3-8-2012

Design Methodologies for Reversible Logic Based Barrel Shifters

Saurabh Kotiyal
University of South Florida, skotiyal@mail.usf.edu

Design Methodologies for Reversible Logic Based Barrel Shifters

by

Saurabh Kotiyal

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Engineering
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: Nagarajan Ranganathan, Ph.D.
Srinivas Katkoori, Ph.D.
Hao Zheng, Ph.D.

Date of Approval:
March 8, 2012

Keywords: Quantum Computing, Modified Fredkin Gate, Quantum Cost, Ancilla Inputs, Garbage Outputs

DEDICATION

To my parents, for all of their support and encouragement.

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Nagarajan Ranganathan for all of his mentoring and guidance and support. I would also like to thank Dr. Himanshu Thapliyal for all of his valuable advises and suggestions thought-out the years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The reversible logic has the promising applications in emerging computing paradigm such as quantum computing, quantum dot cellular automata, optical computing, etc. In reversible logic gates there is a unique one-to-one mapping between the inputs and outputs. To generate an useful gate function the reversible gates require some constant ancillary inputs called ancilla inputs. Also to maintain the reversibility of the circuits some additional unused outputs are required that are referred as the garbage outputs. The number of ancilla inputs, number of garbage outputs and quantum cost plays an important role in the evaluation of reversible circuits. Thus minimizing these parameters are important for designing an efficient reversible circuit. Barrel shifter is an integral component of many computing systems due to its useful property that it can shift and rotate multiple bits in a single cycle.

The main contribution of this thesis is a set of design methodologies for the reversible realization of reversible barrel shifters where the designs are based on the Fredkin gate and the Feynman gate. The Fredkin gate can implement the 2:1 MUX with minimum quantum cost, minimum number of ancilla inputs and minimum number of garbage outputs and the Feynman gate can be used so as to avoid the fanout, as fanout is not allowed in reversible logic. The design methodologies considered in this work targets 1.) Reversible logical right-shifter, 2.) Reversible universal right shifter that supports logical right shift, arithmetic right shift and the right rotate, 3.) Reversible bidirectional logical shifter, 4.) Reversible bidirectional arithmetic and logical shifter, 5) Reversible universal bidirectional shifter that supports bidirectional logical and arithmetic shift and rotate operations. The proposed design methodologies are evaluated in terms of the number of the garbage outputs, the number of ancilla inputs and the quantum cost. The detailed architecture and the design of a (8,3) reversible logical right-shifter and the (8,3) reversible universal right shifter are presented for illustration of the proposed methodologies.

CHAPTER 1

INTRODUCTION

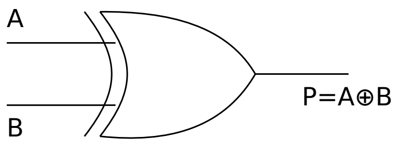
Reversible logic is a logic design style in which there is a one to one mapping between the input and the output vectors. According to [1], if a system is irreversible then erasing a bit causes kTln2 joules of heat energy where k is the Boltzmann's constant and T is the absolute temperature of the environment. This kTln2 joules of heat energy won't be dissipated if a computation is performed reversibly based on reversible logic circuits [2]. Reversible logic has extensive applications in emerging technologies such as quantum computing, quantum dot cellular automata, optical computing, etc [3, 4, 5, 6, 7, 8]. The major application of reversible logic lies in quantum computing. A quantum computer will be viewed as a quantum network (or a family of quantum networks) composed of quantum logic gates; each gate performing an elementary unitary operation on one, two or more two-state quantum systems called qubits. Quantum networks must be built from reversible logical components [9]. A barrel shifter can shift and rotate multiple bits in a single cycle and hence forms the integral component of many computing systems including general purpose processor and digital signal processors, wireless communication systems involving low-density parity-check (LDPC) decoders etc. [10, 11, 12, 13].

The input and output vector of an N-input and N-output reversible logic gate or NXN reversible logic gate can be represented as

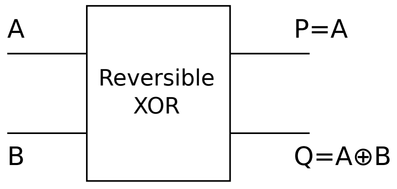
$$I_u = I_1, I_2, I_3, ..., I_N \tag{1.1}$$

$$O_u = O_1, O_2, O_3, ..., O_N \tag{1.2}$$

Here I_u and O_u represents the input and output vectors of a reversible logic gate. The conventional CMOS logic gates are irreversible in nature as the as the input vectors can not be created by the output vectors. Thus erasing a bit or loss of information causes kTln2 joules of heat energy [1]. However in reversible logic gates the their exists a unique one to one mapping between the input and output vectors. An irreversible XOR gate can be represented as shown in Fig. 1.1(a), where A,B and P are the inputs and outputs respectively. In the conventional XOR gate the inputs A,B are mapped to the output P as $P = A \oplus B$. Whereas in the reversible logic, a XOR gate can be represented as shown in Fig. 1.1(b), where A,B and P,Q are the input and output vectors respectively. Here the mapping between the inputs and outputs can be represented as $P = A$ and $Q = A \oplus B$ and holds the property of unique input and output vector mapping property of reversibility. The reconstruction of input vectors from the output vectors can be seen in table 1.1(b). The table 1.1(a) shows the input and output vectors for a conventional XOR gate. In the table 1.1(a) it can be seen that for the output $P = 0$ the input vectors are $AB = 01, 10$, whereas in the table 1.1(b) each output vector corresponds to a unique input vector.



(a) Conventional XOR gate



(b) Reversible XOR gate

Figure 1.1. Conventional and reversible XOR gates

Table 1.1. Truth table of conventional and reversible XOR gates

(a) Conventional XOR gate

A	B	$P = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

(b) Reversible XOR gate

A	B	$P = A$	$Q = A \oplus B$
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

1.1 Motivation

In this work, we propose design methodologies for the realization of reversible logic based barrel shifters. The proposed design methodologies of reversible barrel shifters are capable of shift or rotate the input data in both direction and can also be used for logical or arithmetic shifting. In reversible circuits, there are associated overheads in terms of ancilla inputs and the garbage outputs. An auxiliary constant input is called an ancilla input bit [14], while the garbage outputs refers to the outputs which do not perform any useful operations and are needed to maintain reversibility. The reversible logic circuit optimized in terms of number of garbage outputs, number of ancilla inputs and the quantum cost is considered as an efficient design. For example, quantum computers of many qubits are extremely difficult to realize thus researchers have focused their work towards minimizing the number of qubits in the quantum circuits [15, 16]. The technology to implement quantum computers is yet to be realized and hence, the implementation details are not known. In recent years, the fact that quantum computers will be based on reversible logic has been well established which forms the main motivation for this work. The reversible design methodologies of barrel shifters are based on the Fredkin gate and the Feynman gate. The Fredkin gate is used as it can implement the 2:1 MUX with minimum quantum cost, minimum number of ancilla inputs and minimum number of garbage outputs. The Feynman gate is used to avoid the fanout as fanout is not allowed in reversible logic.

1.2 Contribution of Thesis

The design methodologies of reversible barrel shifters that are considered in this work are

- Reversible logical right shifter
- Reversible universal right shifter
- Reversible bidirectional logical shifter

- Reversible bidirectional arithmetic and logical shifter
- Reversible universal bidirectional shifter

The design methodologies for different types of barrel shifter are proposed with the objective to optimize parameters such as the number of ancilla input bits, the number of garbage outputs and the quantum cost. In the existing literature, there is limited work on the design of reversible barrel shifters and the existing reversible barrel shifter designs are capable of a single operation i.e left rotation. In this work we propose design methodologies for (n,k) reversible barrel shifters that are capable of performing different types of shift/rotate operations. (i) The reversible logical right shifter supports the logical right shift operation; (ii) the reversible universal right shifter that supports logical right shift, arithmetic right shift and the right rotate; (iii) the reversible bidirectional logical shifter supports logical right shift and logical left shift operations; (iv) the reversible bidirectional arithmetic and logical shifter [17] supports logical right shift, arithmetic logical left shift and arithmetic left shift operations; (v) the reversible universal bidirectional shifter that supports bidirectional logical shift, arithmetic shift and rotate operations. The proposed design methodologies are evaluated in terms of the number of the garbage outputs, the number of ancilla inputs and the quantum cost. Also the impact analysis of varying the values of n and k for all proposed (n,k) reversible barrel shifters is also shown in the work.

1.3 Outline of Thesis

The organization of thesis is as follows: Chapter 2 provides some background on reversible logic with description of reversible logic gates used in the design of reversible barrel shifters. It also provides a brief review of related work and the basics of barrel shifters with analysis of various operations as performed by a barrel shifter. The Chapter 3 presents the design methodologies proposed in this work. Section 3.1 discusses the design methodology of reversible logical right shifter; Section 3.2 discusses the design methodology of reversible universal right shifter; Section 3.3 discusses the design methodology of reversible bidirectional logical shifter;Section 3.4 discuss the design of reversible bidirectional arithmetic and logical Shifter with an example of (8,3) reversible bidirectional arithmetic and logical shifter; Section 3.5 discusses the design methodology of reversible universal bidirectional shifter. In Chapter 4 , we discuss the performance evaluation of the proposed design methodologies in terms of the number of the garbage outputs, number of ancilla inputs and the quantum cost. Some conclusion are discussed in Chapter 5.

CHAPTER 2

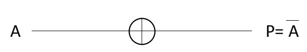
BACKGROUND AND RELATED WORK

2.1 Reversible Logic Gates

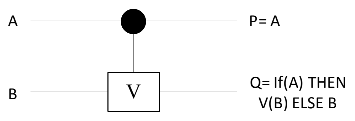
In the existing literature, there are several reversible gates such as the Feynman gate and the Fredkin gate. The number of 1x1 and 2x2 reversible gates needed to design a 3x3 reversible gate from 1x1 and 2x2 reversible gates is called the quantum cost of that gate. The quantum cost of all 1x1 and 2x2 reversible gates are considered as unity[18, 19, 20]. The 3x3 reversible gates are generally implemented using the 1x1 NOT gate and 2x2 reversible gates such as Controlled-V and Controlled-V+ (V is a square-root of NOT gate and V+ is its hermitian) and the Feynman gate also known as Controlled NOT gate (CNOT).

2.1.1 The NOT Gate

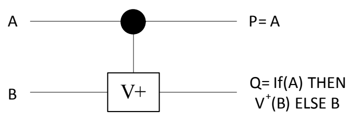
A NOT gate is 1x1 gate represented as shown in Fig. 2.1(a). Since it is a 1x1 gate, its quantum cost is unity.



(a) NOT gate



(b) Controlled-V gate



(c) Controlled-V^+ gate

Figure 2.1. NOT, Controlled-V and Controlled-V^+ gates

2.1.2 Controlled-V and Controlled-V^+ Gates

A controlled-V gate is shown in Fig. 2.1(b). In a controlled-V gate, when the control signal A=0 then the qubit B will pass through the controlled part unchanged, i.e., we will have Q=B. When the value of A=1 then the unitary operation $V = \frac{i+1}{2}\begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix}$ is applied to input B, i.e., Q=V(B). The controlled-V^+ gate is shown in Fig. 2.1(c). In the controlled-V^+ gate when the control signal A=0 then the qubit B will pass through the controlled part unchanged, i.e., we will have Q=B. When A=1 then the unitary operation $V^+ = V^{-1}$ is applied to the input B, i.e.,Q=V^+(B).

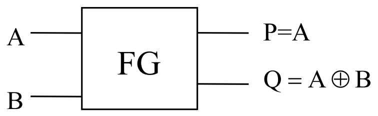
The V and V+ quantum gates have the following properties:

$$V \times V = NOT$$
$$V \times V^+ = V^+ \times V = I$$
$$V^+ \times V^+ = NOT$$

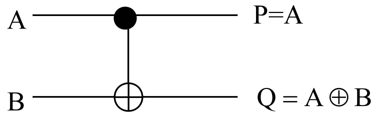
The property as shown above represents that when two V gates are in series they will behave as a NOT gate. Similarly two V^+ gate in series behaves as a NOT gate. A V gate in series with a V^+ gate and vice versa, is an identity. The more details of V and V^+ gate can be found in [9, 19].

2.1.3 Feynman Gate (CNOT Gate)

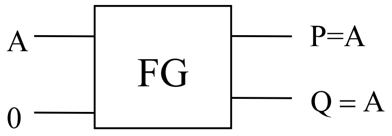
The Feynman gate (FG) or the controlled-NOT gate (CNOT) is a 2-inputs and 2-outputs reversible gate with the mapping (A, B) to (P=A, Q=A$\oplus$B). Here A is the controlling input and B is the controlled input; P, Q are the two outputs. Since the Feynman gate is a 2x2 reversible gate, it has a quantum cost of 1. Figure 2.2(a) and 2.2(b) shows the block diagram and the quantum representation of the Feynman gate. Fanout is not allowed in reversible logic. Feynman gate is helpful in this regard as it can be used for copying the signal thus avoiding the fanout problem as shown in Fig. 2.2(c).It can also be used for generating the complement of a given input signal as shown in Fig. 2.2(d).



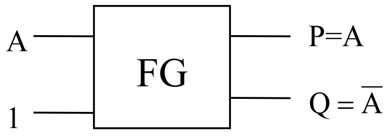
(a) CNOT Gate



(b) Quantum representation of the CNOT gate



(c) Feynman gate for avoiding the fanout

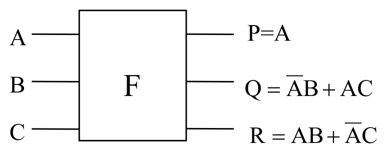


(d) Feynman gate for generating the complement of a signal

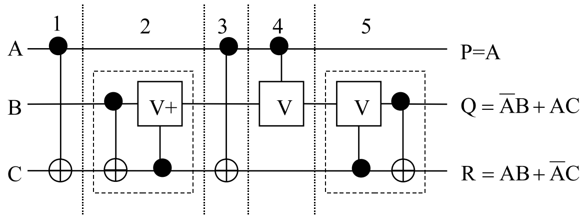
Figure 2.2. CNOT gate, its quantum implementation and its useful properties

2.1.4 Fredkin Gate

Fredkin gate is a 3x3 reversible logic gate with three inputs and three outputs. Figure 2.3(a) shows the block diagram of a Fredkin gate. The Fredkin gate maps (A, B, C) to (P=A, $Q = \bar{A}B + AC$, $R = AB + \bar{A}C$), where A, B, C are the inputs and P, Q, R are the outputs, respectively [21]. A Fredkin gate can work as 2:1 MUX, as it is able to swap its other two inputs depending on the value of its first input. Referring the Fig. 2.3(a), the first input A works as a controlling input while the inputs B and C work as controlled inputs. Thus when A=1 the inputs B and C will be swapped resulting in the value of the outputs as Q=C and R=B. If A=0 the outputs P and Q will be directly connected to inputs A and B. Figure 2.3(b) shows the quantum implementation of a Fredkin gate with a quantum cost of 5 [19]. In Fig. 2.3(b) each dotted rectangle is equivalent to a 2x2 Feynman gate and the quantum cost of each dotted rectangle is considered as 1 [18]. The same assumption is used for calculating the quantum cost of the Fredkin gate in [19]. Thus, the quantum cost of the Fredkin gate is 5 as it consists of 2 dotted rectangle, 1 Controlled-V gate and 2 CNOT gate. In this work, we have also followed the assumption by [18], and in our quantum cost calculations the quantum cost of the Fredkin gate is considered as 5.



(a) Fredkin Gate

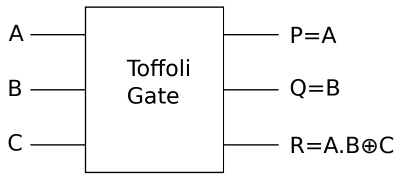


(b) Quantum representation of the Fredkin Gate

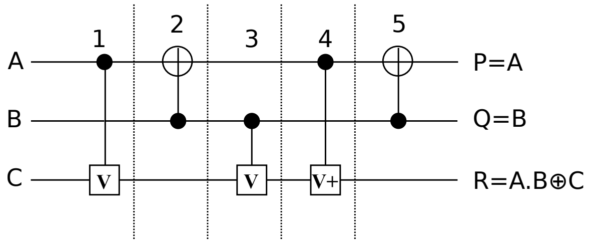
Figure 2.3. Fredkin Gate and its quantum implementation

2.1.5 Toffoli Gate

The Toffoli gate (TG) is a 3X3 reversible logic gate with three inputs and three outputs. The input to output mapping of a Toffoli gate can be represented as (A ,B ,C) to ($P = A$, $Q = B$, $R = A.B \oplus c$), where A,B,C are the inputs and P,Q,R are the outputs of a Toffoli gate. Figure 2.4(a) shows the block diagram of a Toffoli gate. A Toffoli gate has a quantum cost of 5 as it can implemented using 2V gates, $1V^+$ gate and 2 CNOT gates [22]. Figure 2.4(b) shows the quantum implementation of a Toffoli gate.



(a) Toffoli Gate



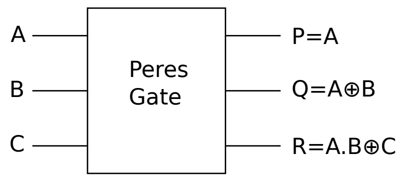
(b) Quantum representation of the Toffoli Gate

Figure 2.4. Toffoli Gate and its quantum implementation

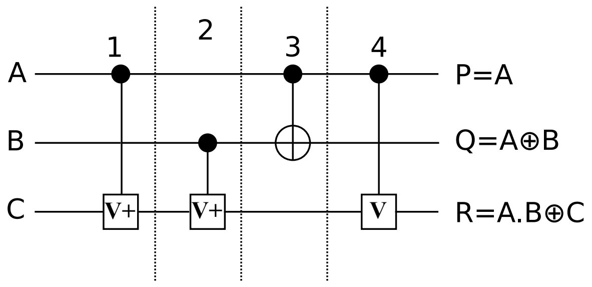
2.1.6 Peres Gate

The Peres gate is 3X3 reversible gate with a mapping between the inputs A,B,C and the outputs P,Q,R as (A, B, C) to ($P = A$, $Q = A \oplus b$, $R = A.B \oplus C$) [23]. A block diagram of Peres gate is shown in Fig. 2.5(a) and Fig. 2.5(b) shows the quantum representation of

Peres gate. The quantum implementation of Peres gate requires $2V^+$ gates, 1V gate and 1 CNOT gate. Thus the quantum cost of a Peres gate can be calculated as 4.



(a) Peres Gate



(b) Quantum representation of the Peres Gate

Figure 2.5. Peres Gate and its quantum implementation

2.2 Basics of Barrel Shifter

A barrel shifter is a n-inputs and n-outputs combinational logic circuit in which k select lines controls the bit shift operation. Barrel shifter can be unidirectional allowing data to be shifted only to left (or right), or bi-directional which provides data to be rotated or shifted in both the directions. A barrel shifter having n inputs and k select lines is called (n,k) barrel shifter. Among the different designs of barrel shifter, the logarithmic barrel shifter is most widely used because of its simple design, less area and the elimination of the decoder circuitry. The conventional irreversible design of a logarithmic barrel shifter is shown in Fig. 2.6. A n-bit Logarithmic Barrel Shifter contains $log_2(n)$ stage where the i_{th} stage either shifts over 2^i bits or leaves the data unchanged. Each stage of a logarithmic barrel shifter is controlled by a control bit. If the control bit is set to one then the input data will be shifted in the associated stage else it remains unchanged. The proposed work presents the design methodologies for reversible barrel shifter that can perform six operations: logical right shift, arithmetic right shift, right rotate, logical left shift, arithmetic left shift and left rotate. For illustration, all of these operation are as shown in Table 2.1 for an 8 bit logarithmic barrel shifter where the 8 bit input data is denoted as $i_7, i_6, i_5, i_4, i_3, i_2, i_1, i_0$

here i_7 is the sign bit, and the shift or rotate operation is performed by 3 bits, and X denotes the shifted result.

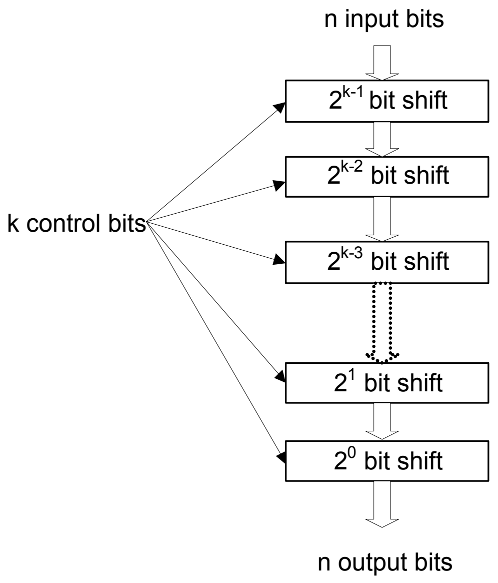


Figure 2.6. Structure of (n, k) logarithmic barrel shifter

Table 2.1. Operations on an 8 bit barrel shifter with 3 bit shift value

Operation Performed	X($\mathit{Final\ Output}^*$)
Logical right shift	$000i_7i_6i_5i_4i_3$
Arithmetic right shift	$i_7i_7i_7i_7i_6i_5i_4i_3$
Right rotation	$i_2i_1i_0i_7i_6i_5i_4i_3$
Logical left shift	$i_4i_3i_2i_1i_0000$
Arithmetic left shift	$i_7i_3i_2i_1i_0000$
Left rotation	$i_4i_3i_2i_1i_0i_7i_6i_5$
*The input data is $i_7i_6i_5i_4i_3i_2i_1i_0$, here i_7 is the sign bit	

All operation that can be performed by a logarithmic barrel shifter as shown in Table 2.1 are as follows:

2.2.1 Logical Right Shift

As illustrated in Table 2.1 a 3-bit logical right shift operation right shifts the input data by 3 bits and sets the leftmost 3-bits to zero thus the output will be $000i_7i_6i_5i_4i_3$.

2.2.2 Arithmetic Right Shift

A 3-bit arithmetic right shift operation right shifts the input data bit by 3-bits and sets the leftmost 3-bits to the sign bit (i_7) thus the output will be $i_7i_7i_7i_7i_6i_5i_4i_3$.

2.2.3 Right Rotation

A 3-bit right rotation operation performs a right shift operation on input data by 3-bits. Further, the leftmost 3-bits are set to the rightmost 3-bits of the original input data. Thus, as illustrated in the Table 2.1 the final output after the right rotation operation will be $i_2i_1i_0i_7i_6i_5i_4i_3$ as the input data $i_7i_6i_5i_4i_3i_2i_1i_0$ is shifted 3 times to the right and the 3 leftmost bits ($i_7i_6i_5$) are set to the 3 rightmost bits ($i_2i_1i_0$) of the original data.

2.2.4 Logical Left Shift

A 3-bit logical left shift operation left shifts the input data by 3-bits and sets the rightmost 3-bits to zero thus the final output will be $i_4i_3i_2i_1i_0000$.

2.2.5 Arithmetic Left Shift

In the arithmetic left shift operation the sign bit of the input data remains intact and the remaining bits are logically left shifted by 3-bits. Thus as illustrated in the Table 2.1 the output will be $i_7i_3i_2i_1i_0000$ as the input data $i_7i_6i_5i_4i_3i_2i_1i_0$ is logically left shifted by 3-bits and the sign bit (i_7) remains intact.

2.2.6 Left Rotation

A 3-bit left rotation operation performs a left shift operation on input data by 3-bits. Further the rightmost 3-bits are set to the leftmost 3-bits of the original input data. Thus as illustrated in the Table 2.1 the final output after the left rotation operation will be $i_4i_3i_2i_1i_0i_7i_6i_5$ as the input data $i_7i_6i_5i_4i_3i_2i_1i_0$ is shifted 3 times to the left and the 3 rightmost bits ($i_2i_1i_0$) are set to the 3 leftmost bits ($i_7i_6i_5$) of the original data.

2.3 Related Work

The research on reversible logic is being pursued towards both design and synthesis. In the synthesis of reversible logic circuits there has been several interesting attempts in the literature such as the work in [24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34]. The researchers have addressed the optimization of reversible logic circuits from the perspective of quantum cost and the number of garbage outputs. Recently, in [35, 36, 37, 38, 39, 40, 41, 42] interesting contributions are made toward deriving exact minimal elementary quantum gate realization of reversible combinational circuits. The design of reversible sequential circuits are also addressed in literature in which various latches, flip-flops, etc. are designed [43, 44, 45, 46, 47]. Arithmetic units such as adders, subtracters, multipliers form the essential component of a computing system [48]. Researchers have addressed the design of reversible adders and multipliers in binary as well as ternary logic such as in [49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61]. In [62] researchers have designed the quantum ripple carry adder having no input carry with one ancilla input bit. In [16, 63], the researchers have investigated new designs of the quantum ripple carry adder with no ancilla input bit and improved delay. In [64], the measurement based deign of carry look-ahead adder is presented while in [65] the concept of arithmetic on a distributed-memory quantum multicomputer is introduced. A comprehensive survey of quantum arithmetic circuits can be found in [15]. The design of reversible barrel shifters have also been attempted. The researchers have attempted the design of reversible barrel shifters but these design are only limited to design a reversible left rotator [66, 67, 68]. The designs of reversible left rotators in the existing literature is evaluated in terms of number of reversible gates used, quantum cost, garbage outputs and delay [67]. Thus to the best of our knowledge researchers have not yet addressed the design methodologies for reversible barrel shifter capable of performing all types of shift operations (arithmetic left/right shift, bidirectional shift, bidirectional rotate, logical left/right shift). In this work we present design methodologies of reversible shifters such as the reversible logical right-shifter, the reversible universal right shifter that supports logical right shift, arithmetic right shift and the right rotate; the reversible bidirectional logical shifter and the reversible universal bidirectional shifter that supports bidirectional logical and arithmetic shift and rotate operations.

CHAPTER 3

PROPOSED DESIGN METHODOLOGIES

The design methodologies of reversible shifters that are considered in this work are the reversible logical right shifter, the reversible universal right shifter that supports logical right shift, arithmetic right shift and the right rotate; the reversible bidirectional logical shifter; the reversible bidirectional arithmetic and logical shifter and the reversible universal bidirectional shifter that supports bidirectional logical and arithmetic shift and rotate operations.

3.1 Reversible Logical Right Shifter

A reversible logical right shifter is capable of logical right shifting the input data as explained in Table 2.1. For a (n,k) reversible logical right shifter the input data is represented as $i_{n-1}, i_{n-2}, i_{n-3}, ..., i_2, i_1, i_0$. As a logarithmic shifter can be designed using $log_2(n)$ stages, the (n,k) reversible logical right shifter can also be designed using $log_2(n)$ stages. *The $log_2(n)$ number of stages of a (n,k) reversible logical right shifter is controlled by control signals S_{k-1},S_{k-2},....,S_1,S_0. The S_{k-1} will work as the control signal for 1st stage, S_{k-2} will work as the control signal for 2nd stage and so on. Thus for m_{th} stage the control signal will be S_{k-m} where m= 1 to k.* Each stage of (n,k) reversible logical right shifter is responsible for shifting input data by 2^{k-1} to 2^0 bits. Since Fredkin gate has the property that two of its outputs work as an 2:1 MUX, thus a chain of Fredkin gates can works as the chain of 2:1 MUXes. This helps in implementing the stages of a (n,k) reversible logical right shifter using the Fredkin gates. The Feynman gates can be used for copying input data bits, thus avoiding the fanout. The final output after the last stage can be represented as $O_{n-1}, O_{n-2}, O_{n-3}, ..., O_2, O_1, O_0$. All stages of a (n,k) reversible logical right shifter can be designed as illustrated.

- **Stage 1:** The input bits for a (n,k) reversible logical right shifter can be represented as $i_{n-1}, i_{n-2}, i_{n-3}, ..., i_2, i_1, i_0$ and the control signal can be written as $S_{k-1}, S_{k-2}, S_{k-3}, ..$ $.., S_2, S_1, S_0$, where n represents the number of bits and k represents the stages in the shifter. To design the k stages of (n,k) reversible logical right shifter, it will require k chains of n Fredkin gates. Thus the inputs and outputs for each Fredkin gate in a stage j can be rewritten as $A(i,j)$,$B(i,j)$,$C(i,j)$ and $P(i,j)$,$Q(i,j)$,$R(i,j)$, respectively. Here i represents the position of the Fredkin gate in a stage and j represents the jth stage and $n-1 \geq i \geq 0$ and $k-1 \geq j \geq 0$.

  For the first stage the inputs to the Fredkin gates can be written as follows:

$$A(i,k-1) = S_{k-1} \quad \text{for } i = (n-1) \text{ to } 0$$
$$B(i,k-1) = i_i \quad \text{for } i = (n-1) \text{ to } 0$$

$$C(i,k-1) = \begin{cases} 0 & \text{for } i = (n-1) \text{ to } ((n-1) \\ & -2^{k-1}+1) \\ i_m & \text{for } i = ((n-1)-2^{k-1}) \text{ to } 0 \\ & \text{and } m = (n-1) \text{ to } 2^{k-1} \end{cases}$$

  As illustrated above, the first stage of a (n,k) reversible logical right shifter is controlled by the control signal S_{k-1}, thus the controlling input for Fredkin gates in first stage is set to $A(i,k-1) = S_{k-1}$, for all $i = (n-1) \quad to \quad 0$. There can be two possible values of S_{k-1} which are 0 and 1. Considering a Fredkin gate having inputs A, B,C and the outputs as P,Q,R, when the value of A=1 we will have Q=C and R=B; when the value of A=0 we will have Q=B and R=C. Thus Fredkin gate can swap the value of B and C depending on the value of input A. Since S_{k-1} works as the input A of the Fredkin gates, we will have the following possible outputs at the 1st stage.

When S_{k-1}=1, we will have following values at the outputs.

$$P(i,k-1) = S_{k-1} \quad \text{for } i = (n-1) \text{ to } 0$$
$$R(i,k-1) = i_i \qquad \text{for } i = (n-1) \text{ to } 0$$

$$Q(i,k-1) = \begin{cases} 0 & \text{for } i = (n-1) \text{ to } ((n-1) \\ & -2^{k-1}+1) \\ i_m & \text{for } i = ((n-1)-2^{k-1}) \text{ to } 0 \\ & \text{and } m = (n-1) \text{ to } 2^{k-1} \end{cases}$$

When S_{k-1}=0, we will have following values at the outputs.

$$P(i,k-1) = S_{k-1} \quad \text{for } i = (n-1) \text{ to } 0$$
$$Q(i,k-1) = i_i \qquad \text{for } i = (n-1) \text{ to } 0$$

$$R(i,k-1) = \begin{cases} 0 & \text{for } i = (n-1) \text{ to } ((n-1) \\ & -2^{k-1}+1) \\ i_m & \text{for } i = ((n-1)-2^{k-1}) \text{ to } 0 \\ & \text{and } m = (n-1) \text{ to } 2^{k-1} \end{cases}$$

Thus, when the value of S_{k-1}=1 it will logical right shift the inputs data $i_{n-1}, i_{n-2}, i_{n-3}$,...., i_2, i_1, i_0 by 2^{k-1} bits. When the value of S_{k-1}=0 there will not be any shift operation. The copying of the input data is achieved by utilizing the Feynman gates.

- **Stage 2:** The second stage of (n,k) reversible logical right shifter uses inputs from the first stage. From the outputs $Q(i,k-1)$ and $R(i,k-1)$ for $i = (n-1)$ to 0 of the Fredkin gates in the first stage only the outputs $Q(n,k-1)$ will be the useful outputs while the outputs $R(n,k-1)$ will work as the garbage output. The inputs at the chain of n Fredkin gates for the second stage can be provided by following the same design methodology as used for the first stage and is as follows:

$$
\begin{array}{ll}
A(i,k-2) = S_{k-2} & \text{for } i = (n-1) \text{ to } 0 \\
B(i,k-2) = Q(i,k-1) & \text{for } i = (n-1) \text{ to } 0
\end{array}
$$

$$
C(i,k-2) = \begin{cases} 0 & \text{for } i = (n-1) \text{ to} \\ & ((n-1) - 2^{k-2} + 1) \\ Q(m,k-1) & \text{for } i = ((n-1) \\ & -2^{k-2}) \text{ to } 0 \\ & \text{and } m = (n-1) \text{ to} \\ & 2^{k-2} \end{cases}
$$

Since S_{k-2} works as the input A of the Fredkin gates, we will have the following possible outputs at the 2nd stage.

When S_{k-2}=1, we will have following values at the outputs.

$$
\begin{array}{ll}
P(i,k-2) = S_{k-2} & \text{for } i = (n-1) \text{ to } 0 \\
R(i,k-2) = Q(i,k-1) & \text{for } i = (n-1) \text{ to } 0
\end{array}
$$

$$
Q(i,k-2) = \begin{cases} 0 & \text{for } i = (n-1) \text{ to} \\ & ((n-1) - 2^{k-2} + 1) \\ Q(m,k-1) & \text{for } i = ((n-1) \\ & -2^{k-2})\ to\ 0 \\ & \text{and } m = (n-1) \text{ to} \\ & 2^{k-2} \end{cases}
$$

When S_{k-2}=0, we will have following values at the outputs.

$$
\begin{array}{ll}
P(i,k-2) = S_{k-2} & \text{for } i = (n-1) \text{ to } 0 \\
Q(i,k-2) = Q(i,k-1) & \text{for } i = (n-1) \text{ to } 0
\end{array}
$$

$$R(i,k-2) = \begin{cases} 0 & \text{for } i = (n-1) \text{ to} \\ & ((n-1) - 2^{k-2} + 1) \\ Q(m,k-1) & \text{for } i = ((n-1) \\ & -2^{k-2})\ to\ 0 \\ & \text{and } m = (n-1) \text{ to} \\ & 2^{k-2} \end{cases}$$

Thus, when the value of S_{k-2}=1 it will logical right shifts the inputs data Q(i,k-1), for i=n-1 to 0, by 2^{k-2} bits. Here, Q(i,k-1) represents the useful output generated at the stage 1. When the value of S_{k-2}=0 there will not be any shift operation. The copying of the input data is achieved by utilizing the Feynman gates.

- **Stage 3 to Stage k** The same design strategy as illustrated in previous stages can also be used to design these stages.

Thus by the end of kth stage the input data $i_{n-1}, i_{n-2}, i_{n-3}, ..., i_2, i_1, i_0$ can be logical right shifted from 0 to k-1 bits depending on the value of S_{k-m} where $m = k-1$ to 1. Thus in the design methodology for a (n,k) reversible logical right shifter each stage requires $n - 2^{k-m}$ number of Feynman gates, here m represents the mth stage. Thus total number of Feynman gates required to design a (n,k) reversible logical right shifter (rlrs) can be written as $FE_{rlrs} = \sum_{m=1}^{k}(n - 2^{(k-m)})$. Each stage consists of chain of n Fredkin gates working as a 2:1 MUX. As there are k stages, the total number of Fredkin gates required to design a (n,k) reversible logical right shifter(rlrs) can be represented as $FR_{rlrs} = n * k$. An example of the proposed design methodology is illustrated for the design of a (8,3) reversible logical right shifter in Fig. 3.1. A (8,3) reversible logical right shifter has input as $i_7, i_6, i_5, i_4, i_3, i_2, i_1, i_0$ with the respective output as $O_7, O_6, O_5, O_4, O_3, O_2, O_1, O_0$. The design has $log_2(8) = 3$ stages, thus has the control signal S_2, S_1, S_0. Each stage of a (8,3) reversible logical right shifter can be designed using chain of 8 Fredkin gates as explained in the design methodology. Feynman gates are used in the design to avoid the fanout. The total number of Feynman gates required to design a (8,3) reversible logical right shifter (rlrs) can be written as $FE_{rlrs} = \sum_{m=1}^{3}(8 - 2^{(3-m)})$=(4+6+7)=17. Each stage consists of chain of 8 Fredkin gates working as a 2:1 MUX. As there are 3 stages, the total number of Fredkin gates required to design a (8,3) reversible logical right shifter(rlrs) can be represented as $FR_{rlrs} = 8 * 3 = 24$.

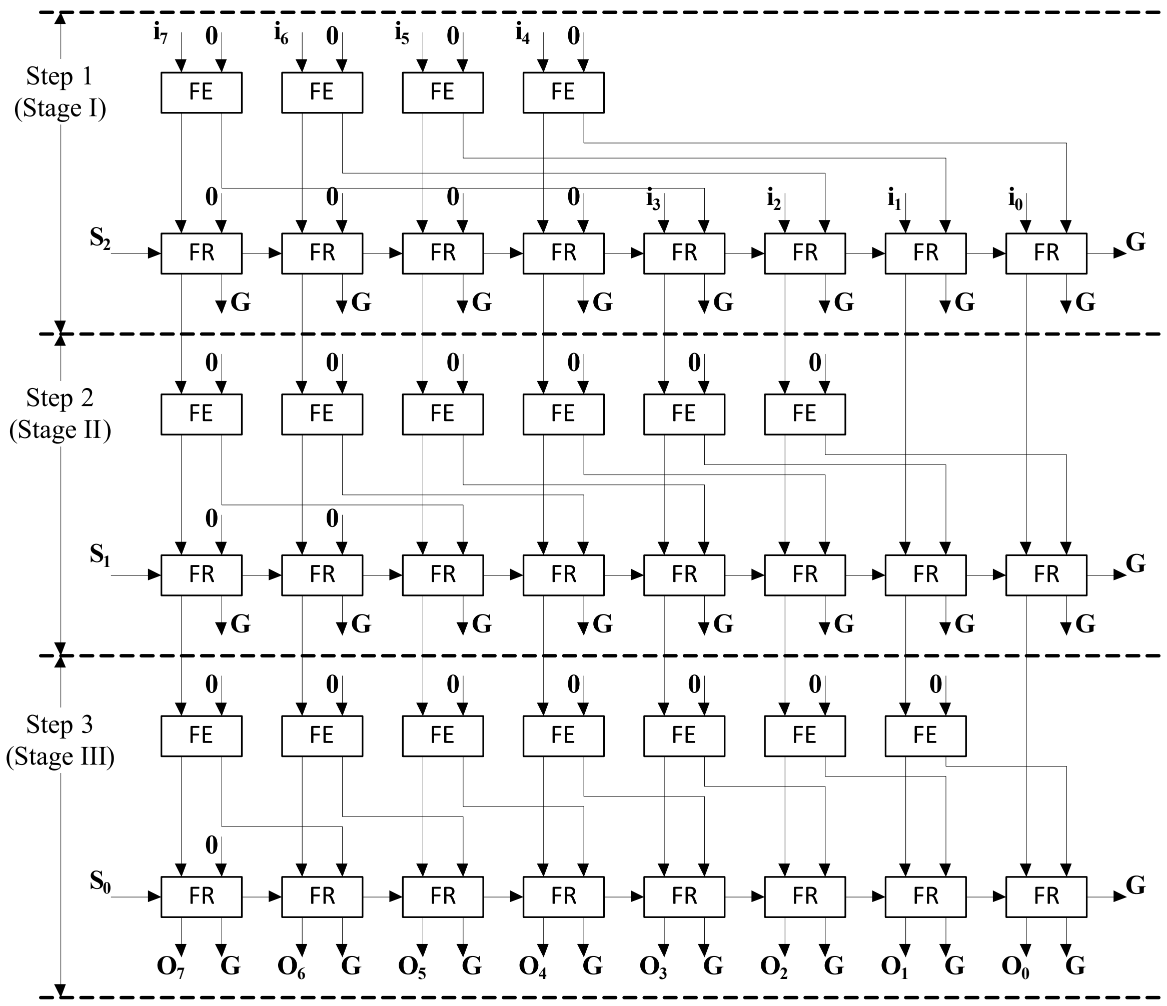


Figure 3.1. A (8,3) reversible logical right shifter (FE: Feynman gate, FR: Fredkin gate, G: Garbage output)

3.2 Reversible Universal Right Shifter

A reversible universal right shifter is capable of logical right shifting, arithmetic right shifting and right rotating the input bits. The logical right shift, arithmetic right shift and the right rotation operations are explained in Table 2.1.

A (n,k) reversible universal right shifter can be designed from (n,k) reversible logical right shifter by using additional Fredkin gates for arithmetic shift and rotate operations. For a (n,k) reversible universal right shifter the input data is represented as $i_{n-1}, i_{n-2}, i_{n-3}, ..., i_2, i_1, i_0$. There are two more control signals *sra*,*rotate* in addition to S_{k-1},S_{k-2},....,S_1,S_0 select signals. The control signal *sra* is responsible for arithmetic right shift operation, while *rotate* is responsible for right rotate operation or logical right shift operation. All the shift or rotation operation performed by (n,k) reversible universal right shifter are described in the Table 3.1 which are controlled by the control signals *sra*,*rotate*.

Table 3.1. Operations on (n,k) reversible universal right shifter

sra	rotate	Operation
1	0	Reversible arithmetic right shift
0	1	Reversible right rotation
0	0	Reversible logical right shift

To facilitate the arithmetic right shift one additional Fredkin gate is used with the design described above for (n,k) reversible logical right shifter. The input and output sequences for this additional Fredkin gate can be written as A_{as}, B_{as}, C_{as} and P_{as}, Q_{as}, R_{as} respectively. The controlling input A for this Fredkin gate is set to $A_{as} = sra$, while the inputs B and C are set to $B_{as} = i_{n-1}$ and $C_{as} = 0$. If the value of controlling input *sra* is set to 1 then (n,k) universal right shifter is responsible for performing arithmetic right shift operation. If the value of control signal *sra* is set to $sra = 0$ then it is responsible for either logical right shift or right rotate operation depending on the value of secondary control signal *rotate*.

A (n,k) reversible universal right shifter can be designed as : The first stage of (n,k) universal right shifter is responsible for shifting or rotating the input data by 2^{k-1}-bits. The first stage requires 2^{k-1} number of additional Fredkin gates in addition with the chain of n Fredkin gates. As shown in Fig.2.3(b), the inputs and the outputs of a Fredkin gate are represented as A,B,C and P,Q,R, respectively. Following that convention, the inputs and outputs for these additional Fredkin gates are represented as $A_a(i,j), B_a(i,j), C_a(i,j)$ and $P_a(i,j), Q_a(i,j), R_a(i,j)$, respectively. Here i represents the position of the Fredkin gate in a stage and j represents the j_{th} stage and $2^{k-1} \geq i \geq 0$ and $k-1 \geq j \geq 0$. The inputs and the select signals will be passed to these additional Fredkin gates as follows.

$$\begin{aligned}
A_a(i,k-1) &= rotate && \text{for } i = (2^{k-1}-1) \text{ to } 0 \\
B_a(i,k-1) &= Q_{as} && \text{for } i = (2^{k-1}-1) \text{ to } 0 \\
C_a(i,k-1) &= i_m && \text{for } i = (2^{k-1}-1) \text{ to } 0 \\
& && \text{and } m = (2^{k-1}-1) \text{ to } 0
\end{aligned}$$

In the first stage the additional Fredkin gates are controlled by the control signal $rotate$ thus the controlling input $A_a(i,k-1)$ is set as $A_a(i,k-1) = rotate$ for all $i = (2^{k-1}-1) \quad to \quad 0$. The assigning of $rotate$ to $A_a(i,k-1)$ facilitates logical right shift or right rotation operation. The input sequences for the second input of the additional Fredkin gates are set to $B_a(i,k-1) = Q_{as}$ for $i = (n-1) \quad to \quad 0$. This facilitates the arithmetic right shift operation by 2^{k-1} bits because when the control signals $sra = 1$ and $rotate = 0$ we will have $Q(i,k-1) = B_a(i,k-1)$. The input sequences for the third input of the additional Fredkin gates are set to $C_a(i,k-1) = i_m$ for $i = (2^{k-1}-1) \quad to \quad 0$ and $m = (2^{k-1}-1) \quad to \quad 0$. This facilitates the right rotate operation by 2^{k-1} bits because when the control signals $sra = 0$ and $rotate = 1$ we will have $Q(i,k-1) = C_a(i,k-1)$ and the logical right shift operation when $sra = 0$ and $rotate = 0$.

From the outputs $Q_a(i,k-1)$ and $R_a(i,k-1)$ for $i = 0$ to $(n-1)$ of the additional Fredkin gates only the outputs $Q_a(i,k-1)$ will be the useful outputs while the outputs $R_a(i,k-1)$ will work as the garbage output. Along the useful outputs $Q_a(i,k-1)$ of the additional Fredkin gate, the original inputs $i_{n-1}, i_{n-2}, i_{n-3}, ..., i_2, i_1, i_0$ will work as the inputs to the chain of n Fredkin gate as used in the design methodology for (n,k) reversible logical right shifter(please refer the stage 1 of the reversible logical shifter). These original inputs and $Q_a(i, k-1)$ outputs of the additional Fredkin gates are assigned to the chain of the n Fredkin gates as follows.

$$A(i, k-1) = S_{k-1} \quad \text{for } i = (n-1) \text{ to } 0$$
$$B(i, k-1) = i_i \qquad \text{for } i = (n-1) \text{ to } 0$$

$$C(i, k-1) = \begin{cases} Q_a(m, k-1) & \text{for } i = (n-1) \text{ to} \\ & ((n-1) - 2^{k-1} + 1) \\ & \text{and } m = (2^{k-1} - 1) \text{ to } 0 \\ i_m & \text{for } i = ((n-1) \\ & -2^{k-1})\ to\ 0 \\ & \text{and } m = (n-1) \text{ to } 2^{k-1} \end{cases}$$

The controlling input of the chain of n Fredkin gates in the 1st stage is set as $A(i, k-1) = S_{k-1}$ for $i = (n-1) \quad to \quad 0$ that facilitates either shift or no shift operation depending on the value of select signal S_{k-1}. Except these changes in the input mapping of the n Fredkin gates in the 1st stage, the working of the stage 1 is similar to the working of the stage 1 of the reversible logical right shifter, and thus is not explained separately. Similar to the design methodology of the reversible logical right shifter, the fanout is avoided by the use of the Feynman gate.

Similarly one can design the other stages of the reversible universal right shifter. We can observed that in each stage we will need 2^{k-m} for $m = 0 \quad to \quad (k-1)$ additional Fredkin gates in addition to the chain of n Fredkin gates. A (n,k) reversible universal right shifter has k stages, where each stage requires n number of Feynman gates to copy the input data. As illustrated above the output data of the single Fredkin gate used to perform arithmetic right shift requires $2^k - 2$ number of Feynman gates to copy its useful output. Further, a Feynman gate is needed to copy the sign bit. Hence, the total number of Feynman gates needed to design a (n,k) reversible universal right shifter(rurs) are $FE_{rurs} = n * k + (2^k - 2) + 1 = n * k + 2^k - 1$. To sum up the number of the Fredkin gates needed in the design, one can see that each stage of a (n,k) reversible universal right shifter requires chain of n Fredkin gates, and 2^k number of additional Fredkin gates to perform the rotate operation. Further as explained in the design methodology, one Fredkin gate is needed in the design to initiate the arithmetic right shift operation. Thus the total number of Fredkin gates that are needed to design a (n,k) reversible universal right shifter (rurs) can be written as $FR_{rurs} = n * k + \sum_{m=0}^{k-1} 2^m + 1$.

An example of the proposed design methodology is illustrated in Fig. 3.2 for the design of a (8,3) reversible universal right shifter. Since in a (8,3) reversible universal right shifter $n = 8$ thus it can be designed using $log_2(8) = 3$ stages. In the Fig. 3.2 the inputs and the outputs are $i_7, i_6, i_5, i_4, i_3, i_2, i_1, i_0$ and $O_7, O_6, O_5, O_4, O_3, O_2, O_1, O_0$ respectively. The proposed (8,3) reversible universal right shifter is controlled by two additional control signals: (1) sra for arithmetic or logical right shift, and (2) $rotate$ for either shift or rotate operation. The stages of the proposed (8,3) reversible universal right shifter are controlled by the select signals S_2,S_1,S_0. To better explain the working of the proposed (8,3) reversible universal right shifter an example of right shifting(arithmetic or rotate or logical) the input data by 6 bits is illustrated in the Table 3.2. In the (8,3) reversible universal right shifter as illustrated in Table 3.2 the select signals will have the values $S_2 = 1, S_1 = 1, S_0 = 0$, while the arithmetic right shift, right rotation, logical right shift operations are performed by changing the values of the control signals sra and $rotate$.

Table 3.2. Operations on (8,3) reversible universal right shifter

Stage/ Operation	Arithmetic Right Shift sra=1 rotate=0	Right Rotation sra=0 rotate=1	Logical Right Shift sra=0 rotate=0
Stage I ($S_2 = 1$)	$i_7, i_7, i_7, i_7, i_7, i_6, i_5, i_4$	$i_3, i_2, i_1, i_0, i_7, i_6, i_5, i_4$	$0, 0, 0, 0, i_7, i_6, i_5, i_4$
Stage II ($S_1 = 1$)	$i_7, i_7, i_7, i_7, i_7, i_7, i_7, i_6$	$i_5, i_4, i_3, i_2, i_1, i_0, i_7, i_6$	$0, 0, 0, 0, 0, 0, i_7, i_6$
Stage III ($S_0 = 0$)	$i_7, i_7, i_7, i_7, i_7, i_7, i_7, i_6$	$i_5, i_4, i_3, i_2, i_1, i_0, i_7, i_6$	$0, 0, 0, 0, 0, 0, i_7, i_6$

The working of a (8,3) reversible universal right shifter as shown in Fig. 3.2 can be understood by following the steps mentioned in Table 3.2. The details of which are also discussed in the following depending on the values of the control signals *sra* and *rotate*.

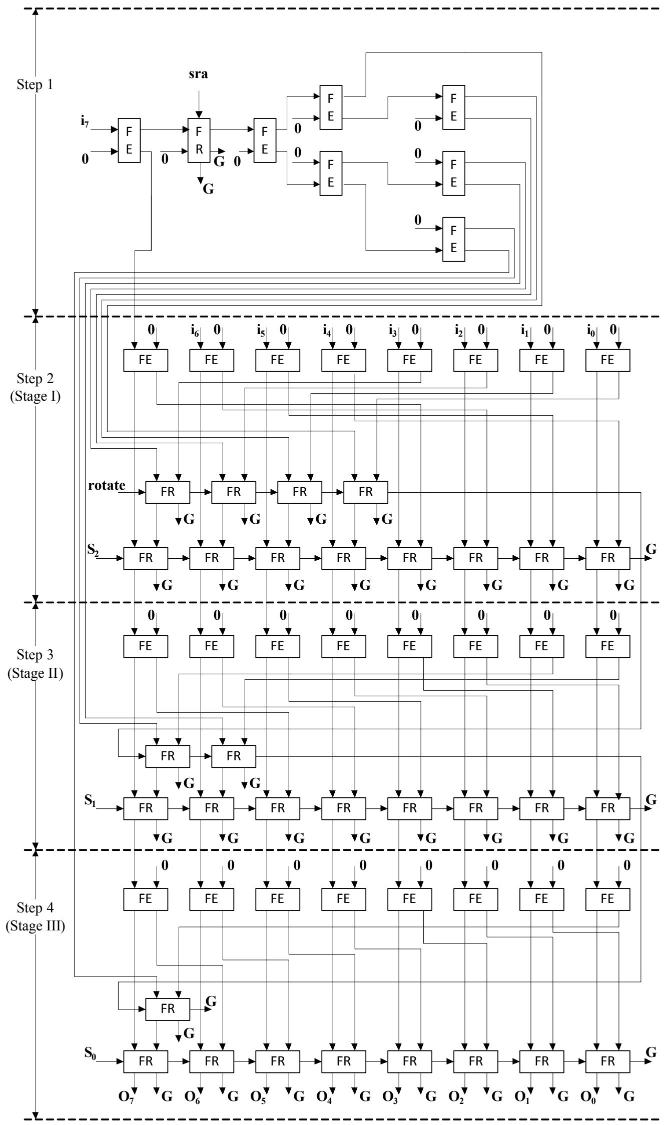


Figure 3.2. A (8,3) reversible universal right shifter (FE: Feynman gate, FR: Fredkin gate, G: Garbage output)

3.2.1 Arithmetic Right Shift

For $sra = 1$ and $rotate = 0$, the (8,3) reversible universal right shifter works as a (8,3) reversible arithmetic right shifter.

- Stage I: The first stage of a (8,3) reversible universal right shifter arithmetic right shifts the input data by $2^{3-1} = 4$ bits as the value of select signal S_2 is 1.Thus for the inputs as $i_7, i_6, i_5, i_4, i_3, i_2, i_1, i_0$ the first stage generates the outputs as $i_7, i_7, i_7, i_7, i_7, i_6, i_5, i_4$.

- Stage II: The outputs generated at the first stage is used as the inputs for the second stage. For the value of select signal as $S_1 = 1$ the second stage of a (8,3) reversible universal right shifter does arithmetic right shift on the input data provided by first stage by $2^{3-2} = 2$ bits. Hence the outputs generated by the second stage of (8,3) reversible universal right shifter are $i_7, i_7, i_7, i_7, i_7, i_7, i_7, i_6$.

- Stage II: The stage III which is the final stage has the value of select signal as $S_0 = 0$ and the outputs generated at the stage II are used as the inputs for this stage. Since in this stage the value of the select signal S_0 is 0 thus there will not be any shift operation in the stage. The final output at this stage are $i_7, i_7, i_7, i_7, i_7, i_7, i_7, i_6$ which is the arithmetic right shift by 6 bits since $S_2S_1S_0 = 110(decimal6)$.

3.2.2 Right Rotation

The right rotate operation depends on the value of control signal $rotate$.If the value of control signals are $sra = 0$ and $rotate = 1$, the (8,3) reversible universal right shifter operates as (8,3) reversible right rotator which can be explained as follows.

- Stage I: The first stage of (8,3)reversible universal right shifter right rotates the input data $i_7, i_6, i_5, i_4, i_3, i_2, i_1, i_0$ by $2^{3-1} = 4$ bits as the value of select signal S_2 is 1. This stage will generate the outputs as $i_3, i_2, i_1, i_0, i_7, i_6, i_5, i_4$.

- Stage II: In this stage the output of the first stage is right rotated by $2^{3-2} = 2$ bits as the value of the select signal S_1 is 1. The outputs generated at this stage are $i_5, i_4, i_3, i_2, i_1, i_0, i_7, i_6$.

- Stage III: The stage does not performs any operation and pass the output of the second stage as such as the value of the select signal is $S_0 = 0$. The final output at this stage are $i_5, i_4, i_3, i_2, i_1, i_0, i_7, i_6$ which is right rotation by 6 bits since $S_2S_1S_0 = 110(decimal6)$.

3.2.3 Logical Right Shift

For the logical right shift operation the value of the control signals will be $sra = 0$ and $rotate = 0$.

- Stage I: In this stage the input data $i_7, i_6, i_5, i_4, i_3, i_2, i_1, i_0$ is logically right shifted by $2^{3-1} = 4$ bits as the value of select signal S_2 is set as 1. Thus the outputs generated at this stage are $0, 0, 0, 0, i_7, i_6, i_5, i_4$.

- Stage II: At this stage the value of the select signal $S_1 = 1$ thus this stage logically right shifts the outputs of the first stage by $2^{3-2} = 2$ bits. The outputs generated at this stage are $0, 0, 0, 0, 0, 0, i_7, i_6$

- Stage III: This stage does not performs any operation on the output of the second stage as the value of select signal is $S_0 = 0$. The final output at this stage are $0, 0, 0, 0, 0, 0, i_7, i_6$ which is logical right shift by 6 bits since $S_2S_1S_0 = 110(decimal6)$.

3.3 Reversible Bidirectional Logical Shifter

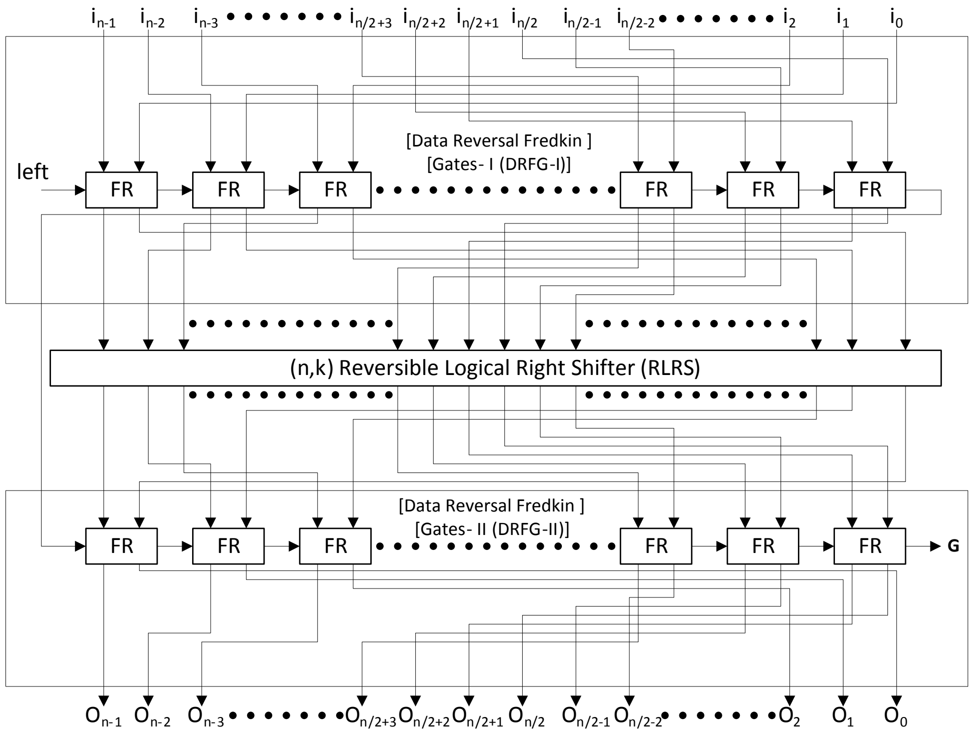
A (n,k) reversible bidirectional logical shifter is capable of logical shifting the input data in both the left and right directions depending on the value of the control signal labeled as $left$ in this section. All the operation that can be performed by a (n,k) reversible bidirectional logical shifter is shown in Table 3.3. From Table 3.3 it can be seen that when $left$=1 the reversible bidirectional logical shifter will perform the logical left shift operation

Table 3.3. Operations on (n,k) reversible bidirectional logical shifter

left	Operation Performed
0	Reversible logical right shift
1	Reversible logical left shift

else it will perform the logical right shift operation. A (n,k) reversible bidirectional logical shifter can be designed by extending the (n,k) reversible logical right shifter explained earlier in Section 3.1. As illustrated in Fig. 3.3 the n/2 Fredkin gates controlled by the control signal $left$ are added as a chain at both the inputs and outputs of the (n,k) reversible logical right shifter to make it work as a (n,k) reversible bidirectional logical shifter. In the (n,k) reversible bidirectional logical shifter to perform a logical left shift operation the value of control signal $left$ is one that will result in reversal of the input data by the n/2 Fredkin gates which are added as a chain. The reverse input data is passed to the (n,k) reversible logical right shifter to produce logical right shifted reversed input data. Finally the the logical right shifted reversed input data is reversed by the chain of n/2 Fredkin gates to produce the final outputs as logically left shifted input data.

The total number of Fredkin gates required to design a (n,k) reversible bidirectional logical shifter(rbls) is the summation of the number of Fredkin gates required to design a (n,k) reversible logical right shifter(rlrs) + n/2 Fredkin gates in the first stage+ n/2 Fredkin gates in the last stage. As explained in Section 3.1 the number of Fredkin gates needed to design a (n,k) reversible logical right shifter is $FR_{rlrs} = n * k$. Thus the total number of Fredkin gates required to design a (n,k) reversible bidirectional logical shifter(rbls) are $FR_{rbls}{=}n/2{+}FR_{rlrs}{+}n/2{=}n{*}k{+}n{=}n{*}(k{+}1)$. The design methodology does not use any additional Feynman gate compared to the design methodology for (n,k) reversible logical right shifter thus it will have the same number of Feynman gates as in (n,k) reversible logical right shifter(rlrs). Thus the total number of Feynman gates can be written as $FE_{rbls} = FE_{rlrs} = \sum_{m=1}^{k}(n - 2^{(k-m)})$.



i_{n-1} i_{n-2} i_{n-3} i_{n/2+3} i_{n/2+2} i_{n/2+1} i_{n/2} i_{n/2-1} i_{n/2-2} i_2 i_1 i_0
left
[Data Reversal Fredkin]
[Gates- I (DRFG-I)]
FR
(n,k) Reversible Logical Right Shifter (RLRS)
[Data Reversal Fredkin]
[Gates- II (DRFG-II)]
G
O_{n-1} O_{n-2} O_{n-3} O_{n/2+3} O_{n/2+2} O_{n/2+1} O_{n/2} O_{n/2-1} O_{n/2-2} O_2 O_1 O_0

Figure 3.3. A (n,k) reversible bidirectional logical shifter (FE: Feynman gate, FR: Fredkin gate, G: Garbage output)

3.4 Reversible Bidirectional Arithmetic and Logical Shifter

A reversible bidirectional arithmetic and logical shifter is capable can perform logical right shifting, arithmetic right shifting, logical left shifting and arithmetic left shifting operations. The proposed design approach is illustrated with an example of a (8,3) reversible bidirectional arithmetic and logical shifter as shown in Fig. 3.4. All operations that can be performed by a (8,3) reversible bidirectional arithmetic and logical shifter are shown in Table 3.4 for different values of control signals sra, sla and $left$. As explained in Table 3.4, the barrel shifter performs the various operations such as logical right shift, logical left shift etc. depending on the values of sra, sla and $left$ control signals. In the proposed design, the input data is represented as $i_7, i_6, i_5, i_4, i_3, i_2, i_1, i_0$ while the shift value is controlled by select signals represented as $S_2S_1S_0$.

The design of a reversible arithmetic and logical shifter can be divided into four modules: (i) Data reversal control unit-I, (ii) Arithmetic right shift control unit, (iii) Shifter unit which consists of three sub-modules that performs Stage I, Stage II and Stage III operations discussed later, (iii) Arithmetic left shift control unit, (iv) Data reversal control unit-II. The reversible design of the modules of the reversible bidirectional arithmetic and logical shifter along with their working are explained as follows:

Table 3.4. Operation performed by a (n,k) reversible bidirectional arithmetic and logical shifter

Operation performed	Control signal values		
Logical right shift	sra=0	sla=0	left=0
Arithmetic right shift	sra=1	sla=0	left=0
Logical left shift	sra=0	sla=0	left=1
Arithmetic left shift	sra=0	sla=1	left=1

3.4.1 Data Reversal Control Unit-I

The direction of the shift operation performed on reversible arithmetic and logical shifter is controlled by the control signal $left$ as can be seen in the Table 3.4. For the value of control signal $left$ as 1, the reversible bidirectional arithmetic and logical shifter performs the shift operation in the left direction, that is, the arithmetic left shift operation or logical left shift operation. Otherwise, for the value of $left$=0 the shift operation is performed in the right direction, that is, arithmetic right shift operation or logical right shift operation. The data reversal control unit-I has Fredkin gates as the key components, since two outputs of the Fredkin gate can work as 2:1 MUXes. By utilizing two outputs of the Fredkin gate as 2:1 Muxes, 4 Fredkin gates can be used to reverse the 8 bit input data. After observing the behavior of right shift and left shift operation. We noticed that for a n bit input data a left shift operation by k-bit can be performed in three steps : (i) reverse the input data, (ii) perform k bit right shift operation, and (iii) reverse the outputs of the step (ii). For example, for a 8-bit input data $i_7, i_6, i_5, i_4, i_3, i_2, i_1, i_0$ the three steps of logical left shift operation by 3 bits will be: (i) reverse $i_7, i_6, i_5, i_4, i_3, i_2, i_1, i_0$ to produce $i_0, i_1, i_2, i_3, i_4, i_5, i_6, i_7$, (ii) perform the 3 bit logical right shift operation to produce $0, 0, 0, i_0, i_1, i_2, i_3, i_4$, and (iii) reverse the outputs of step (ii) to yield $i_4, i_3, i_2, i_1, i_0, 0, 0, 0$. The date reversal control unit-I is shown in Fig. 3.4.

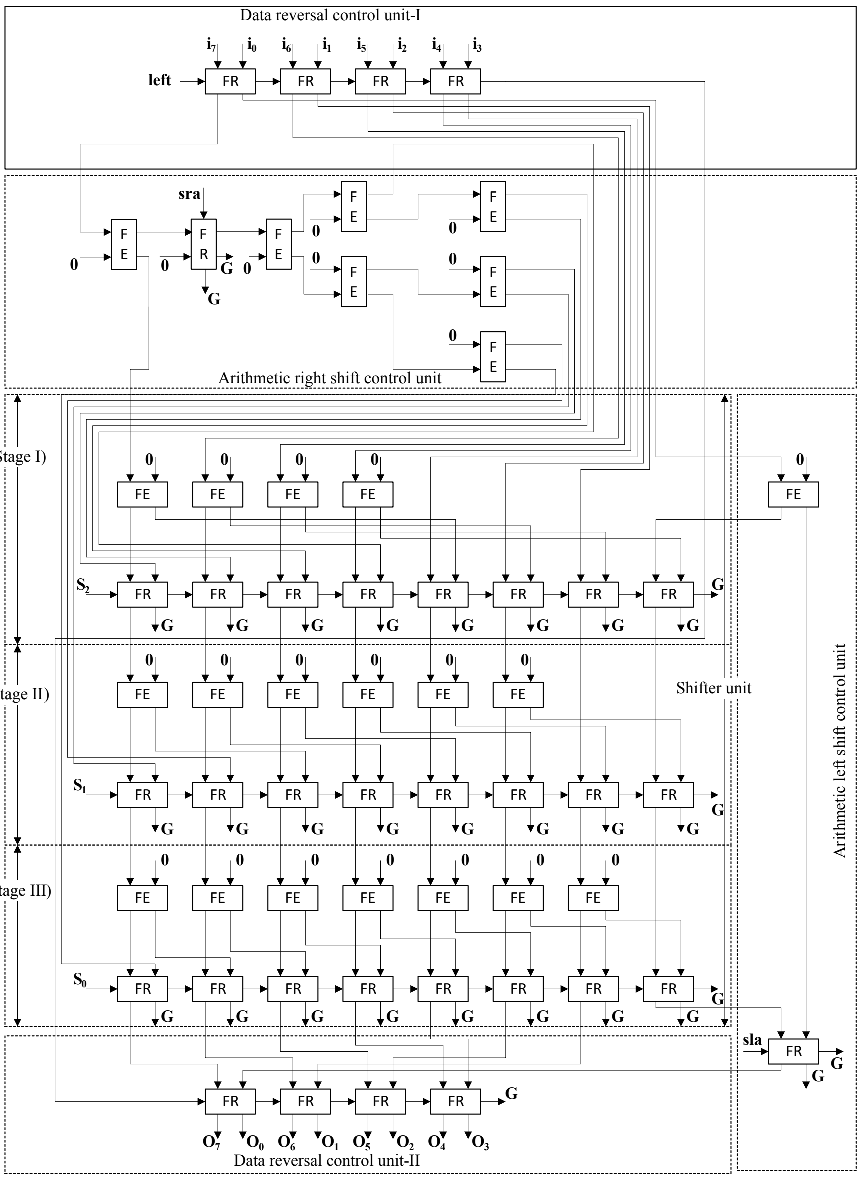

Data reversal control unit-I
left
sra
Arithmetic right shift control unit
(Stage I)
(Stage II)
(Stage III)
Shifter unit
Arithmetic left shift control unit
sla
Data reversal control unit-II

Figure 3.4. A (8,3) reversible bidirectional arithmetic and logical shifter (FE: Feynman gate, FR: Fredkin gate, G: Garbage output)

3.4.2 Arithmetic Right Shift Control Unit

The arithmetic right shift control unit controls the arithmetic right shift operation. This unit is designed using a single Fredkin gate controlled by the control signal sra, and preserves the sign bit of input data. If the value of control signal $sra = 1$, the arithmetic right shift operation is performed otherwise it simply passes the data to the next module. As fanout is not allowed in reversible logic multiple copies of the sign bit are created using the Feynman gates. The reversible arithmetic right shift control unit is shown in Fig.3.4.

3.4.3 Shifter Unit

The shifter unit in the design of reversible bidirectional arithmetic and logical shifter is responsible for the amount of shift operation performed. This unit is controlled by the control signals S_2, S_1 and S_0. The shifter unit can be divided into three stages. All the three stages are designed using the chain of 8 Fredkin gates controlled by the control signals S_2, S_1 and S_0. The first, second and the third stages of the shifter unit right shifts the input data by 2^2,2^1 and 2^0 bits depending on the value of control signal S_2, S_1 and S_0, respectively. Fig. 3.4 shows the three stage design of the reversible shifter unit. The Feynman gates are used in the design to avoid the fanout problem. The working of the three stages of the shifter unit is explained as follows:

- $Stage - I$: The first stage of shifter unit is responsible for shifting the input data by 2^2-bits and is controlled by the control signal S_2. If the value of control signal S_2 is 1 the input data is right shifted by 2^2-bits, else the input data remains unchanged. The outputs of the Stage I is passed as inputs to Stage II of the shifter unit.

- $Stage - II$: The second stage of the shifter unit works on the outputs of the first stage and is controlled by the control signal S_1. If the value of control signal S_1 is 1 the input data provided to the second stage is right shifted by 2^1-bits, else the input data remains unchanged. The outputs of the Stage II is passed as inputs to Stage III of the shifter unit.

- $Stage - III$: The third stage of the shifter unit is controlled by the control signal S_0. If the value of control signal S_0 is 1 the output data generated by the stage-II is right shifted by 2^0-bits else the output data remains unchanged. The outputs of this stage is passed as inputs to the next module in the design of reversible bidirectional arithmetic and logical shifter.

3.4.4 Arithmetic Left Shift Control Unit

The design of the arithmetic left shift control unit is similar to the design of the arithmetic right shift control unit. This control unit is responsible to perform the arithmetic left shift operation, and is controlled by the control signal sla. This unit is implemented using a single Fredkin gate. If the value of control signal $sla = 1$ this unit preserves the sign bit needed to perform the arithmetic left shift operation, else it simply passes the LSB of the shifter unit. The arithmetic left shift control unit is shown in the Fig. 3.4.

3.4.5 Data Reversal Control Unit II

The design of this unit is same as explained for data reversal control unit I and is shown in Fig. 3.4. The data reversal control unit II reverses its 8 bit input which consists of 7 bits from the outputs of the shifter unit and 1 bit from the output of the arithmetic left shift control unit. The data reversal control unit is controlled by the control signal $left$. If the value of control signal $left$ is 1, this unit reverses its input data to generate a left shifted result else it simply passes the input data to its outputs.

The proposed method illustrated above to design a (8,3) reversible bidirectional arithmetic and logical shifter can be generalized to design a (n,k) reversible bidirectional arithmetic and logical shifter. To design a (n,k) reversible bidirectional arithmetic and logical shifter, the Feynman gates are used for copying the input data. It is to be noted that in the proposed design Feynman gate is only used to avoid the fanout problem. The data reversal unit-I and data reversal unit-II consist of chains of n/2 Fredkin gates. The shifter unit at each stage requires 2^{k-m} for $m = 0 \quad to \quad (k-1)$ number of Feynman gates and chain of n Fredkin gates. The arithmetic right shift control unit uses one Fredkin and $2^k - 1$ Feynman gates. The arithmetic left shift control unit requires one Fredkin gate and one Feynman gate. Thus the total number of Feynman gates used to design a (n,k) reversible bidirectional arithmetic and logical shifter is:

FE=Number of Feynman gates required to design arithmetic right shift control unit+ number of Feynman gates used in shifter control unit+ number of Feynman gates used in arithmetic left shift control unit $= \sum_{m=0}^{k-1}(n-2^m) + (2^k - 1) + 1 {=} 2^k + \sum_{m=0}^{k-1}(n-2^m)$.

The total number of Fredkin gates used to design a (n,k) reversible bidirectional arithmetic and logical shifter can be written as:

FR= Number of Fredkin gates used in data reversal control unit-I+ Number of Fredkin gates used in arithmetic right shift control unit+ Number of Fredkin gates used in shifter unit+ Number of Fredkin gates used in arithmetic left shift control unit +Number of Fredkin gates used in data reversal control unit-II$=n/2 + 1 + (n*k) + 1 + n/2 {=} n*(k+1) + 2$.

The (8,3) reversible bidirectional arithmetic and logical shifter uses 25 Feynman gate to copy the input data to avoid the fanout, and 34 Fredkin gates are used for arithmetic and logical bidirectional shifting.

3.5 Reversible Universal Bidirectional Shifter

A reversible universal bidirectional shifter is capable of performing logical left and right shift, left and right rotate, and left and right arithmetic shift operation. A (n,k) reversible universal bidirectional shifter can be designed by extending the design methodology of (n,k) reversible universal right shifter explained earlier in Section 3.2. its design from the (n,k) reversible universal right shifter can be explained as : The chain of n/2 Fredkin gates controlled by the control signal $left$ are added at the inputs and outputs of the (n,k) reversible universal right shifter as illustrated in Fig.3.5. As can be seen in Fig. 3.5 there is an additional Fredkin gate after the (n,k) reversible universal right shifter that is to preserve the sign bit in case of arithmetic left shift operation when the value of control signal sla is 1. To copy the sign bit of the input data for arithmetic right shift an additional Feynman gate

Table 3.5. Operations on (n,k) reversible universal bidirectional shifter

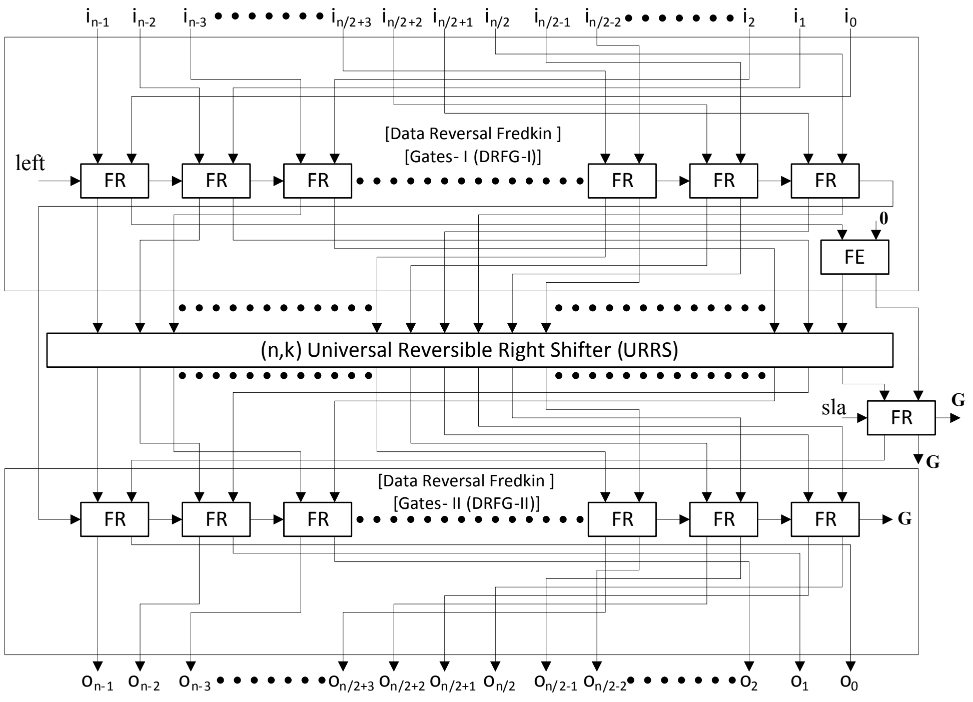
left	sra	rotate	Operation Performed
0	1	0	Reversible arithmetic right shift
0	0	1	Reversible right rotation
0	0	0	Reversible logical right shift
1	0	0	Reversible arithmetic left shift
1	0	1	Reversible left rotation
1	0	0	Reversible logical left shift

is used as shown in Fig. 3.5. All the operations that can be performed by a (n,k) reversible universal bidirectional shifter are shown in Table 3.5. For $left = 1$, the (n,k) reversible universal bidirectional shifter will perform all type of left shifts such as logical left shift or arithmetic left shift or left rotate operation. When $left = 0$ the (n,k) reversible universal bidirectional shifter will work as (n,k) reversible universal right shifter that can perform the operation such as logical right shift or arithmetic right shift or right rotate operation. The working of (n,k) reversible universal bidirectional shifter can be explained as follows:

- Stage I: In this stage, the input data $i_{n-1}, i_{n-2}, i_{n-3}, ..., i_2, i_1, i_0$ is provided to the n/2 Fredkin gates which are controlled by the control signal $left$. When the value of control signal $left$ is 1 the input data is reversed otherwise it remains same.

- Stage II: The outputs generated by the n/2 Fredkin gates are used as the inputs of the (n,k) reversible universal right shifter. The (n,k) reversible universal right shifter performs the right shift (logical or arithmetic or rotate) on its inputs and generates the right shifted outputs (logical or arithmetic or rotate).

- Stage III: The stage III consists of a single Fredkin gate and will be needed only if the arithmetic left shift operation is performed. The additional Fredkin gate used in the design as shown in Fig. 3.5 helps in changing the 0_{th} bit of the output generated by the (n,k) reversible universal right shifter based on the condition: $left = 1$ and $sla = 1$, to perform the arithmetic left operation. Thus, when the condition is satisfied the Fredkin gate will change the 0_{th} bit of the output generated by the (n,k) reversible universal right shifter to the sign bit i_{n-1}. If this stage is used, then in the final stage



i_{n-1} i_{n-2} i_{n-3} $i_{n/2+3}$ $i_{n/2+2}$ $i_{n/2+1}$ $i_{n/2}$ $i_{n/2-1}$ $i_{n/2-2}$ i_2 i_1 i_0
left
FR
[Data Reversal Fredkin]
[Gates- I (DRFG-I)]
FE
0
(n,k) Universal Reversible Right Shifter (URRS)
sla
G
[Data Reversal Fredkin]
[Gates- II (DRFG-II)]
o_{n-1} o_{n-2} o_{n-3} $o_{n/2+3}$ $o_{n/2+2}$ $o_{n/2+1}$ $o_{n/2}$ $o_{n/2-1}$ $o_{n/2-2}$ o_2 o_1 o_0

Figure 3.5. A (n,k) reversible universal bidirectional shifter (FE: Feynman gate, FR: Fredkin gate, G: Garbage output)

that consists on n/2 Fredkin gates and performs the operation of reversing the data, the outputs will be arithmetic left shifted result of the original inputs.

- Stage IV: This stage consists of n/2 Fredkin gates as shown in Fig. 3.5 which are controlled by the control signal $left$. For the value of control signal $left = 1$ the inputs passed to this stage are reversed by these n/2 Fredkin gates to produce the final output as logically left shifted input data or arithmetic left shifted input data or left rotated input data. Otherwise when $left = 0$, the final outputs will be same as the outputs of the Stage III which is logically right shifted input data or arithmetic right shifted input data or right rotated input data.

The number of Fredkin gates required to design a (n,k) reversible universal bidirectional right shifter is summation of number of Fredkin gates in Stage I+ number of Fredkin gates in Stage II+ number of Fredkin gates in Stage III+ number of Fredkin gates in

Stage IV. Thus, the total number of Fredkin gates needed to design a (n,k) reversible universal bidirectional shifter(rubs) can be written as $FR_{rubs} = n/2 + FR_{rurs} + 1 + n/2 = n/2 + (n * k + \sum_{k=0}^{k-1} 2^k + 1) + 1 + n/2 = n(k+1) + \sum_{k=0}^{k-1} 2^k + 2$, here Stage II is a (n,k) reversible universal right shifter(rurs) that consists of $FR_{rurs} = n * k + \sum_{k=0}^{k-1} 2^k + 1$ Fredkin gates. The number of Feynman gates required to design a (n,k) reversible universal bidirectional shifter is one more than the number of Feynman gates required to design a (n,k) reversible universal right shifter. The extra Feynman is needed for copying the sign bit as illustrated in Fig.3.5. Thus, the total number of Feynman gates needed in the design are $FE_{rubs} = FE_{rurs} + 1 = (n * k + 2^k - 1) + 1 = n * k + 2^k$.

CHAPTER 4

COST MEASUREMENT

All the proposed reversible design methodologies for the reversible barrel shifter are evaluated in terms of the number of the garbage outputs, number of ancilla inputs and the quantum cost.

4.1 Garbage Outputs

The garbage outputs represent the number of outputs that do not perform any useful operation and thus is considered as an overhead. Hence, we used the number of garbage outputs as an important criteria to evaluate the efficiency of the different design methodologies for reversible barrel shifters proposed in this work.

4.1.1 Number of Garbage Outputs in Reversible Logical Right Shifter (rlrs)

Each stage of a (n,k) reversible logical right shifter has the chain of n Fredkin gates, in which n-1 Fredkin gates in each stage have one garbage output and the nth Fredkin gate has two garbage outputs. The *nth* Fredkin gate has two garbage outputs because one of its output which regenerates the control signal is not utilized further in the computation. Thus the total number of garbage outputs in the design methodology for a (n,k) reversible logical right shifter(rlrs) can be written as: GO_{rlrs}=$n * k + k$=$n * (k + 1)$. Table 4.1 shows the garbage outputs for different values of n and k, where n is input data bits and k represents the shift values.

Table 4.1. Garbage outputs in (n,k) reversible logical right shifter

n/k	n=4	n=8	n=16	n=32	n=64
k=2	10	18	34	66	130
k=3		27	51	99	195
k=4			68	132	260
k=5				165	325
k=6					390

4.1.2 Number of Garbage Outputs in Reversible Universal Right Shifter (rurs)

In the (n,k) reversible universal right shifter as illustrated in Section 3.2, there are k stages and each stage consists of the chain of n Fredkin gates to perform the shift operation, and 2^{k-i} Fredkin gates to perform the rotate operation if needed, where i represents the stage on which the operation is being performed. The n Fredkin gates in each stage that are used to perform shift operation have one garbage output except the 1st Fredkin gate which has two garbage outputs. The 2^{k-i} Fredkin gates in each stage that are used to perform the rotate operation each will have 1 garbage output, except the last Fredkin gate in the final stage that has 2 garbage outputs. In addition, there will be 2 garbage outputs from the Fredkin gate that is used to initialize the arithmetic right shift operation as can be seen from the example of (8,3) reversible universal right shifter shown in Fig. 3.2. Thus, the total number of garbage outputs in (n,k) reversible universal right shifter will be $GO_{rurs} = (n-1)*k + 2*k + (\sum_{m=0}^{k-1} 2^m + 1) + 2 = (n+1)*k + \sum_{m=0}^{k-1} 2^m + 3$. Table 4.2 shows the garbage output in the (n,k) reversible universal right shifter for different values of n and k.

Table 4.2. Garbage outputs in (n,k) reversible universal right shifter

n/k	n=4	n=8	n=16	n=32	n=64
k=2	16	24	40	72	136
k=3		37	61	109	205
k=4			86	150	278
k=5				199	359
k=6					456

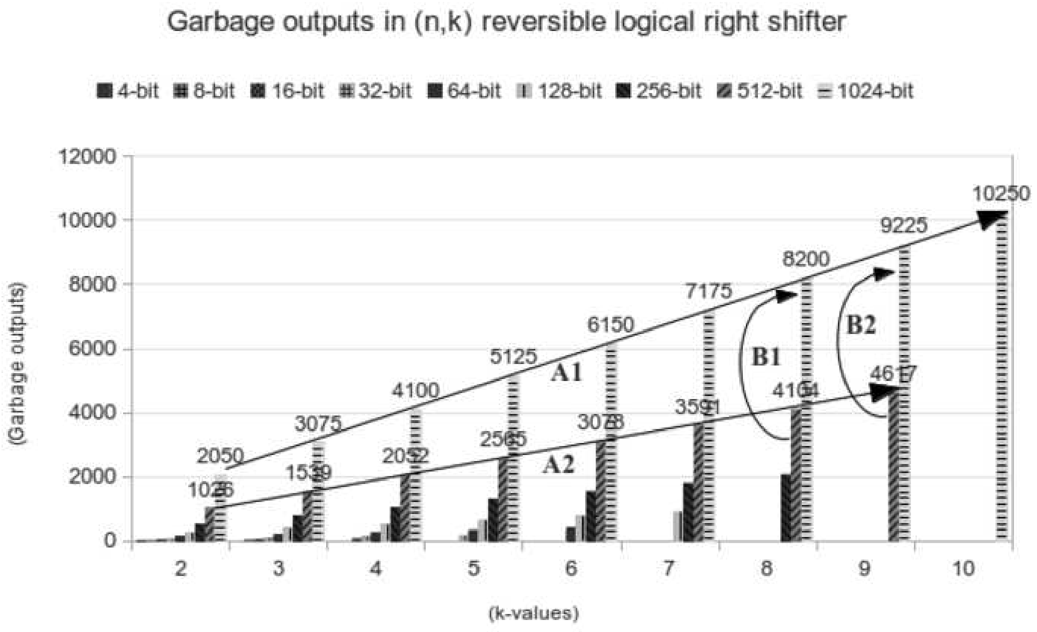


Figure 4.1. Garbage outputs in (n,k) reversible logical right shifter

4.1.3 Number of Garbage Outputs in Reversible Bidirectional Logical Shifter (rbls)

As illustrated earlier a (n,k) reversible bidirectional logical shifter can be designed by extending the (n,k) reversible logical right shifter by adding n/2 Fredkin gates as a chain both at the inputs and the outputs of the (n,k) reversible logical right shifter. In our design, we have carefully connected the chain of additional n/2 Fredkin gates at the inputs and the outputs of (n,k) reversible logical right shifter such that only one garbage output is produced. This one garbage output is due to a control signal called $left$ that remains unused in the last Fredkin gate of the final stage. Thus, the total number of garbage outputs for a (n,k) reversible universal bidirectional shifter(rbls) is the number of garbage outputs produced by a (n,k) reversible logical right shifter(rlrs)+ one additional garbage output, and can be written as GO_{rbls}=$GO_{rlrs}+1$=$n*k+k+1$=$k*(n+1)+1$. Table 4.3 shows

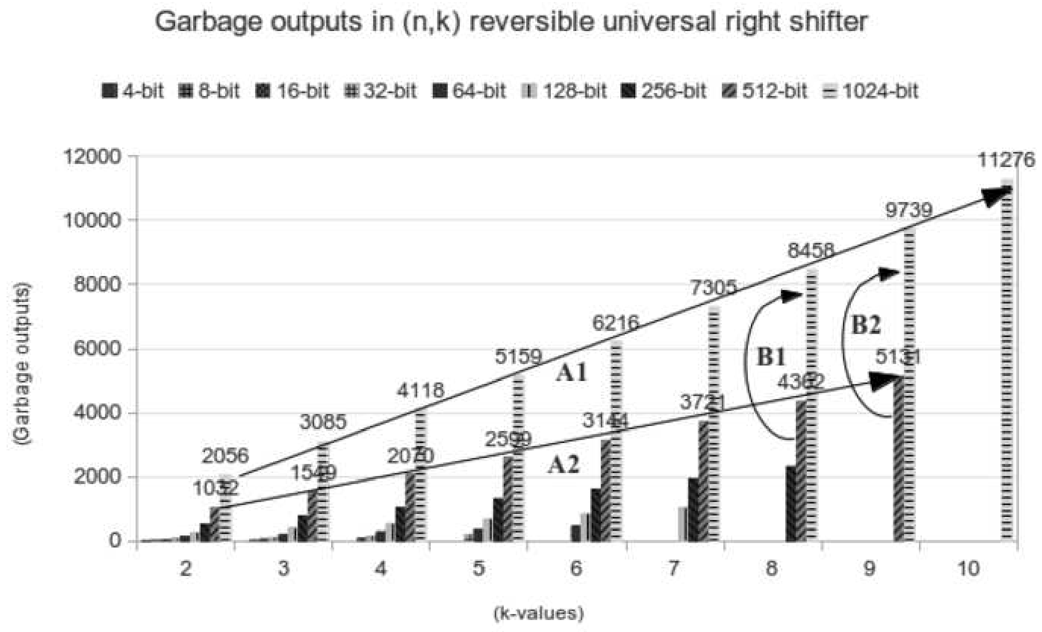


Figure 4.2. Garbage outputs in (n,k) reversible universal right shifter

the number of garbage outputs produced by the (n,k) reversible bidirectional logical shifter for different values of n and k.

Table 4.3. Garbage outputs in (n,k) reversible bidirectional logical shifter

n/k	n=4	n=8	n=16	n=32	n=64
k=2	11	19	35	67	131
k=3		28	52	100	196
k=4			69	133	261
k=5				166	326
k=6					391

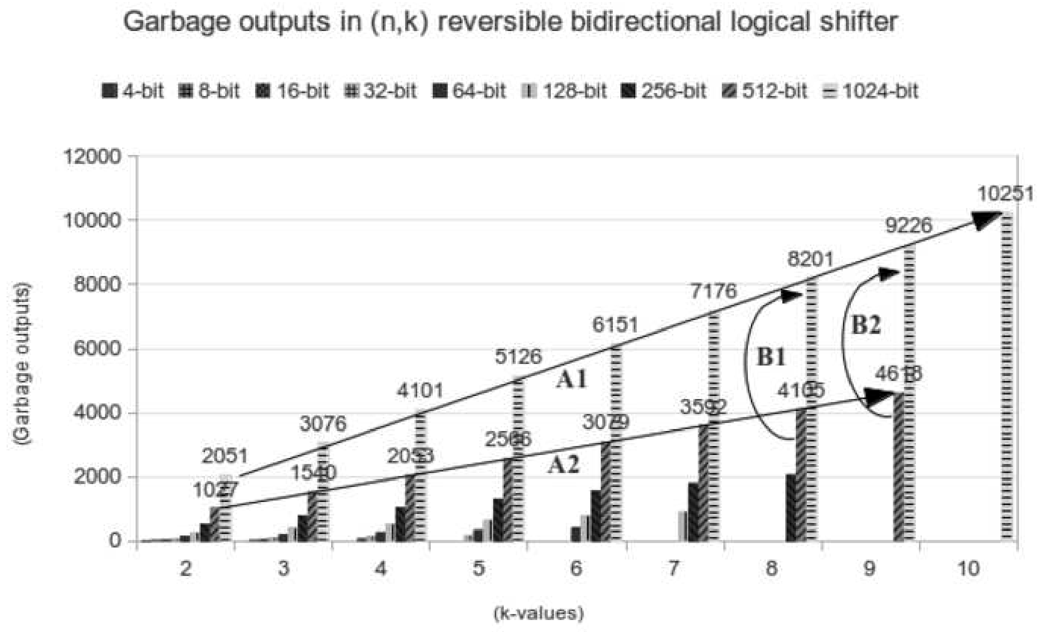


Figure 4.3. Garbage outputs in (n,k) reversible bidirectional logical shifter

4.1.4 Number of Garbage Outputs in Reversible Bidirectional Arithmetic and Logical Shifter (rbals)

The shifter unit in the design of (n,k) reversible bidirectional arithmetic and logical shifter (rbals) can be designed in k stages and each stage consists of the chain of n Fredkin gates to perform the shift operation. Each Fredkin gate in the chain of n Fredkin gates produces atleast one garbage output except the last Fredkin gate which produces two garbage outputs. Further, each Fredkin gate used in the design of arithmetic right shift control unit and arithmetic left shift control unit produces two garbage outputs. The last Fredkin gate of the data reversal control unit-II produces one garbage output as the control signal $left$ cannot be utilized further. Hence the number of garbage outputs required to design a (n,k) reversible bidirectional arithmetic and logical shifter (rbals) can be written as GO_{rbals}=$k(n+1)+5$. Table 4.4 shows the number of garbage outputs produced for different reversible bidirectional arithmetic and logical shifter designs. In the table, n is the number of input data bits and k represents the shift value. It can be seen that the number of garbage outputs in (8,3) reversible bidirectional arithmetic and logical shifter design that was illustrated in Fig. 3.4 are 32 which matches with the result in Table 4.4.

Table 4.4. Garbage outputs in (n,k) reversible bidirectional arithmetic and logical shifter

n/k	n=4	n=8	n=16	n=32	n=64
k=2	15	23	39	71	135
k=3		32	56	104	200
k=4			73	137	265
k=5				170	330
k=6					395

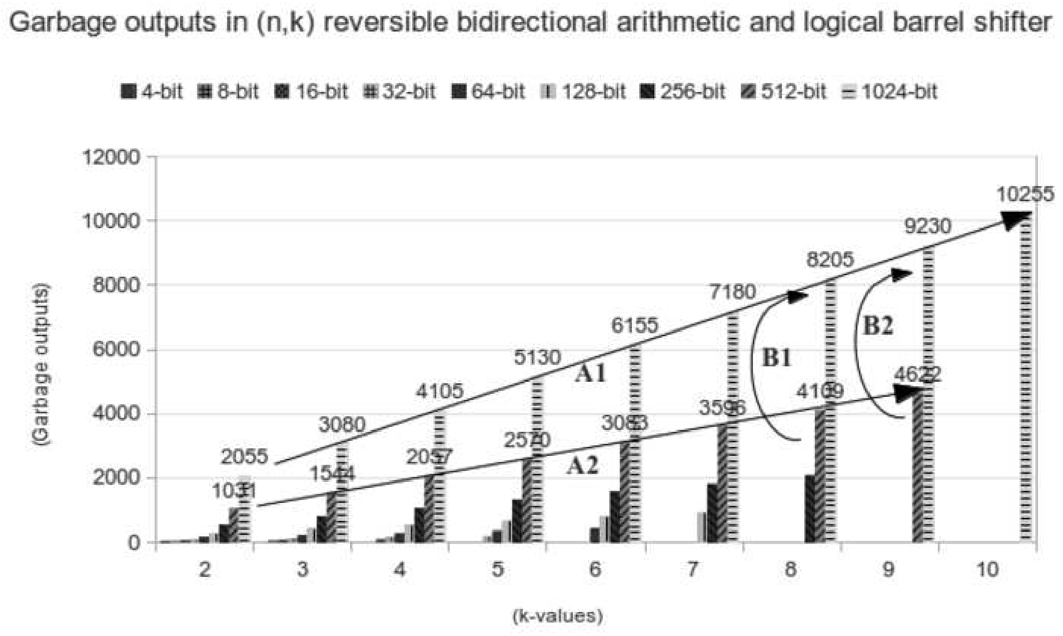


Figure 4.4. Garbage outputs in (n,k) reversible bidirectional arithmetic and logical shifter

4.1.5 Number of Garbage Outputs in Reversible Universal Bidirectional Shifter (rubs)

A (n,k) reversible universal bidirectional shifter can be designed by extending the (n,k) reversible universal right shifter by adding the n/2 Fredkin gates as a chain both at the inputs and the outputs of the (n,k) reversible universal right shifter. The number of garbage outputs generated by a (n,k) reversible universal bidirectional shifter(rubs) can be written as the summation of the number of garbage outputs to design the (n,k) reversible universal right shifter, the two garbage outputs produced by the Fredkin gate used in Stage III as explained earlier in Section 3.2, the single garbage output generated by the chain of n/2 Fredkin gates in Stage IV. Thus,the total number of garbage outputs are $GO_{rubs} = GO_{rurs} + 2 + 1 = (n*k + \sum_{m=0}^{k-1} 2^m + k + 3) + 2 + 1 = (n+1)k + \sum_{m=0}^{k-1} 2^m + 6$. Table 4.5 shows the number of garbage outputs generated in the design methodology of a (n,k) reversible universal bidirectional shifter for the different values of n and k.

Table 4.5. Garbage outputs in (n,k) reversible universal bidirectional shifter

n/k	n=4	n=8	n=16	n=32	n=64
k=2	17	25	41	73	137
k=3		38	62	110	206
k=4			87	151	279
k=5				200	360
k=6					457

4.1.6 Analysis of The Impact of n And k Values on The Garbage Outputs

For the reversible barrel shifters proposed in this work, we have plotted the graphs of the number of garbage outputs by varying the values of n and k. The reason for plotting the graphs is to evaluate the impact of varying n by keeping k as a constant, and vice versa. This study of the affect of n and k on the number of garbage outputs will help the designer to choose among the two possible design approaches for designing the reversible barrel shifter: (1) increase the value of n and keeping the k as a constant, (2) increase the value of k and keeping the n as a constant. By referring to the graph, the designer can evaluate

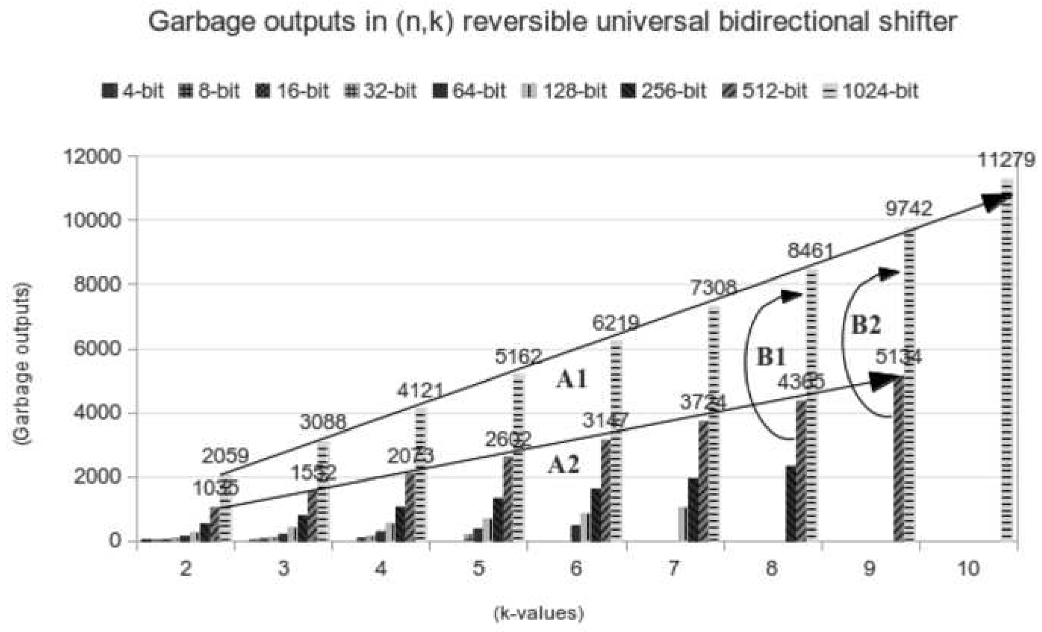


Figure 4.5. Garbage outputs in (n,k) reversible universal bidirectional shifter

the advantages and disadvantages of both the approaches in terms of garbage outputs, and choose the value of n and k that can generate the design with reduced overhead of garbage outputs. The graphs of the number of garbage outputs by varying the values of n and k for the various reversible design methodologies for the barrel shifters proposed in this work are shown in Figures 4.1, 4.2, 4.3, 4.4 and 4.5. In the graphs we have shown two cases A1 and A2 in which the values of n are kept constant as 1024 and 512, respectively, while the values of k are varied from 2 to 10 for A1 and 2 to 9 for A2. From the arrows A1 and A2, it can be easily seen that there is a linear growth in the number of garbage outputs when n is kept as a constant while the values of k are varied. The B1 and B2 curves in the graphs shows the cases where the values of k are kept constant as k=9 and k=10, respectively, while the values of n is varied from 512 to 1024. It can be observed from the arrows B1 and B2 that when the value of n is increased in power of 2 while keeping k as a constant, there is 2X (doubles) increase in the number of garbage outputs compared to the previous value of n. Thus, it can be concluded that the number of garbage outputs for the proposed (n,k) reversible barrel shifters increases more rapidly by varying n and keeping k as a constant compared to the reversible barrel shifter designs in which n is kept as a constant while the values of k are varied.

4.2 Ancilla Inputs

The constant input bits called the ancilla inputs are used to realize the boolean logic functions from the reversible gates. The more the number of the ancilla inputs needed in the design, the more will be the number of bits need to realize a reversible circuit. Due to technology limitations, the quantum computers of many qubits that have the reversible gates as the basic building blocks are difficult to realize, the ancilla inputs in the reversible gates are considered as a major overhead and need to be minimized. Thus, we evaluated the proposed design methodologies for the reversible barrel shifter in terms of number of ancilla inputs as discussed in the following.

4.2.1 Ancilla Inputs in Reversible Logical Right Shifter (rlrs)

The total number of Feynman gates used in the design methodology for a (n,k) reversible logical right shifter (rlrs) are $FE_{rlrs} = \sum_{m=1}^{k}(n - 2^{(k-m)})$. Each Feynman gate used in the design has one ancilla input. Further, in each stage of the (n,k) reversible logical right shifter the chain of n Fredkin gates requires $2^(k - m)$ number of ancilla inputs, where m represents the stage of the barrel shifter and will have values from 1 to k depending on the stage number. Thus the total number of ancilla inputs required to design a (n,k) reversible logical right shifter can be written as $AN_{rlrs} = \sum_{m=1}^{k}(n - 2^{(k-m)}) + \sum_{m=1}^{k}(2^{(k-m)}) = n * k$. Table 4.6 shows the number of ancilla inputs needed to design a reversible logical right shifter for different values of n and k.

Table 4.6. Ancilla inputs in (n,k) reversible logical right shifter

n/k	n=4	n=8	n=16	n=32	n=64
k=2	8	16	32	64	128
k=3		24	48	96	192
k=4			64	128	256
k=5				160	320
k=6					384

4.2.2 Ancilla Inputs in Reversible Universal Right Shifter (rurs)

As illustrated earlier in Section 3.2, the total number of Feynman gates required to design a (n,k) reversible universal right shifter (rurs) are $FE_{rurs} = n * k + 2^k - 1$. Each Feynman gate in the design will have one ancilla input. Further, in the initial stage of designing (n,k) reversible universal right shifter one additional Fredkin gate is used to provide the arithmetic right shift operation which requires one additional ancilla input. Thus the total number of ancilla inputs for (n,k) reversible universal right shifter can be written as $AN_{rurs} = n * k + 2^k - 1 + 1 = n * k + 2^k$, and Table 4.7 shows the number of ancilla inputs to design a reversible universal right shifter for different values of n and k.

Table 4.7. Ancilla inputs in (n,k) reversible universal right shifter

n/k	n=4	n=8	n=16	n=32	n=64
k=2	12	20	36	68	132
k=3		32	56	104	200
k=4			80	144	272
k=5				192	352
k=6					448

4.2.3 Ancilla Inputs in Reversible Bidirectional Logical Shifter (rbls)

The number of ancilla inputs required to design a (n,k) reversible bidirectional logical shifter(rbls) will be same as the number of ancilla inputs required to design a (n,k) reversible logical right shifter(rlrs). Thus,the number of ancilla inputs required to design a (n,k) reversible bidirectional logical shifter(rbls) can be written as $AN_{rbls} = AN_{rlrs} = n * k$. It can observed that the proposed design methodology for the reversible bidirectional logical

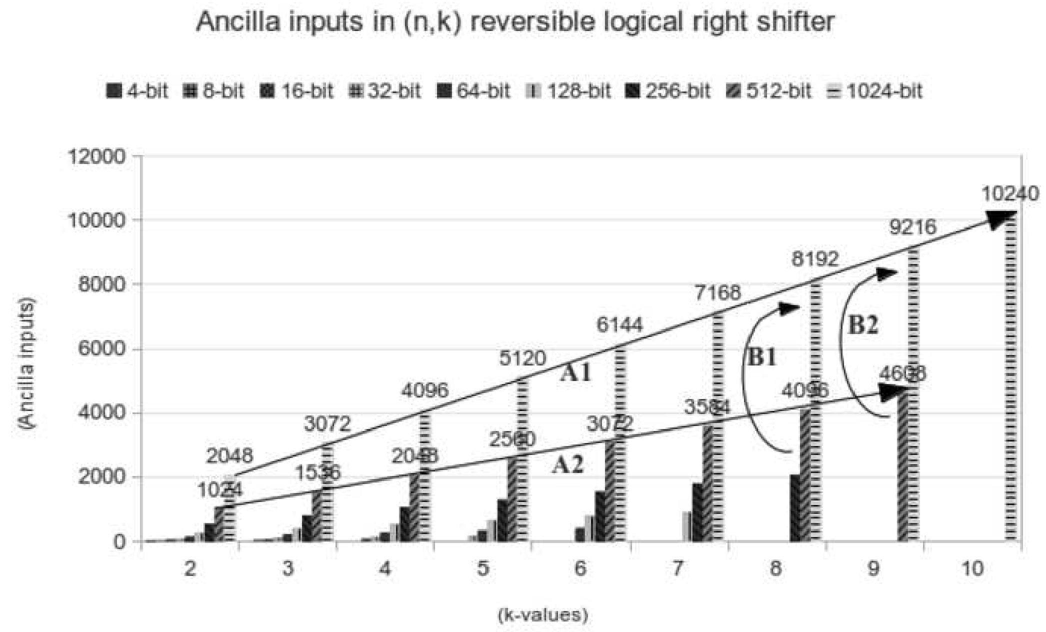


Figure 4.6. Ancilla inputs in (n,k) reversible logical right shifter

shifter is an efficient design as it uses the same number of ancilla inputs as used in the design methodology for reversible logical right shifter but provide the additional functionality of logical left shift compared to reversible logical right shifter which can only perform logical right shift operation. Table 4.8 shows the number of ancilla inputs for different values of n and k.

Table 4.8. Ancilla inputs in (n,k) reversible bidirectional logical shifter

n/k	n=4	n=8	n=16	n=32	n=64
k=2	8	16	32	64	128
k=3		24	48	96	192
k=4			64	128	256
k=5				160	320
k=6					384

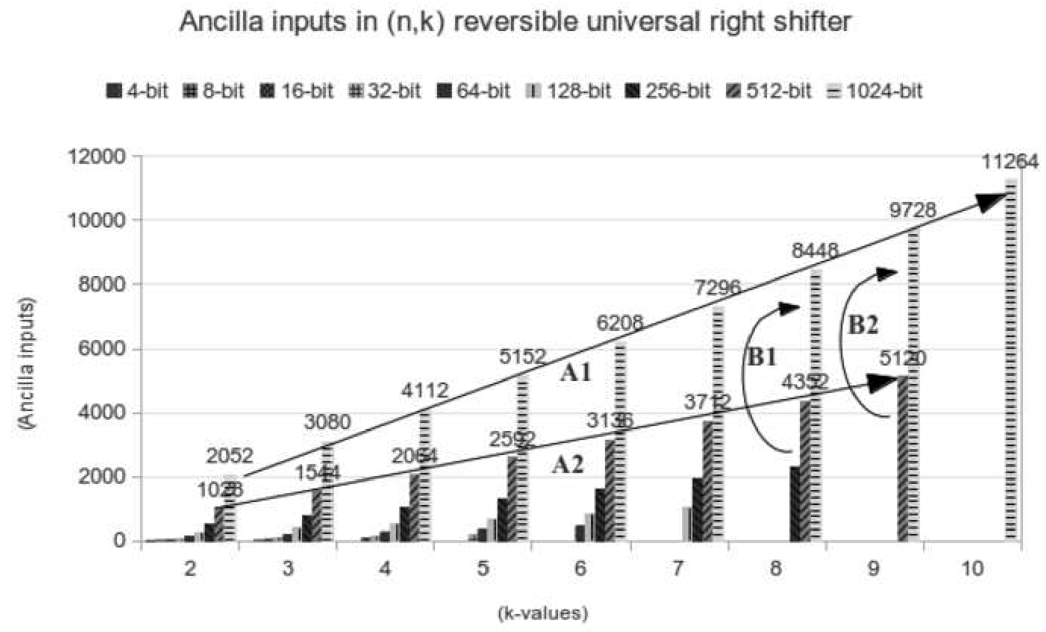


Figure 4.7. Ancilla inputs in (n,k) reversible universal right shifter

4.2.4 Ancilla Inputs in Reversible Bidirectional Arithmetic and Logical Shifter (rbals)

A (n,k) reversible bidirectional arithmetic and logical shifter can be designed using $2^k + \sum_{m=0}^{k-1}(n-2^m)$ Feynman gates. Each Feynman gate requires one ancilla input bit to copy the input data. Additionally, the Fredkin gate used in arithmetic right shift control unit requires one ancilla bit. Thus the total number of ancilla inputs required to design a (n,k) reversible bidirectional arithmetic and logical shifter (rbals) is $AN_{rbals}=2^k + \sum_{m=0}^{k-1}(n-2^m) + 1$. The table 4.9 shows the number of ancilla bits required to design a reversible bidirectional arithmetic and logical shifter (rbals) for different values of n and k. It can be seen from the Table 4.9 that the total number of ancilla inputs to design a (8,3) reversible bidirectional arithmetic and logical shifter are 26 which is same as illustrated in Fig. 3.4.

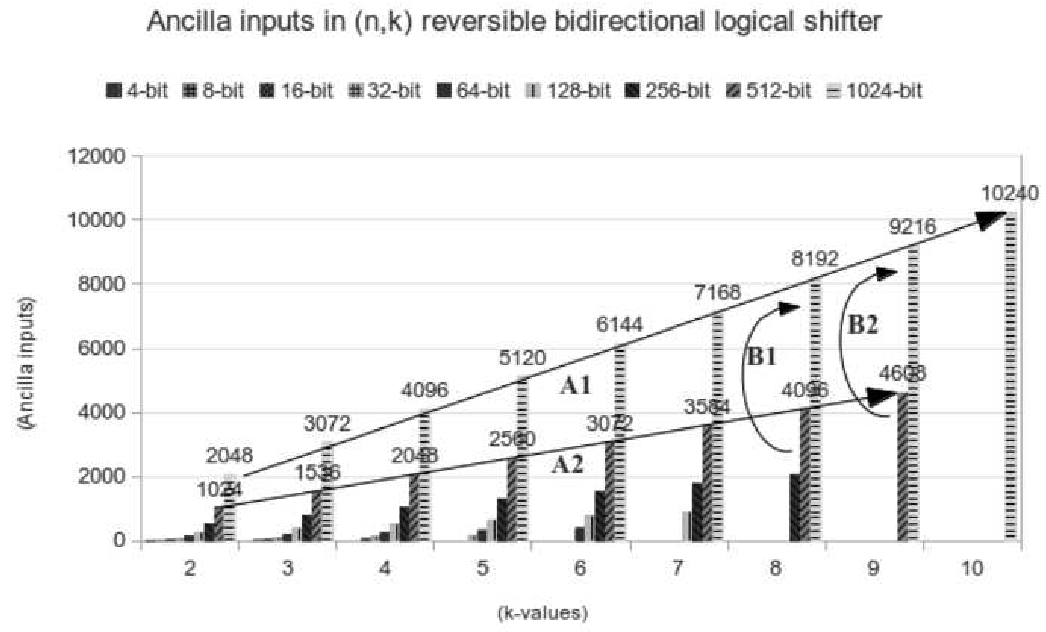


Figure 4.8. Ancilla inputs in (n,k) reversible bidirectional logical shifter

4.2.5 Ancilla Inputs in Reversible Universal Bidirectional Shifter (rubs)

The total number of ancilla inputs needed in the design methodology for a (n,k) reversible universal bidirectional shifter will be one more than the number of ancilla inputs needed in the design methodology for (n,k) reversible universal right shifter. Compared to the design methodology for a (n,k) reversible universal right shifter, a (n,k) reversible universal bidirectional shifter will have one additional ancilla input because of the use of an additional Feynman gate to copy the sign bit to perform the arithmetic left shift operation. Thus the total number of ancilla inputs for a (n,k) reversible universal bidirectional shifter(rubs) can be written as $AN_{rubs} = AN_{rurs} + 1 = n * k + 2^k + 1$. Table 4.10 shows the number of ancilla inputs in the design methodology for a (n,k) reversible universal bidirectional shifter for different values of n and k.

Table 4.9. Ancilla inputs in (n,k) reversible bidirectional arithmetic and logical shifter

n/k	n=4	n=8	n=16	n=32	n=64
k=2	10	18	34	66	130
k=3		26	50	98	194
k=4			66	130	258
k=5				162	322
k=6					386

Table 4.10. Ancilla inputs in (n,k) reversible universal bidirectional shifter

n/k	n=4	n=8	n=16	n=32	n=64
k=2	13	21	37	69	133
k=3		33	57	105	201
k=4			81	145	273
k=5				193	353
k=6					449

4.2.6 Analysis of The Impact of n And k Values on The Ancilla Inputs

For the reversible barrel shifters proposed in this work, we have plotted the graphs of the number of ancilla inputs by varying the values of n and k similar to the ones plotted for number of garbage outputs. The graphs of the number of ancilla inputs by varying the values of n and k for the various reversible design methodologies for the barrel shifters proposed in this work are shown in Figures 4.6, 4.7, 4.8, 4.9 and 4.10. From the graphs, we observed that the impact of n and k on the number of ancilla inputs is similar to their impact on the number of the garbage outputs. This leads us to conclude that the number of ancilla inputs for the proposed (n,k) reversible barrel shifters increases more rapidly by varying n and keeping k as a constant compared to the reversible barrel shifter designs in which n is kept as a constant while the values of k are varied.

4.3 Quantum Cost

In the quantum cost calculation, the quantum cost of the Feynman gate is considered as one, while the quantum cost of the Fredkin gate is considered as five, similar to as considered by the researchers in [18, 19]. Thus, the quantum cost of a proposed design methodology for a reversible barrel shifters can be written as $QuantumCost =$

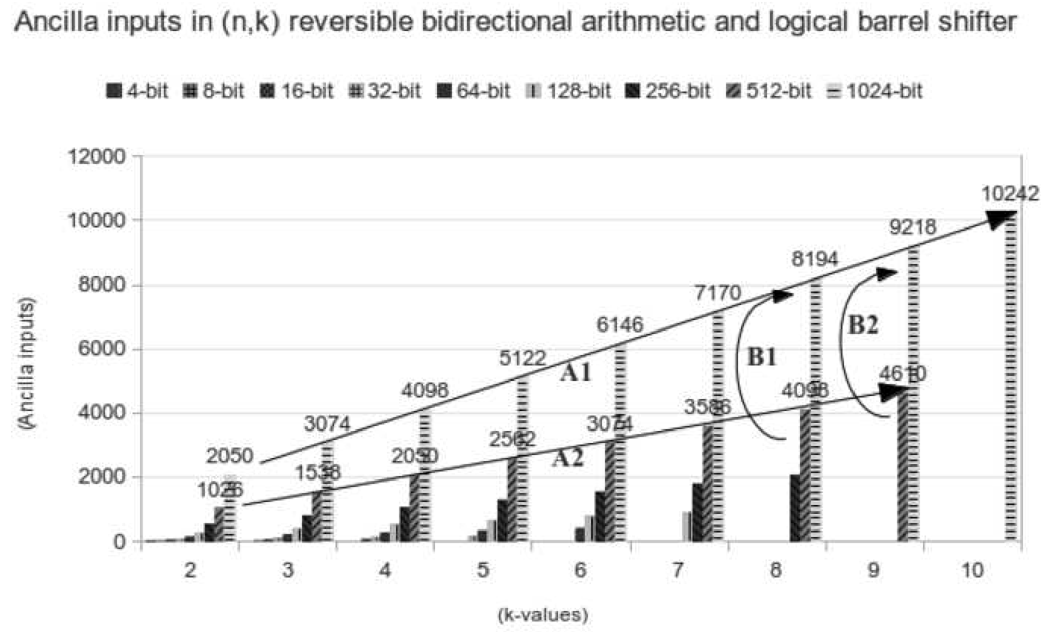


Figure 4.9. Ancilla inputs in (n,k) reversible bidirectional arithmetic and logical shifter

$5 * (number \quad of \quad Fredkin \quad gates) + (number \quad of \quad Feynman \quad gates)$. Thus the quantum cost of various reversible design methodologies for barrel shifters proposed in this work as follows.

- Reversible logical right shifter: The quantum cost for a (n,k) reversible logical right shifter(rlrs) can be written as $QuantumCost_{rlrs} = 5 * FR_{rlrs} + FE_{rlrs}$=$5 * (n * k) + (\sum_{m=1}^{k}(n - 2^{(k-m)}))$, here FR_{rlrs} is the number of Fredkin gates and FE_{rlrs} is the number of Feynman gates required to design a (n,k) reversible logical right shifter.

- Reversible universal right shifter: The quantum cost for a (n,k) reversible universal right shifter(rurs) can be written as $QuantumCost_{rurs} = 5 * FR_{rurs} + FE_{rurs}$=$5 * (n * k + \sum_{k=0}^{k-1} 2^k + 1) + (n * k + 2^k - 1)$, here FR_{rurs} is the number of Fredkin gates and FE_{rurs} is the number of Feynman gates required to design a (n,k) reversible universal right shifter.

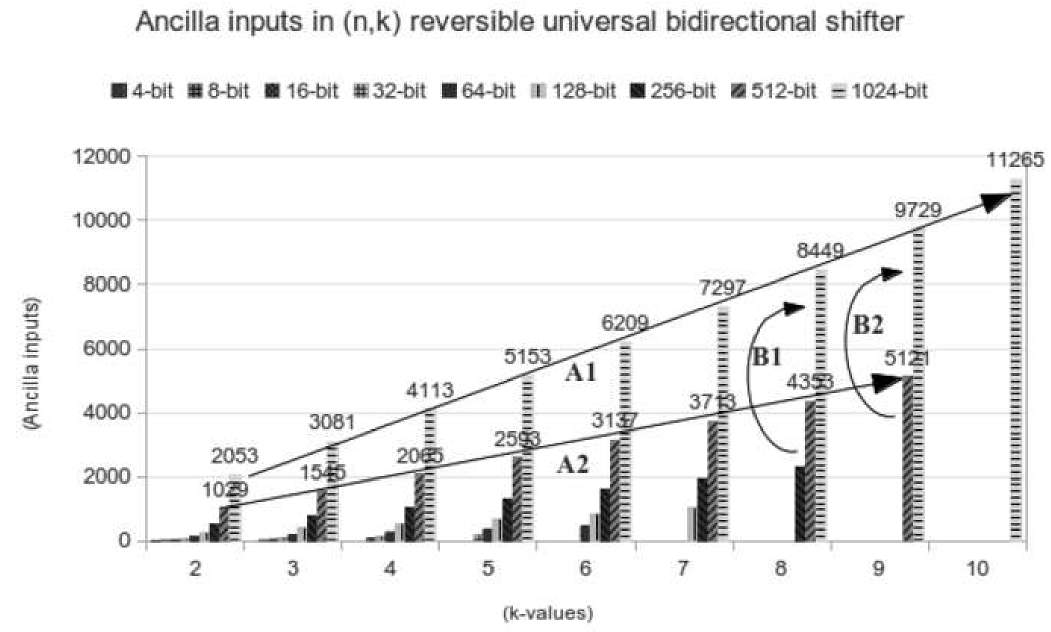


Figure 4.10. Ancilla inputs in (n,k) reversible universal bidirectional shifter

- Reversible bidirectional logical shifter: The quantum cost for a (n,k) reversible bidirectional logical shifter(rbls) can be written as $QuantumCost_{rbls} = 5 * FR_{rbls} + FE_{rbls}$=$5 * (n * (k + 1)) + (\sum_{m=1}^{k}(n - 2^{(k-m)}))$, here FR_{rbls} is the number of Fredkin gates and FE_{rbls} is the number of Feynman gates required to design a (n,k) reversible bidirectional logical shifter.

- Reversible bidirectional arithmetic and logical shifter: The quantum cost of the proposed design of reversible bidirectional arithmetic and logical shifter (rbals) can be written as $QuantumCost_{rbals} = 5 * (FR_{rbals}) + FE_{rbals}$=$5 * (n * (k + 1) + 2) + 2^k + \sum_{m=0}^{k-1}(n-2^m)$. here FR_{rbals} is the number of Fredkin gates and FE_{rbals} is the number of Feynman gates required to design a (n,k) reversible bidirectional arithmetic and logical shifter.

- Reversible universal bidirectional shifter: The quantum cost for a (n,k) reversible universal bidirectional shifter(rubs) can be written as $QuantumCost_{rubs} = 5{*}FR_{rubs}+FE_{rubs}$=$5 * (n(k+1) + \sum_{k=0}^{k-1} 2^k + 2) + (n * k + 2^k)$, here FR_{rubs} is the number of Fredkin gates and FE_{rubs} is the number of Feynman gates required to design a (n,k) reversible universal bidirectional shifter.

Tables 4.11(a), 4.11(b), 4.11(c), 4.11(d) and 4.11(e) shows the quantum costs in the design methodologies for proposed reversible barrel shifters for different values of n and k.

4.3.1 Analysis of The Impact of n And k Values on The Quantum Cost

For the reversible barrel shifters proposed in this work, we have also plotted the graphs of quantum cost by varying the values of n and k similar to the ones plotted for number of garbage outputs and the number of ancilla inputs. *The graphs are not shown as we found that the impact of n and k on the quantum cost is similar to their impact on the number of the garbage outputs and the number of ancilla inputs.* Thus, the quantum cost of the proposed (n,k) reversible barrel shifters increases more rapidly by varying n and keeping k as a constant compared to the reversible barrel shifter designs in which n is kept as a constant while the values of k are varied.

Table 4.11. Quantum costs in the proposed (n,k) reversible barrel shifters

(a) Quantum cost in (n,k) reversible logical right shifter

n/k	n=4	n=8	n=16	n=32	n=64
k=2	45	93	189	381	765
k=3		137	281	569	1145
k=4			369	753	1521
k=5				929	1889
k=6					2241

(b) Quantum cost in (n,k) reversible universal right shifter

n/k	n=4	n=8	n=16	n=32	n=64
k=2	71	119	215	407	791
k=3		191	335	623	1199
k=4			479	863	1631
k=5				1151	2111
k=6					2687

(c) Quantum cost in (n,k) reversible bidirectional logical shifter

n/k	n=4	n=8	n=16	n=32	n=64
k=2	65	133	269	541	1085
k=3		177	361	729	1465
k=4			449	913	1841
k=5				1089	2209
k=6					2561

(d) Quantum cost in (n,k) reversible bidirectional arithmetic and logical shifter

n/k	n=4	n=8	n=16	n=32	n=64
k=2	79	147	283	555	1099
k=3		195	379	747	1483
k=4			475	939	1867
k=5				1131	2251
k=6					2635

(e) Quantum cost in (n,k) reversible universal right shifter

n/k	n=4	n=8	n=16	n=32	n=64
k=2	77	125	221	413	797
k=3		197	341	629	1205
k=4			485	869	1637
k=5				1157	2117
k=6					2693

CHAPTER 5

CONCLUSIONS

In this work, we have presented methodologies for the design of (n,k) reversible barrel shifters such as the reversible logical right-shifter, the reversible universal right shifter that supports logical right shift, arithmetic right shift and the right rotate; the reversible bidirectional logical shifter; the reversible bidirectional arithmetic and logical shifter and the design methodology for the reversible universal bidirectional shifter that supports bidirectional logical and arithmetic shift and rotate operations. The functional verification of all the proposed reversible design methodologies for the barrel shifters are done in Verilog HDL in which library of the reversible gates coded in Verilog HDL are used as the basic building blocks. In this work, the design methodologies for the reversible barrel shifters have considered the number of garbage outputs, the number of ancilla inputs, and the quantum cost as the parameters to be optimized. This is achieved by the use of the Fredkin gates and the Feynman gates, and the best possible reuse of the garbage outputs and the ancilla inputs. For example, the (n,k) reversible bidirectional logical shifter that can perform logical left and right shift operations has the same number of ancilla inputs as in the design methodology for (n,k) reversible logical right shifter that can perform only the logical right shift operation, while the number of the garbage outputs in (n,k) reversible bidirectional logical shifter is only one more than the number of the garbage outputs of the (n,k) reversible logical right shifter. Further, the (n,k) universal reversible bidirectional barrel shifter that can perform the arithmetic, logical and rotate operations in both left and right directions has only more garbage output and ancilla input compared to the design (n,k) reversible universal right shifter. We also observed that the number of garbage outputs, the number of ancilla inputs and the quantum cost of the (n,k) reversible barrel shifters increases more rapidly by varying n and keeping k as a constant compared to the reversible barrel shifter designs in which n is kept as a constant while the values of k are varied.

REFERENCES

[1] R. Landauer. Irreversibility and heat generation in the computational process. *IBM J. Research and Development*, 5:183–191, December 1961.

[2] C.H. Bennett. Logical reversibility of computation. *IBM J. Research and Development*, 17:525–532, November 1973.

[3] H. Thapliyal and N. Ranganathan. Reversible logic-based concurrently testable latches for molecular qca. volume 9, pages 62–69, January 2010.

[4] C. Taraphdara, T. Chattopadhyay, and J.N. Roy. Machzehnder interferometer-based all-optical reversible logic gate. *Optics and Laser Technology*, 42(2):249–259, 2010.

[5] S. Kotiyal, H. Thapliyal, and N. Ranganathan. Mach-zehnder interferometer based design of all optical reversible binary adder. In *to appear Proceedings of the Design Automation and Test in Europe (DATE)*, Dresden, Germany, 2012.

[6] H. Thapliyal and N. Ranganathan. Bit conserving logic as a potential integration platform for hybrid molecular & nanoscale cmos-based architectures. In *Proc. 2009 Nanoelectronic Devices for Defense & Security (NANO-DDS) Conference*, sept. 2009.

[7] H. Thapliyal and N. Ranganathan. Concurrently testable fpga design for molecular qca using conservative reversible logic gate. In *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*, pages 1815 –1818, may 2009.

[8] H. Thapliyal and N. Ranganathan. Conservative qca gate (cqca) for designing concurrently testable molecular qca circuits. In *VLSI Design, 2009 22nd International Conference on*, pages 511 –516, jan. 2009.

[9] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information.* Cambridge Univ. Press, New York, 2000.

[10] J. Scott, L. Lee, J. Arends, and B. Moyer. Designing the low-power mcore architecture. In *Proc. Intl. Symp. on Computer Architecture Power Driven Microarchitecture Workshop*, pages 145–150, Barcelona, Spain, 1998.

[11] M.R. Pillmeier, M.J. Schulte, and E.G. Walters III. Design alternatives for barrel shifters. In *Proc. SPIE: Advanced Signal Processing Algorithms, Architectures, and Implementations XII, Vol.4791*, pages 436–447, Seattle, Washington, 2002.

[12] O. Daesun and K.K. Parhi. Area efficient controller design of barrel shifters for reconfigurable ldpc decoders. pages 240–243, May 2008.

[13] S. Das and S.P Khatri. A timing-driven approach to synthesize fast barrel shifters. *IEEE Trans. Cir.and Sys. II*, 55(1):3135, 2008.

[14] Mozammel H.A. Khan and Marek A. Perkowski. Quantum ternary parallel adder/subtractor with partially-look-ahead carry. volume 53, pages 453 – 464, 2007.

[15] Y. Takahashi. Quantum arithmetic circuits: a survey. *IEICE Trans. Fundamentals*, E92-A(5):276–1283, 2010.

[16] Y. Takahashi and N. Kunihiro. A linear-size quantum circuit for addition with no ancillary qubits. *Quantum Information and Computation*, 5(6):440448, 2005.

[17] S. Kotiyal, H. Thapliyal, and N. Ranganathan. Design of a reversible bidirectional barrel shifter. In *Proceedings of the 11th IEEE International Conference on Nanotechnology (IEEE NANO)*, Portland, Oregon, August 2011.

[18] J. A. Smolin and D. P. DiVincenzo. Five two-bit quantum gates are sufficient to implement the quantum fredkin gate. *Physical Review A*, 53:2855–2856, 1996.

[19] W.N. N. Hung, X. Song, G.Yang, J.Yang, and M. Perkowski. Optimal synthesis of multiple output boolean functions using a set of quantum gates by symbolic reachability analysis. *IEEE Trans. Computer-Aided Design*, 25(9):1652–1663, September 2006.

[20] D. Maslov and D. M. Miller. Comparison of the cost metrics for reversible and quantum logic synthesis. *http://arxiv.org/abs/quant-ph/0511008*, 2006.

[21] E. Fredkin and T Toffoli. Conservative logic. *International J. Theor. Physics*, 21:219–253, 1982.

[22] T. Toffoli. Reversible computing. Technical Report Tech memo MIT/LCS/TM-151, MIT Lab for Computer Science, 1980.

[23] A. Peres. Reversible logic and quantum computers. *Phys. Rev. A, Gen. Phys.*, 32(6):3266–3276, December 1985.

[24] James Donald and Niraj K. Jha. Reversible logic synthesis with fredkin and peres gates. *J. Emerg. Technol. Comput. Syst.*, 4:2:1–2:19, April 2008.

[25] Oleg Golubitsky and Dmitri Maslov. A study of optimal 4-bit reversible toffoli circuits and their synthesis. *IEEE Transactions on Computers*, 99(PrePrints), 2011.

[26] Oleg Golubitsky, Sean M. Falconer, and Dmitri Maslov. Synthesis of the optimal 4-bit reversible circuits. In *Proceedings of the 47th Design Automation Conference*, DAC '10, pages 653–656, New York, NY, USA, 2010. ACM.

[27] D. Maslov, S.M. Falconer, and M. Mosca. Quantum circuit placement. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(4):752 –763, april 2008.

[28] D. Maslov, G. W. Dueck, and D. M. Miller. Techniques for the synthesis of reversible toffoli networks. *ACM Trans. Des. Autom. Electron. Syst.*, 12, September 2007.

[29] P. Gupta, A. Agarwal, and N. K. Jha. An algorithm for synthesis of reversible logic ciruits. *IEEE Trans. Computer-Aided Design*, 25(11):2317–2330, Nov 2006.

[30] V. V. Shende, A.K. Prasad, I.L. Markov, and J.P. Hayes. Synthesis of reversible logic circuits. *IEEE Trans. on CAD*, 22:710–722, 2003.

[31] D. Maslov and G. W. Dueck. Reversible cascades with minimal garbage. *IEEE Trans. Computer-Aided Design*, 23(11):1497–1509, November 2004.

[32] G.Yang, X. Song, W.N. N. Hung, and M.A. Perkowski. Bi-directional synthesis of 4-bit reversible circuits. *Computer Journal*, 51(2):207–215, March 2008.

[33] Maher Hawash, Marek Perkowski, Steve Bleiler, John Caughman, and Amjad Hawash. Reversible function synthesis of large reversible functions with no ancillary bits using covering set partitions. *Information Technology: New Generations, Third International Conference on*, 0:1008–1013, 2011.

[34] A. K. Prasad, V.V. Shende, I.L. Markov, J.P. Hayes, and K. N. Patel. Data structures and algorithms for simplifying reversible circuits. *ACM JETC*, 2(4):277–293, 2006.

[35] R. Wille, O. Keszo andcze, and R. Drechsler. Determining the minimal number of lines for large reversible circuits. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, pages 1 –4, march 2011.

[36] D. Große, R. Wille, G. W. Dueck, and R. Drechsler. Exact synthesis of elementary quantum gate circuits for reversible functions with dont cares. In *Proc. of the Intl Symp. on Multi-Valued Logic*, pages 214–219, Dallas, Texas, May 2008.

[37] D. Große, R. Wille, G.W. Dueck, and R. Drechsler. Exact multiple control toffoli network synthesis with sat techniques. *IEEE Trans. on CAD*, 28(5):703715, 2009.

[38] D. Maslov and M. Saeedi. Reversible circuit optimization via leaving the boolean domain. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 30(6):806 –816, june 2011.

[39] H. Thapliyal, N. Ranganathan, and R. Ferreira. Design of a comparator tree based on reversible logic. In *Nanotechnology (IEEE-NANO), 2010 10th IEEE Conference on*, pages 1113 –1116, aug. 2010.

[40] H. Thapliyal and N. Ranganathan. Reversible logic based concurrent error detection methodology for emerging nanocircuits. In *Nanotechnology (IEEE-NANO), 2010 10th IEEE Conference on*, pages 217 –222, aug. 2010.

[41] M. Morrison and N. Ranganathan. Design of a reversible alu based on novel programmable reversible logic gate structures. In *VLSI (ISVLSI), 2011 IEEE Computer Society Annual Symposium on*, pages 126 –131, july 2011.

[42] M. Nachtigal and N. Ranganathan. Design and analysis of a novel reversible encoder/decoder. In *Nanotechnology (IEEE-NANO), 2011 11th IEEE Conference on*, pages 1543 –1546, aug. 2011.

[43] J. E. Rice. An introduction to reversible latches. *Comput. J.*, 51(6):700–709, 2008.

[44] Min-Lun Chuang and Chun-Yao Wang. Synthesis of reversible sequential elements. *J. Emerg. Technol. Comput. Syst.*, 3(4):1–19, 2008.

[45] S.K.Sastry, H.S.Shroff, S. N. Mahammad, and V. Kamakoti. Efficient building blocks for reversible sequential circuit design. In *Proc. the 49th IEEE Intl. l Midwest Symp.on Cir. and Sys.*, pages 437–441, Puerto Rico, August 2006.

[46] H. Thapliyal and N. Ranganathan. Design of reversible latches optimized for quantum cost, delay and garbage outputs. In *VLSI Design, 2010. VLSID '10. 23rd International Conference on*, pages 235 –240, jan. 2010.

[47] H. Thapliyal and N. Ranganathan. Design of reversible sequential circuits optimizing quantum cost, delay and garbage outputs. *ACM Journal of Emerging Technologies in Computing Systems*, 6(4, Article 14):14:1–14:35, December 2010.

[48] H. Thapliyal and N. Ranganathan. Design of efficient reversible binary subtractors based on a new reversible gate. In *VLSI, 2009. ISVLSI '09. IEEE Computer Society Annual Symposium on*, pages 229 –234, may 2009.

[49] Bart Desoete and A. De Vos. A reversible carry-look-ahead adder using control gates. *Integration, the VLSI Journal*, 33(1):89 – 104, 2002.

[50] M. Haghparast, S.J. Jassbi, K. Navi, and O.Hashemipour. Design of a novel reversible multiplier circuit using hng gate in nanotechnology. *World App. Sci. J.*, 3(6):974–978, 2008.

[51] Ashis Kumer Biswas, Md. Mahmudul Hasan, Ahsan Raja Chowdhury, and Hafiz Md. Hasan Babu. Efficient approaches for designing reversible binary coded decimal adders. *Microelectron. J.*, 39(12):1693–1703, 2008.

[52] J. W. Bruce, M. A. Thornton, L. Shivakumaraiah, P. S. Kokate, and X. Li. Efficient adder circuits based on a conservative reversible logic gate. In *Proc. IEEE Symposium on VLSI, 2002*, pages 83–88, 2002.

[53] H. Thapliyal and N. Ranganathan. A new design of the reversible subtractor circuit. In *Nanotechnology (IEEE-NANO), 2011 11th IEEE Conference on*, pages 1430 –1435, aug. 2011.

[54] M.H.A. Khan. Design of full-adder with reversible gates. In *Proc. International Conference on Computer and Information Technology*, pages 515–519, 2002.

[55] H. Thapliyal and N. Ranganathan. Design of efficient reversible logic based binary and bcd adder circuits. *To appear ACM Journal of Emerging Technologies in Computing Systems*, 2011.

[56] M. Nachtigal, H. Thapliyal, and N. Ranganathan. Design of a reversible single precision floating point multiplier based on operand decomposition. In *Nanotechnology (IEEE-NANO), 2010 10th IEEE Conference on*, pages 233 –237, aug. 2010.

[57] M. Nachtigal, H. Thapliyal, and N. Ranganathan. Design of a reversible floating-point adder architecture. In *Nanotechnology (IEEE-NANO), 2011 11th IEEE Conference on*, pages 451 –456, aug. 2011.

[58] M. Morrison, M. Lewandowski, R. Meana, and N. Ranganathan. Design of a novel reversible alu using an enhanced carry look- ahead adder. In *Nanotechnology (IEEE-NANO), 2011 11th IEEE Conference on*, pages 1436 –1440, aug. 2011.

[59] M. Morrison and N. Ranganathan. Design of a moore finite state machine using a novel reversible logic gate, decoder and synchronous up-counter. In *Nanotechnology (IEEE-NANO), 2011 11th IEEE Conference on*, pages 1445 –1449, aug. 2011.

[60] M. Morrison, M. Lewandowski, R. Meana, and N. Ranganathan. Design of static and dynamic ram arrays using a novel reversible logic gate and decoder. In *Nanotechnology (IEEE-NANO), 2011 11th IEEE Conference on*, pages 417 –420, aug. 2011.

[61] M. H. A. Khan and M. A. Perkowski. Quantum ternary parallel adder/subtractor with partially-look-ahead carry. *J. Systems Architecture*, 53(7):453–464, 2007.

[62] S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. P. Moulton. A new quantum ripple-carry addition circuit. *http://arXiv.org/quant-ph/0410184*, Oct 2004.

[63] Y. Takahashi, S. Tani, and N. Kunihiro. Quantum addition circuits and unbounded fan-out. *http://arxiv.org/abs/0910.2530*, Oct 2009.

[64] A. Trisetyarso and R. V. Meter. Circuit design for a measurement-based quantum carry-lookahead adder. *http://arxiv.org/abs/0903.0748*, 2009.

[65] R.V. Meter, W.J. Munro, K. Nemoto, and K. M. Itoh. Arithmetic on a distributed-memory quantum multicomputer. *http://arxiv.org/abs/quant-ph/0607160*, 2009.

[66] S.Gorgin and A. Kaivani. Reversible barrel shifters. In *Proc. 2007 Intl. Conf. on Computer Systems and Applications*, pages 479–483, Amman, May 2007.

[67] I. Hashmi and H.M.H. Babu. An efficient design of a reversible barrel shifter. In *VLSI Design, 2010. VLSID '10. 23rd International Conference on*, pages 93 –98, Jan 2010.

[68] S. Kotiyal, H. Thapliyal, and N. Ranganathan. Design of a ternary barrel shifter using multiple-valued reversible logic. In *Proceedings of the 11th IEEE International Conference on Nan-otechnology (IEEE NANO)*, pages 1104–1108, Seoul, Korea, August 2010.