reducible and Autosymmetric functions has highlighted the potential for creating compact reversible circuits by utilizing specific regularities. Future research should aim to identify additional regularities in Boolean functions that could be harnessed for optimizing quantum circuits. This entails analyzing other classes of functions and their properties to uncover new optimization possibilities. Furthermore, assessing alternative decomposition techniques and their advantages may provide complementary or superior approaches for deriving compact quantum circuits. Implementing these techniques across various Boolean function representations, such as multi-level logic networks, could yield valuable insights into their effectiveness and adaptability.

The advancement of multivalued reversible logic systems, including ternary (qutrit) and quaternary (qudit) circuits, also offers exciting prospects for future research. The theoretical aspects of the novel multivalued circuits developed in this work are essential for evaluating their real-world performance and scalability. Future studies should also concentrate on creating advanced multivalued logic gates, aiming to further decrease quantum cost and enhance circuit efficiency. Exploring the application of multivalued logic in specific quantum algorithms, such as quantum image processing and quantum simulations, will help illustrate its practical significance and potential for addressing complex computational challenges.

As quantum computing technology evolves, the integration of these optimized techniques into larger and more complex quantum systems will become increasingly crucial. Research initiatives should focus on overcoming the challenges associated with scaling quantum circuit designs, such as circuit size, gate count, and computational resources. Furthermore, creating scalable quantum architectures that effectively leverage multivalued circuits and integrating optimized circuits into hybrid quantum-classical systems could significantly improve overall system performance and enhance the efficiency of solving real-world problems.

In summary, the future research directions in quantum circuit optimization are extensive and promising. By advancing PSOP decomposition techniques, improving quantum compilation through the exploitation of regularities, and investigating multivalued reversible logic systems, researchers can continue to advance the field. Collaborative and practical approaches will be essential for unlocking the full potential of quantum computing and addressing the challenges posed by future quantum technologies.

Bibliography

[1] Reza Akbari-Hasanjani and Reza Sabbaghi-Nadooshan. New design of binary to ternary converter. *IETE Journal of Research*, 69(4):2212–2223, 2023.

[2] A. Al-Rabadi, L. Casperson, M. Perkowski, and X. Song. Multiple-valued quantum logic. *Quantum*, 10(2):1, 2002.

[3] Matthew Amy, Dmitri Maslov, Michele Mosca, and Martin Roetteler. A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(6):818–830, 2013.

[4] M. Anderlini, P. J. Lee, B. L. Brown, J. Sebby-Strabley, W. D. Phillips, and J. V. Porto. Controlled exchange interaction between pairs of neutral atoms in an optical lattice. *arXiv preprint arXiv:0708.2073*, 2007.

[5] Mohammad-Ali Asadi, Mohammad Mosleh, and Majid Haghparast. A novel reversible ternary coded decimal adder/subtractor. *Journal of Ambient Intelligence and Humanized Computing*, 12:7745–7763, 2021.

[6] Mohammad-Ali Asadi, Mohammad Mosleh, and Majid Haghparast. Toward novel designs of reversible ternary 6: 2 compressor using efficient reversible ternary full-adders. *The Journal of Supercomputing*, 77:5176–5197, 2021.

[7] Mohammad-Ali Asadi, Mohammad Mosleh, and Majid Haghparast. Towards designing quantum reversible ternary multipliers. *Quantum Information Processing*, 20(7):226, 2021.

[8] A. Ashikhmin and E. Knill. Nonbinary quantum stabilizer codes. *IEEE Transactions on Information Theory*, 47(7):3065–3072, 2001.

[9] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52(5):3457, 1995.

[10] Helle Bechmann-Pasquinucci and Asher Peres. Quantum cryptography with 3-state systems. *Physical Review Letters*, 85(15):3313, 2000.

[11] C. Bennett. Logical reversibility of computation. *IBM Journal of Research and Development*, 17:525–532, 1973.

[12] A. Bernasconi. The optimization of kep-sops: Computational complexity, approximability and experiments. *ACM Transactions on Design Automation of Electronic Systems*, 13(2), 2008.

[13] A. Bernasconi, A. Berti, V. Ciriani, G. De 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*, pages 1–1, 2022.

[14] A. Bernasconi, S. Cimato, V. Ciriani, and M. C. Molteni. Multiplicative complexity of autosymmetric functions: Theory and applications to security. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

[15] A. Bernasconi and V. Ciriani. Dredsop: Synthesis of a new class of regular functions. In *9th EUROMICRO Conference on Digital System Design (DSD'06)*, pages 377–384. IEEE, 2006.

[16] A. Bernasconi and V. Ciriani. Logic synthesis and testability of d-reducible functions. In *2010 18th IEEE/IFIP International Conference on VLSI and System-on-Chip*, pages 280–285. IEEE, 2010.

[17] A. Bernasconi and V. Ciriani. Dimension-reducible boolean functions based on affine spaces. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 16(2):1–21, 2011.

[18] A. Bernasconi and V. Ciriani. Multiplicative complexity of xor based regular functions. *IEEE Transactions on Computers*, 71(11):2927–2939, 2022.

[19] A. Bernasconi, V. Ciriani, F. Luccio, and L. Pagli. Exploiting regularities for boolean function synthesis. *Theory of Computing Systems*, 39:485–501, 2006.

[20] A. Bernasconi, V. Ciriani, F. Luccio, and L. Pagli. Synthesis of autosymmetric functions in a new three-level form. *Theory of Computing Systems*, 42:450–464, 2008.

[21] A. Bernasconi, V. Ciriani, and L. Monfrini. Autosymmetric and d-reducible functions: Theory and application to security. In *Advanced Boolean Techniques: Selected Papers from the 15th International Workshop on Boolean Problems*, pages 95–110. Springer, 2023.

[22] Anna Bernasconi, Alessandro Berti, Valentina Ciriani, Gianna M. Del Corso, and Innocenzo Fulginiti. XOR-AND-XOR logic forms for autosymmetric functions and applications to quantum computing. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 42(6):1861–1872, 2023.

[23] Anna Bernasconi, Stelvio Cimato, Valentina Ciriani, and Maria Chiara Molteni. Multiplicative complexity of XOR based regular functions. *IEEE Trans. Computers*, 71(11):2927–2939, 2022.

[24] Anna Bernasconi, Valentina Ciriani, and Roberto Cordone. On Projecting Sums of Products. In *11th Euromicro Conference on Digital Systems Design: Architectures, Methods and Tools*, 2008.

[25] Anna Bernasconi, Valentina Ciriani, and Roberto Cordone. The optimization of kEP-SOPs: Computational complexity, approximability and experiments. *ACM Trans. Design Autom. Electr. Syst.*, 13(2), 2008.

[26] Anna Bernasconi, Valentina Ciriani, Gianmarco Cuciniello, and Asma Taheri Monfared. On exploiting psop decomposition for quantum synthesis. *WiPiEC Journal-Works in Progress in Embedded Computing Journal*, 10(2), 2024.

[27] Anna Bernasconi, Valentina Ciriani, Asma Taheri Monfared, and Stefano Zanoni. Compact quantum circuits for dimension reducible functions. In *26th Euromicro Conference on Digital System Design, DSD 2023, Golem, Albania, September 6-8, 2023*, pages 776–781. IEEE, 2023.

[28] Anna Bernasconi, Valentina Ciriani, Gabriella Trucco, and Tiziano Villa. On decomposing boolean functions via extended cofactoring. In *2009 Design, Automation & Test in Europe Conference & Exhibition*, pages 1464–1469. IEEE, 2009.

[29] J. C. Bioch. The complexity of modular decomposition of boolean functions. *Discrete Applied Mathematics*, 149(1-3):1–13, 2005.

[30] Mohamed Bourennane, Anders Karlsson, and Gunnar Björk. Quantum key distribution using multilevel encoding. *Physical Review A*, 64(1):012306, 2001.

[31] Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. Tight bounds on quantum searching. *Fortschritte der Physik: Progress of Physics*, 46(4-5):493–505, 1998.

[32] R.K. Brayton, G.D. Hachtel, C. McMullen, and A.L. Sangiovanni-Vincentelli. *Logic Minimization Algorithms for VLSI Synthesis.* Kluwer Academic Publishers, 1984.

[33] Du˘sanka Bundalo, Zlatko Bundalo, and Branimir Dordevi´c. Design of quaternary logic systems and circuits. *Facta Universitatis: Series Electronics and Energetics*, 18(1):45–56, 2005.

[34] Tanay Chattopadhyay. Design of optical reconfigurable balanced ternary arithmetic logic unit using mems based design. *Optics Communications*, 356:123–135, 2015.

[35] G. Chen, Y. Wang, L. Jian, Y. Zhou, and S. Liu. Ternary quantum key distribution protocol based on hadamard gate. *International Journal of Theoretical Physics*, 61(2):1–13, Feb. 2022.

[36] G. L. Chen, X. H. Song, S. E. Venegas-Andraca, and A. A. Abd El-Latif. Qirhsi: Novel quantum image representation based on hsi color space model. *Quantum Information Processing*, 21(1):5, 2022.

[37] Donny Cheung, Dmitri Maslov, Jimson Mathew, and Dhiraj K Pradhan. On the design and optimization of a quantum polynomial-time attack on elliptic curve cryptography. In *Workshop on Quantum Computation, Communication, and Cryptography*, pages 96–104. Springer, 2008.

[38] V. Ciriani. Synthesis of spp three-level logic networks using affine spaces. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(10):1310–1323, 2003.

[39] P. M. Cohn. *Algebra: v. 1.* John Wiley & Sons, Chichester, England, 2nd edition, Jun. 1982.

[40] S. Das and F. Caruso. A hybrid-qudit representation of digital rgb images. *Scientific Reports*, 13(1):13671, 2023.

[41] Vitaly Deibuk and Andrij Biloshytskyi. Genetic synthesis of new reversible/quantum ternary comparator. *Advances in Electrical and Computer Engineering*, 15(3):147–152, 2015.

[42] Hao Dong, Dayong Lu, and Chan Li. A novel qutrit representation of quantum image. *Quantum Information Processing*, 21(3):108, 2022.

[43] A. Norouzi Doshanlou, M. Haghparast, and M. Hosseinzadeh. Novel quaternary quantum reversible half adder and full adder circuits. *IETE Journal of Research*, 68(2):1525–1531, Mar. 2022.

[44] A. Norouzi Doshanlou, M. Haghparast, M. Hosseinzadeh, and M. Reshadi. Efficient binary to quaternary and vice versa converters: Embedding in quaternary arithmetic circuits. *Journal of Supercomputing*, 77(12):14600–14616, Dec. 2021.

[45] Rolf Drechsler, Alexander Finder, and Robert Wille. Improving ESOP-based synthesis of reversible logic using evolutionary algorithms. In *European Conference on the Applications of Evolutionary Computation*, pages 151–161. Springer, 2011.

[46] Ehsan Faghih, MohammadReza Taheri, Keivan Navi, and Nader Bagherzadeh. Efficient realization of quantum balanced ternary reversible multiplier building blocks: A great step towards sustainable computing. *Sustainable Computing: Informatics and Systems*, 40:100908, 2023.

[47] Kenneth Fazel, Mitchell A Thornton, and Jacqueline E Rice. ESOP-based Toffoli gate cascade generation. In *2007 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pages 206–209. IEEE, 2007.

[48] A. Fedorov, L. Steffen, M. Baur, M. P. da Silva, and A. Wallraff. Implementation of a toffoli gate with superconducting circuits. *arXiv preprint arXiv:1108.3966*, 2011.

[49] A. Galda, M. Cubeddu, N. Kanazawa, P. Narang, and N. Earnest-Noble. Implementing a ternary decomposition of the toffoli gate on fixed-frequency transmon qutrits. *arXiv preprint arXiv:2109.00558*, 2021.

[50] W. B. Gao, P. Xu, X. C. Yao, O. Gühne, A. Cabello, C. Y. Lu, and J. W. Pan. Experimental realization of a controlled-not gate with four-photon six-qubit cluster states. *Physical Review Letters*, 104(2):020501, 2010.

[51] N. Gershenfeld. Signal entropy and the thermodynamics of computation. *IBM Systems Journal*, 35(3.4):577–586, 1996.

[52] S. M. Ghadamgahi, R. Sabbaghi-Nadooshan, and K. Navi. Novel ternary adders and subtractors in quantum cellular automata. *Journal of Supercomputing*, 78:1–43, Jun. 2022.

[53] Rafael C. Gonzalez. *Digital Image Processing.* Pearson Education India, 2009.

[54] Eiichi Goto and Hidetosi Takahasi. Some theorems useful in threshold logic for enumerating boolean functions. In *IFIP congress*, pages 747–752, 1962.

[55] Andrew D. Greentree, Simon G. Schirmer, F. Green, Lloyd C. L. Hollenberg, A. R. Hamilton, and R. G. Clark. Maximizing the hilbert space for a finite number of distinguishable quantum states. *Physical Review Letters*, 92(9):097901, 2004.

[56] A. M. Grigoryan and S. S. Agaian. New look on quantum representation of images: Fourier transform representation. *Quantum Information Processing*, 19(5):148, 2020.

[57] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[58] Pallav Gupta, Abhinav Agrawal, and Niraj K Jha. An algorithm for synthesis of reversible logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(11):2317–2330, 2006.

[59] M. Haghparast and A. T. Monfared. Novel quaternary quantum decoder, multiplexer and demultiplexer circuits. *International Journal of Theoretical Physics*, 56(5):1694–1707, May 2017.

[60] Majid Haghparast and Asma Taheri Monfared. Designing novel quaternary quantum reversible subtractor circuits. *International Journal of Theoretical Physics*, 57(1):226–237, 2018.

[61] Majid Haghparast, Robert Wille, and Asma Taheri Monfared. Towards quantum reversible ternary coded decimal adder. *Quantum Information Processing*, 16:1–25, 2017.

[62] M. E. Haque, M. Paul, A. Ulhaq, and T. Debnath. Advanced quantum image representation and compression using a dct-efrqi approach. *Scientific Reports*, 13(1):4129, 2023.

[63] Hurst. Multiple-valued logic—its status and its future. *IEEE transactions on Computers*, 100(12):1160–1179, 1984.

[64] Muhammad Ilyas, Shuxian Cui, and Marek Perkowski. Ternary logic design in topological quantum computing. *arXiv preprint arXiv:2204.01000*, 2022.

[65] M.Z. Jahangir and J. Mounika. Design and simulation of an innovative cmos ternary 3 to 1 multiplexer and the design of ternary half adder using ternary 3 to 1 multiplexer. *Microelectronics Journal*, 90:82–87, 2019.

[66] Ugur Kalay, Douglas V Hall, and Marek A Perkowski. A minimal universal test set for self-test of exor-sum-of-products circuits. *IEEE Transactions on Computers*, 49(3):267–276, 2000.

[67] Pawel Kerntopf. New generalizations of shannon decomposition. In *Int. Workshop on Applications of Reed-Muller Expansion in Circuit Design*, pages 109–118, 2001.

[68] M. H. Khan. Design of reversible/quantum ternary multiplexer and demultiplexer. *Engineering Letters*, 13(3), 2006.

[69] M. H. Khan. Design of reversible/quantum ternary comparator circuits. *Engineering Letters*, 16(2), 2008.

[70] M. H. Khan, M. A. Perkowski, and P. Kerntopf. Multi-output galois field sum of products synthesis with new quantum cascades. In *33rd International Symposium on Multiple-Valued Logic, 2003. Proceedings*, pages 146–153. IEEE, May 2003.

[71] M. H. A. Khan. A recursive method for synthesizing quantum/reversible quaternary parallel adder/subtractor with look-ahead carry. *Journal of Systems Architecture*, 54(12):1113–1121, Dec. 2008.

[72] M. H. A. Khan. Reversible realization of quaternary decoder, multiplexer, and demultiplexer circuits. In *Proceedings of the 38th International Symposium on Multiple Valued Logic (ISMVL)*, pages 208–213, May 2008.

[73] M. H. A. Khan. Synthesis of quaternary reversible/quantum comparators. *Journal of Systems Architecture*, 54(10):977–982, Oct. 2008.

[74] M. H. A. Khan. Scalable architectures for design of reversible quaternary multiplexer and demultiplexer circuits. In *Proceedings of the 39th International Symposium on Multiple-Valued Logic*, pages 343–348, 2009.

[75] M. H. A. Khan and M. A. Perkowski. GF(4) based synthesis of quaternary reversible/quantum logic circuits. In *Proceedings of the 37th International Symposium on Multiple-Valued Logic (ISMVL)*, page 11, May 2007.

[76] M. H. A. Khan, H. Thapliyal, and E. Munoz-Coreas. Automatic synthesis of quaternary quantum circuits. *Journal of Supercomputing*, 73(5):1733–1759, May 2017.

[77] Md Mahmud Muntakim Khan, Ayan Kumar Biswas, Shuvro Chowdhury, Mehbuba Tanzid, Kazi Mohammad Mohsin, Masud Hasan, and Asif Islam Khan. Quantum realization of some quaternary circuits. In *TENCON 2008-2008 IEEE Region 10 Conference*, pages 1–5. IEEE, 2008.

[78] Mohammad H. Khan and Marek Perkowski. Quantum realization of ternary encoder and decoder. In *Proc. 7th Int. Symp. Representations and Methodology of Future Computing Technologies (RM2005)*, Tokyo, Japan, 2005.

[79] Mozammel HA Khan and Marek A Perkowski. Quantum ternary parallel adder/subtractor with partially-look-ahead carry. *Journal of Systems Architecture*, 53(7):453–464, 2007.

[80] AB Klimov, R Guzmán, JC Retamal, and Carlos Saavedra. Qutrit quantum computer with trapped ions. *Physical Review A*, 67(6):062313, 2003.

[81] Emanuel Knill. Fault-tolerant postselected quantum computation: Schemes. *arXiv preprint quant-ph/0402171*, 2004.

[82] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free xor gates and applications. In *International Colloquium on Automata, Languages, and Programming*, pages 486–498. Springer, 2008.

[83] R. Landauer. Irreversibility and heat generation in the computation process. *IBM Journal of Research and Development*, 5:183–191, 1961.

[84] P. Q. Le, F. Dong, and K. Hirota. A flexible representation of quantum images for polynomial preparation, image compression, and processing operations. *Quantum Information Processing*, 10:63–84, 2011.

[85] Robert A Liebler. *Basic matrix algebra with algorithms and applications.* Chapman and Hall/CRC, 2018.

[86] Z. H. Liu, H. W. Chen, J. Xu, W. J. Liu, and Z. Q. Li. High-dimensional deterministic multiparty quantum secret sharing without unitary operations. *Quantum Information Processing*, 11:1785–1795, 2012.

[87] F. Luccio and L. Pagli. On a new boolean function with applications. *IEEE Transactions on Computers*, 48(3):296–310, 1999.

[88] D. Maslov. Advantages of using relative-phase Toffoli gates with an application to multiple control Toffoli optimization. *Physical Review A*, 93:022311, 2016.

[89] Dmitri Maslov, Gerhard W Dueck, and D Michael Miller. Synthesis of fredkin-toffoli reversible networks. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 13(6):765–769, 2005.

[90] P. Mercy Nesa Rani and P. L. Thangkhiew. An overview of different approaches for ternary reversible logic circuits synthesis using ternary reversible gates with special reference to virtual reality. In *Advances in Augmented Reality and Virtual Reality*, pages 73–90. Jan. 2022.

[91] G. Meuli, M. Soeken, E. Campbell, M. Roetteler, and G. D. Micheli. The role of multiplicative complexity in compiling low t-count oracle circuits. In D. Z. Pan, editor, *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*, pages 1–8, Westminster, CO, USA, November 2019. ACM.

[92] Giulia Meuli, Mathias Soeken, and Giovanni De Micheli. Xor-and-inverter graphs for quantum compilation. *npj Quantum Information*, 8(1):7, 2022.

[93] D. M. Miller and M. A. Thornton. Multiple valued logic: Concepts and representations. *Synthesis Lectures on Digital Circuits and Systems*, 2(1):1–127, Jan. 2007.

[94] D Michael Miller, Dmitri Maslov, and Gerhard W Dueck. A transformation based algorithm for reversible logic synthesis. In *Proceedings 2003. design automation conference (ieee cat. no. 03ch37451)*, pages 318–323. IEEE, 2003.

[95] D Michael Miller, Mathias Soeken, and Rolf Drechsler. Mapping ncv circuits to optimized clifford+t circuits. In *International Conference on Reversible Computation*, pages 163–175. Springer, 2014.

[96] D Michael Miller and Mitchell A Thornton. *Multiple-Valued Logic: Concepts and Representations.* Springer Nature, 2022.

[97] Alan Mishchenko and Marek Perkowski. Fast heuristic minimization of exclusive-sums-of-products. *5th International Reed-Muller Workshop*, 2001.

[98] Takaaki Mizuki, Taro Otagiri, and Hideaki Sone. An application of ESOP expressions to secure computations. *Journal of Circuits, Systems, and Computers*, 16(02):191–198, 2007.

[99] Majid Mohammadi and Mohammad Eshghi. On figures of merit in reversible and quantum logic designs. *Quantum Information Processing*, 8:297–318, 2009.

[100] B. Mondal, P. Sarkar, P. K. Saha, and S. Chakraborty. Synthesis of balanced ternary reversible logic circuit. In *2013 IEEE 43rd International Symposium on Multiple-Valued Logic*, pages 334–339. IEEE, May 2013.

[101] A. T. Monfared and M. Haghparast. Design of new quantum/reversible ternary subtractor circuits. *Journal of Circuits, Systems, and Computers*, 25(02):1650014, 2016.

[102] Asma Taheri Monfared, Valentina Ciriani, Tommi Mikkonen, and Majid Haghparast. Quaternary reversible circuit optimization for scalable multiplexer and demultiplexer. *IEEE Access*, 11:46592–46603, 2023.

[103] Asma Taheri Monfared and Majid Haghparast. Novel design of quantum/reversible ternary comparator circuits. *Journal of Computational and Theoretical Nanoscience*, 12(12):5670–5673, 2015.

[104] Asma Taheri Monfared and Majid Haghparast. Designing new ternary reversible subtractor circuits. *Microprocessors and Microsystems*, 53:51–56, 2017.

[105] Asma Taheri Monfared and Majid Haghparast. Quantum ternary multiplication gate (qtmg): toward quantum ternary multiplier and a new realization for ternary toffoli gate. *Journal of Circuits, Systems and Computers*, 29(05):2050071, 2020.

[106] Ashok Muthukrishnan and Carlos R Stroud Jr. Multivalued logic gates for quantum computation. *Physical review A*, 62(5):052309, 2000.

[107] A. Navidi, R. Sabbaghi-Nadooshan, and M. Dousti. Introducing an innovative d flip-flop for designing quaternary qca register. *Journal of Intelligent Procedures in Electrical Technology*, 13(49):91–101, May. 2022.

[108] Alireza Navidi, Reza Sabbaghi-Nadooshan, and Massoud Dousti. Introducing an innovative d flip-flop for designing quaternary qca register. *Journal of Intelligent Procedures in Electrical Technology*, 13(49):91–101, 2022.

[109] M. Nielsen and I. Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, Cambridge, 2000.

[110] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, 2016.

[111] Abdollah Norouzi Doshanlou, Majid Haghparast, and Mehdi Hosseinzadeh. Novel quaternary quantum reversible half adder and full adder circuits. *IETE Journal of Research*, 68(2):1525–1531, 2022.

[112] Alexandru Paler and Robert Basmadjian. Energy cost of quantum circuit optimisation: predicting that optimising shor's algorithm circuit uses 1 gwh. *ACM Transactions on Quantum Computing*, 3(1):1–14, 2022.

[113] Mohammad Mehdi Panahi, Omid Hashemipour, and Keivan Navi. A novel design of a ternary coded decimal adder/subtractor using reversible ternary gates. *Integration*, 62:353–361, 2018.

[114] Mohammad Mehdi Panahi, Omid Hashemipour, and Keivan Navi. A novel design of a multiplier using reversible ternary gates. *IETE Journal of Research*, 67(6):744–753, 2021.

[115] George Papakonstantinou. A parallel algorithm for minimizing ESOP expressions. *Journal of Circuits, Systems, and Computers*, 23(01):1450015, 2014.

[116] KG Papakonstantinou and G Papakonstantinou. A nonlinear integer programming approach for the minimization of boolean expressions. *Journal of Circuits, Systems and Computers*, 27(10):1850163, 2018.

[117] J. H. Plantenberg, P. C. De Groot, C. J. P. M. Harmans, and J. E. Mooij. Demonstration of controlled-not quantum gates on a pair of superconducting quantum bits. *Nature*, 447(7146):836, 2007.

[118] A Raja, K Mukherjee, and JN Roy. Design of dual semiconductor optical amplifier structure based all-optical standard quaternary inverter and quaternary clocked sr flip-flop. *Optical and Quantum Electronics*, 54(1):39, 2022.

[119] Heinz Riener, Rüdiger Ehlers, Bruno de O Schmitt, and Giovanni De Micheli. Exact synthesis of ESOP forms. In *Advanced boolean techniques*, pages 177–194. Springer, 2020.

[120] Mehdi Saeedi and Igor L. Markov. Synthesis and Optimization of Reversible Circuits - A Survey. *ACM Comput. Surv.*, 45(2):21:1–21:34, 2013.

[121] Amit Saha, Anupam Chattopadhyay, and Amlan Chakrabarti. Robust quantum arithmetic operations with intermediate qutrits in the nisq-era. *International Journal of Theoretical Physics*, 62(4):92, 2023.

[122] Marinos Sampson, Marios Kalathas, Dimitrios Voudouris, and G Papakonstantinou. Exact ESOP expressions for incompletely specified functions. *Integration*, 45(2):197–204, 2012.

[123] Zarin Tasnim Sandhie, Jill Arvindbhai Patel, Farid Uddin Ahmed, and Masud H Chowdhury. Investigation of multiple-valued logic technologies for beyond-binary era. *ACM Computing Surveys (CSUR)*, 54(1):1–30, 2021.

[124] J. Sang, S. Wang, and Q. Li. A novel quantum representation of color digital images. *Quantum Information Processing*, 16:1–14, 2017.

[125] Tsutomu Sasao. Exmin2: a simplification algorithm for exclusive-or-sum-of-products expressions for multiple-valued-input two-valued-output functions. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 12(5):621–632, 1993.

[126] Tsutomu Sasao. Logic synthesis with exor gates. In *Logic Synthesis and Optimization*, pages 259–285. Springer, 1993.

[127] Tsutomu Sasao. Representations of logic functions using exor operators. *Representations of discrete functions*, pages 29–54, 1996.

[128] Bruno Schmitt, Ali Javadi-Abhari, and Giovanni De Micheli. Compilation flow for classically defined quantum operations. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 964–967. IEEE, 2021.

[129] Bruno Schmitt, Fereshte Mozafari, Giulia Meuli, Heinz Riener, and Giovanni De Micheli. From boolean functions to quantum circuits: A scalable quantum compilation flow in C++. In *Design, Automation & Test in Europe Conference & Exhibition*, pages 1044–1049. IEEE, 2021.

[130] Bruno Schmitt, Mathias Soeken, Giovanni De Micheli, and Alan Mishchenko. Scaling-up esop synthesis for quantum compilation. In *2019 IEEE 49th International Symposium on Multiple-Valued Logic (ISMVL)*, pages 13–18. IEEE, 2019.

[131] Peter W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 124–134. IEEE, 1994.

[132] X. Song, S. Wang, and X. Niu. Multi-channel quantum image representation based on phase transform and elementary transformations. *Journal of Information Hiding and Multimedia Signal Processing*, 5(4):574–585, 2014.

[133] Robert W Spekkens and Terry Rudolph. Degrees of concealment and bindingness in quantum bit commitment protocols. *Physical Review A*, 65(1):012310, 2001.

[134] Asma Taheri Monfared, Valentina Ciriani, and Majid Haghparast. Qutrit representation of quantum images: new quantum ternary circuit design. *Quantum Information Processing*, 23(8):288, 2024.

[135] Asma Taheri Monfared, Valentina Ciriani, Lauri Kettunen, and Majid Haghparast. Novel qutrit circuit design for multiplexer, de-multiplexer, and decoder. *Quantum Information Processing*, 22(1):12, 2022.

[136] Asma Taheri Monfared, Majid Haghparast, and Kamalika Datta. Quaternary quantum/reversible half-adder, full-adder, parallel adder and parallel adder/subtractor circuits. *International Journal of Theoretical Physics*, 58:2184–2199, 2019.

[137] E. Testa, M. Soeken, H. Riener, L. Amaru, and G. D. Micheli. A logic synthesis toolbox for reducing the multiplicative complexity in logic networks. In *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 568–573, 2020.

[138] Eleonora Testa, Mathias Soeken, Luca G. Amarù, and Giovanni De Micheli. Reducing the multiplicative complexity in logic networks for cryptography and security applications. In *Proceedings of the 56th Annual Design Automation Conference 2019, DAC 2019, Las Vegas, NV, USA*, page 74, 2019.

[139] Eleonora Testa, Mathias Soeken, Heinz Riener, Luca Amaru, and Giovanni De Micheli. A logic synthesis toolbox for reducing the multiplicative complexity in logic networks. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 568–573. IEEE, 2020.

[140] S. E. Venegas-Andraca and J. Ball. Processing images in entangled quantum systems. *Quantum Information Processing*, 9(1):1–11, 2010.

[141] S. E. Venegas-Andraca and S. Bose. Storing, processing, and retrieving an image using quantum mechanics. In *Quantum Information and Computation*, volume 5105, pages 137–147. SPIE, 2003.

[142] A. G. White, A. Gilchrist, G. J. Pryde, J. L. O'Brien, M. J. Bremner, and N. K. Langford. Measuring two-qubit gates. *Journal of the Optical Society of America B*, 24(2):172–183, 2007.

[143] R. Wille et al. Syrec: a hardware description language for the specification and synthesis of reversible circuits. *Integration, the VLSI Journal*, 53:39–53, 2016.

[144] Robert Wille, Stefan Hillmich, and Lukas Burgholzer. Efficient and correct compilation of quantum circuits. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2020.

[145] S. Yang. Logic synthesis and optimization benchmarks user guide version 3.0. User guide, Microelectronic Center, 1991.

[146] Mingsheng Ying and Yuan Feng. An algebraic language for distributed quantum computing. *IEEE Transactions on Computers*, 58(6):728–743, 2009.

[147] Reza Parvin Zadeh and Majid Haghparast. A new reversible/quantum ternary comparator. *Australian Journal of Basic and Applied Sciences*, 5(12):2348–2355, 2011.

[148] Christof Zalka. Grover's quantum searching algorithm is optimal. *Physical Review A*, 60(4):2746, 1999.

[149] Y. Zhang, K. Lu, Y. Gao, and M. Wang. Neqr: A novel enhanced quantum representation of digital images. *Quantum Information Processing*, 12:2833–2860, 2013.

[150] V. V. Zhirnov, R. K. Cavin, J. A. Hutchby, and G. I. Bourianoff. Limits to binary logic switch scaling–a gedanken model. *Proceedings of the IEEE*, 91(11):1934–1939, 2003.

[151] Zeljko Zilic and Katarzyna Radecka. Scaling and better approximating quantum fourier transform by higher radices. *IEEE Transactions on computers*, 56(2):202–207, 2007.