2015

Explorations for Efficient Reversible Barrel Shifters and Their Mappings in QCA Nanocomputing

Ke Chen
University of Kentucky, kch246@g.uky.edu

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained needed written permission statement(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine) which will be submitted to UKnowledge as Additional File.

I hereby grant to The University of Kentucky and its agents the irrevocable, non-exclusive, and royalty-free license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless an embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's thesis including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

Ke Chen, Student

Dr. Himanshu Thapliyal, Major Professor

Dr. Caicheng Lu, Director of Graduate Studies

EXPLORATIONS FOR EFFICIENT REVERSIBLE BARREL SHIFTERS AND THEIR MAPPINGS IN QCA NANOCOMPUTING

THESIS

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science in Electrical Engineering
in the College of Engineering
at the University of Kentucky

By
Ke Chen
University of Kentucky
Lexington, Kentucky
Director: Dr. Himanshu Thapliyal, Professor of Electrical and Computer Engineering
University of Kentucky
Lexington, Kentucky

2015

ABSTRACT OF THESIS

EXPLORATIONS FOR EFFICIENT REVERSIBLE BARREL SHIFTERS AND THEIR MAPPINGS IN QCA NANOCOMPUTING

This thesis is based on promising computing paradigm of reversible logic which generates unique outputs out of the inputs and. Reversible logic circuits maintain one-to-one mapping inside of the inputs and the outputs. Compared to the traditional irreversible computation, reversible logic circuit has the advantage that it successfully avoids the information loss during computations. Also, reversible logic is useful to design ultra-low-power nanocomputing circuits, circuits for quantum computing, and the nanocircuits that are testable in nature. Reversible computing circuits require the ancilla inputs and the garbage outputs. Ancilla input is the constant input in reversible circuits. Garbage output is the output for maintaining the reversibility of the reversible logic but is not any of the primary inputs nor a useful bit. An efficient reversible circuit will have the minimal number of garbage and ancilla bits.

Barrel shifter is one of main computing systems having applications in high speed digital signal processing, floating-point arithmetic, FPGA, and Center Processing Unit (CPU). It can operate the function of shifting or rotation for multiple bits in only one clock cycle. The goal of this thesis is to design barrel shifters based on the reversible computing that are optimized in terms of the number of ancilla and garbage bits. In order to achieve this goal, a new Super Conservative Reversible Logic Gate (SCRL gate) has been used. The SCRL gate has 1 control input depending on the value of which it can swap any two n-1 data inputs. We proved that the SCRL gate is superior to the existing conservative reversible Fredkin gate. This thesis develops 5 design methodologies for reversible barrel shifters using SCRL gates that are primarily optimized with the criteria of the number of ancilla and garbage bits. The five proposed methodologies consist of reversible right rotator, reversible logical right shifter, reversible arithmetic right shifter, reversible universal right shifter and reversible universal bidirectional shifter. The proposed reversible barrel shifter design is compared with the existing works in literature and have shown improvement ranging from 8.5% to 92% by the number of garbage and ancilla bits. The SCRL gate and design methodologies of reversible barrel shifter are mapped in Quantum Dot Cellular Automata (QCA) computing. It is illustrated that the SCRL-based designs of reversible barrel shifters have less QCA cost (cost in terms of number of inverters and majority voters) compared to the Fredkin gate-based designs of reversible barrel shifters.

KEYWORDS: Reversible logic, SCRL gates, QCA computing, Barrel shifter

Ke Chen

May 17, 2015

EXPLORATIONS FOR EFFICIENT REVERSIBLE BARREL SHIFTERS AND THEIR MAPPINGS QCA NANOCOMPUTING

By

Ke Chen

Himanshu Thapliyal

(Director of Thesis)
Caicheng Lu

(Director of Graduate Studies)
May 18, 2015

(Date)

ACKNOWLEDGMENTS

I would like to thank my thesis director Dr. Himanshu Thapliyal for all his guidance and support. And I would like to thank my parents for their support throughout my life. Also I would like to thank Jie Chen for her love.

Table of Contents

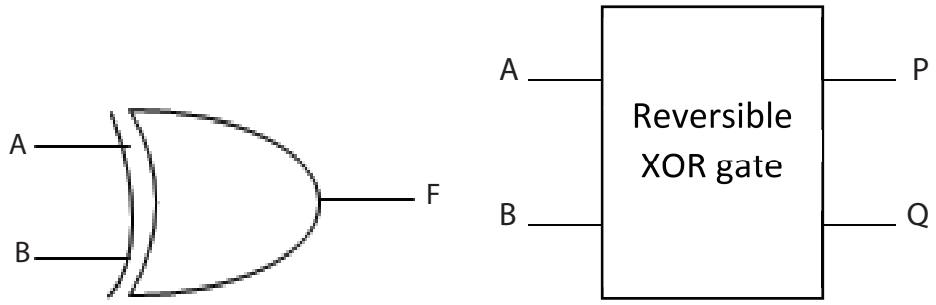
List of Figures

List of Tables

Chapter 1

Introduction

Reversible logic is a promising technology that can be applied in a wide technology domain such as quantum computing, optical computing, molecular computing and DNA computing, etc. [41]. Reversible logic makes it possible for no information loss during computations. Reversible logic circuits have one-to-one mapping between the inputs and the outputs, and yield unique output out of the inputs. Researchers have proved that in irreversible logic computations, the information loss of 1 bit causes $KT \ln 2$ Joules of energy dissipation [20]. Bennett proved that this $KT \ln 2$ J of energy dissipation will not appear in reversible logic [3]. The inputs of reversible logic can be derived from the outputs while this is impossible in irreversible logic circuits since the inputs and outputs are not uniquely corresponded. Let's take an example of classical irreversible XOR gates. Figure 1.1(a) shows the irreversible XOR gate and its truth table is shown in Table 1.1. As we can observe from Table 1.1, it is impossible to derive the inputs from the outputs since there are two possible inputs $AB = (00, 11)$ that can yield the output 0. However, a reversible XOR gate shown in Figure 1.1(b) is an example of one-to-one mapping. As shown in Table 1.2, each output is generated from a unique input. $F = A \oplus B$, $P = A$, $Q = A \oplus B$.



(a) Irreversible XOR gate (b) Reversible XOR gate

Figure 1.1: Irreversible and reversible XOR gates

Table 1.1: Truth table for irreversible XOR gate

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

Table 1.2: Truth table for reversible XOR gate

A	B	P	Q
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

1.1 Motivation

Reversible logic successfully avoid the energy loss during circuits computations. Barrel shifter is one of main computing systems having applications in high speed digital signal processing, floating-point arithmetic, FPGA, and Center Processing Unit (CPU)[29]. Barrel shifter can operate the function of shifting and rotation for any possible bits within one clock cycle. In the existing literature, researchers have proposed several designs of barrel shifter. The existing reversible barrel shifter designs have the limitations that they have unexpected amount of ancilla and garbage bits [17, 19, 26]. Ancilla input is the constant input in the reversible circuit. Garbage output is only used for maintaining the reversibility but is not primary nor useful in circuits. Reversible circuits generate a lot of bits of garbage outputs to maintain the reversibility. The increasing number of garbage outputs have several penalties such as increasing circuits size, extra interconnections, more heat generation and increasing number of I/O pins. Another significant reason for minimizing the garbage and ancilla bits is that the several emerging technologies having application of reversible logic are in premature stage. Hence large reversible circuits in emerging technologies are difficult to fabricate[34]. The existing reversible barrel shifters were entirely constructed from the Fredkin and Feynman gates. The use of the Fredkin gate to design the barrel shifter was the cause of enormous amount of garbage bits. In this thesis, we have investigated the application of super consevative reversible logic gate (SCRL) to minimize the ancilla and garbage bits in the design of barrel shifters. Because of the several advantages of QCA technology, the SCRL gate and corresponding reversible barrel shifter designs have been mapped in field coupled QCA computing.

1.2 Contribution of Thesis

This thesis develops 5 design methodologies for reversible barrel shifters using SCRL gates that are primarily optimized by reducing the amount of ancilla and garbage bits. The proposed design of reversible barrel shifters are compared with the existing works in

literature and have shown improvement ranging from 8.5% to 92% in terms of number of ancilla inputs and the garbage outputs. The design methodologies of reversible barrel shifter are mapping in field coupled QCA computing. The contributions can be summarized as follows:

1. Reversible right rotator using SCRL gates
2. Reversible logical right shifter using SCRL gates
3. Reversible arithmetic right shifter using SCRL gates
4. Reversible universal right shfiter using SCRL gates
5. Reversible universal bidirectional shifter using SCRL gates
6. Mapping of SCRL gate in field coupled QCA computing
7. Design evaluation of reversible barrel shifters in QCA computing

1.3 Outline of Thesis

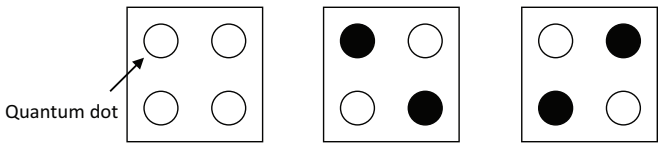
Chapter 2 is the background of reversible logic and QCA computing. Some common and widely used reversible logic gates are also introduced in Chapter 2. Chapter 3 briefly explains the main theory of the SCRL gate and how it can be realized using QCA technology. Chapter 4 introduces the main theory of barrel shifters. The 5 design methodologies of barrel shifters are proposed in this chapter. Section 4.1 introduces the design methodology for right rotators; Section 4.2 introduces the design methodology for logical right shifters; Section 4.3 introduces the design methodology for arithmetic right shifter; Section 4.4 introduces the design methodology for universal right shifters; Section 4.5 introduces the design methodology for universal bidirectional shifters. Chapter 5 shows the evaluations of the proposed methodologies by the amount of garbage and ancilla bits and the devices used in QCA implementation. Chapter 6 is a conclusion of this thesis.

Chapter 2

Background and Related Work

2.1 Basics of QCA Computing

Quantum dots cellular automata computing (QCA computing) uses QCA cells which can be considered as a box with 4 dots at the four vertices of the box (shown as Figure 2.1(a)). An electron pair is brought into the box and placed in two of the quantum dots to form different states of circuits [22, 27]. Due to electrostatic repulsion, the two electrons of the electron pair would automatically occupy locations to be far from each other, which are the two diagonals. By providing tunneling junctions with potential barriers, the electrons are prevented from escaping from the cell. In this way, a QCA cell forms two different states representing logic 0 and 1. In QCA logic, the two states are represented as $P = 1$ and $P = -1$ as shown in Figure 2.1(b) and Figure 2.1(c).



(a) QCA cell (b) Logic 0 (c) Logic 1

Figure 2.1: QCA cells and states

Due to to electrostatic interactions a cell will always keep the same state as the one next to it when arranged in linear array as shown in Figure 2.3(a), [27]. This is called binary wire. Hence, the signal is propagated and the information is transmitted from the input to the output.
Figure 2.2 shows a 45 degree QCA cell. When several this QCA cells are aligned in an array as shown in Figure 2.3(b), each cell has the opposite state of its neighbors. This propagation of information is called inverter chain. A binary wire and inverter chain can cross each other without interaction in crossing wires(Figure 2.3(c)).

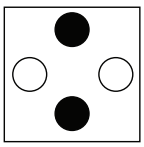
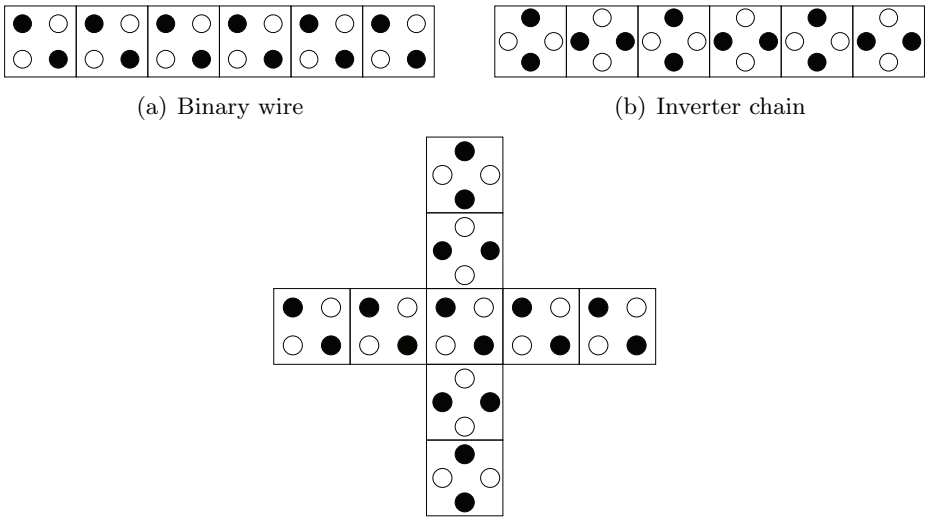


Figure 2.2: 45 degree QCA cell



(a) Binary wire

(b) Inverter chain

(c) Crossing connection

Figure 2.3: QCA wires

1. Majority gate.

   Majority gate is one of the frequently used gates in QCA designs. As shown in Figure 2.4, majority gate can be considered as a crossing wire of same type of cells (binary wire) with five QCA cells. It has the equation as follows:

$$F = AB + AC + BC \tag{2.1}$$

   Table 2.1 shows the truth table of majority gate.

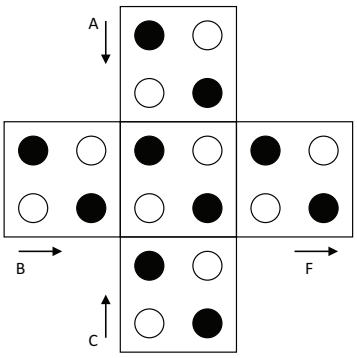


Figure 2.4: Majority gate

Table 2.1: Truth table for majority gate

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

If one of the inputs is set to -1, which is logic 0, for example C is set to logic 0, a majority gate will then turn into an AND gate. Since:

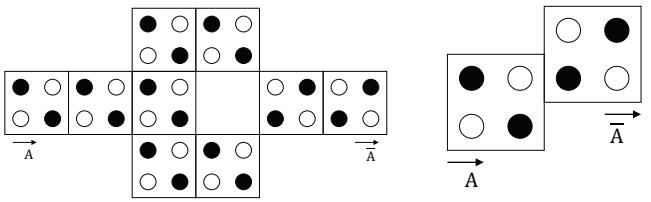
$$F = AB + A \cdot 0 + B \cdot 0 = AB \tag{2.2}$$

In another case, if C is set to logic 1, it will then become an OR gate. Since:

$$F = AB + A \cdot 1 + B \cdot 1 = A(B + 1) + B = A + B \tag{2.3}$$

2. Inverters.
   There are two kinds of inverter designs as shown in Figure 2.5(a) and Figure 2.5(b). The first inverter is more stable in nature than the second one. But the second one is more frequently used in QCA computing because it is simpler and saves cells.

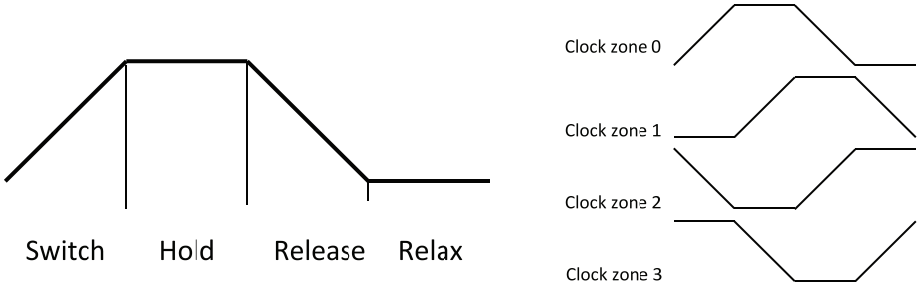
QCA clock zones contribute to successively transmitting the information from one clock zone to the next [14]. The information flows in a pipeline from the input to the output. Figure 2.6(a) shows the different functions of a cell in 4 different clock zones. A cell in a certain clock accepts the information from the cell in previous clock zone upon a switch excitation. After holding the information for a quarter clock cycle, this cell releases the



(a) Inverter (b) Inverter

Figure 2.5: QCA inverter designs

current information to clear the memory for next information. Then it relaxes for another quarter clock cycle. The procedure is shown as Figure 2.6(b).



(a) Clock zones (b) Four phase clocking

Figure 2.6: Information flow process

2.2 Reversible Logic Gates and Their QCA Implementations

2.2.1 Feynman Gate

Feynman gate (FE) is a 2 by 2 reversible logic gate with the outputs $P = A, Q = A \oplus B$ [41]. The block diagram is shown in Figure 2.7. Table 2.2 is the truth table for Feynman gate.

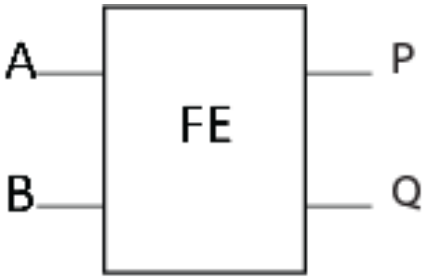


Figure 2.7: Feynman gate

Table 2.2: Truth table for Feynman gate

A	B	P	Q
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Feynman gates are frequently used for data copying while avoiding fanouts. When $B = 0$, the Feynman gate performs an operation of copying since $P = A$ and $Q = A \oplus 0 = A$. Figure 2.8 shows the QCA design of Feynman gate in four-phase clocking scheme.

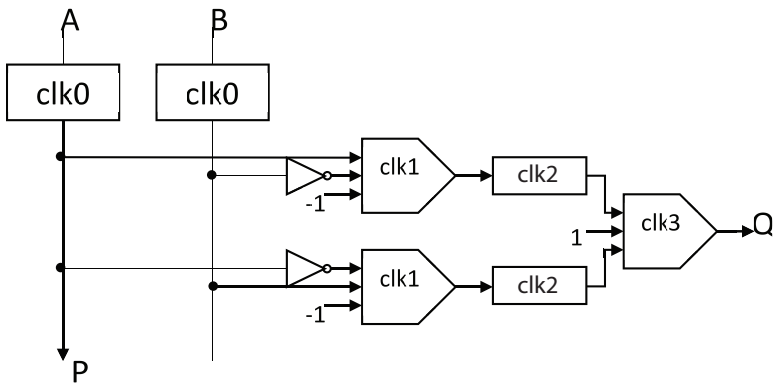


Figure 2.8: QCA design of Feynman gate

"clk0" means that the current wire is set to clock zone 0. Similarly "clk1, clk2, clk3" means that the current wire is set to clock zone 1, 2 and 3 respectively. The triangle represents an inverter. The pentagon represents a majority voter. When one input in a majority voter is set to -1 which represents a logic 0, the majority voter performs as an AND gate and when one of the inputs of the majority voter is set to 1, the majority voter performs as an OR gate. The QCA implementation is shown in Figure 2.9

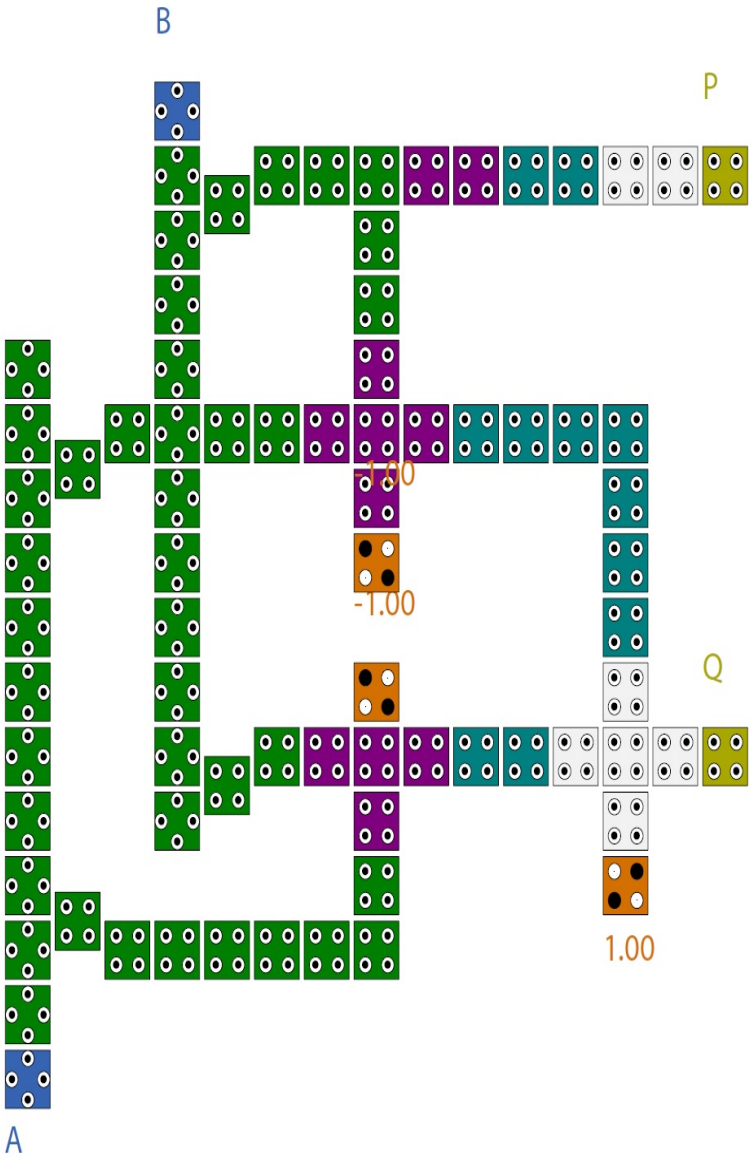


Figure 2.9: Feynman gate QCA implementation

2.2.2 Fredkin Gate

Fredkin gate (FR) is a 3 by 3 logic gate. It has the function of $P = A$, $Q = \overline{A}B + AC$, $R = \overline{A}C + AB$ [41]. The block diagram is shown in Figure 2.10, and the truth table is shown in Table 2.3.

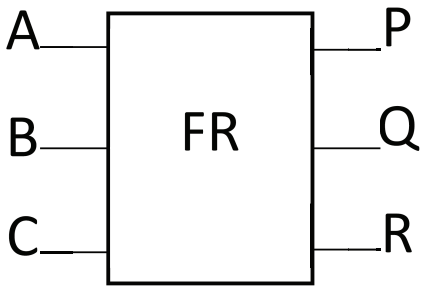


Figure 2.10: Fredkin gate

Table 2.3: Truth table of Fredkin gate

A	B	C	P	Q	R
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	0
1	1	1	1	1	1

As can be observed in Table 2.3, the input A is a control signal of the Fredkin gate. When $A = 0$, the Fredkin gate performs an operation of swapping data of B and C. When $A = 1$, the outputs are the same as the inputs. This is similar to a $2 : 1$ multiplexer. Figure 2.11 shows the QCA design of Fredkin gate. The QCA implementation is shown in Figure 2.12.

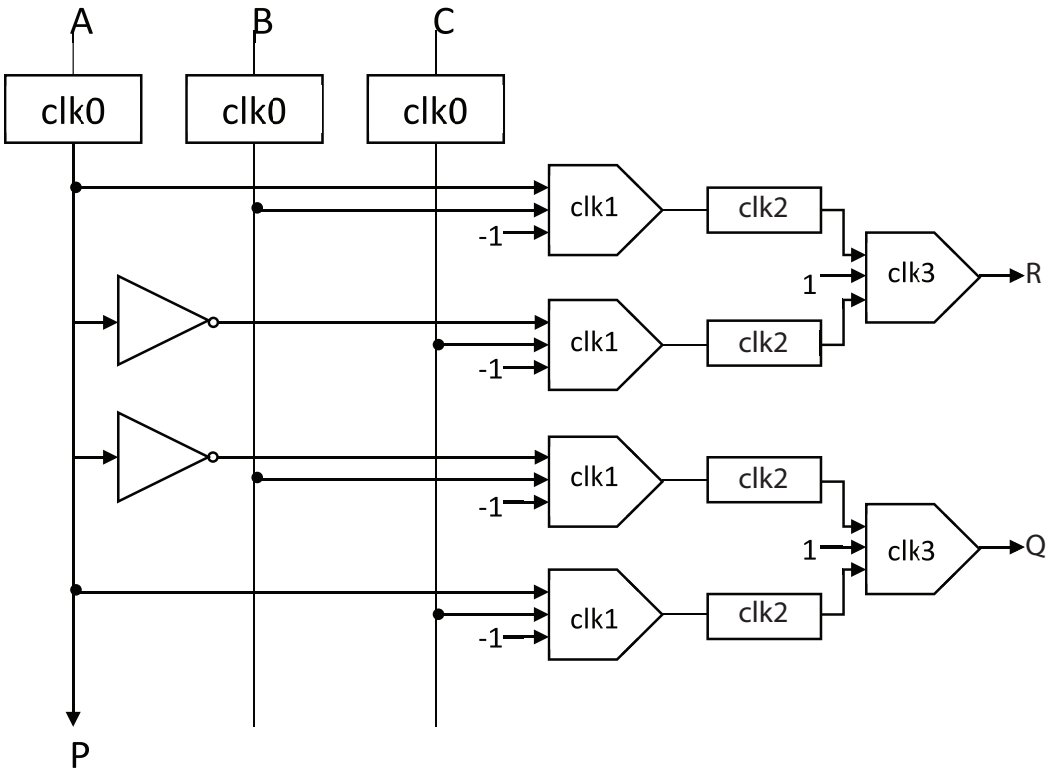


Figure 2.11: QCA design of Fredkin gate

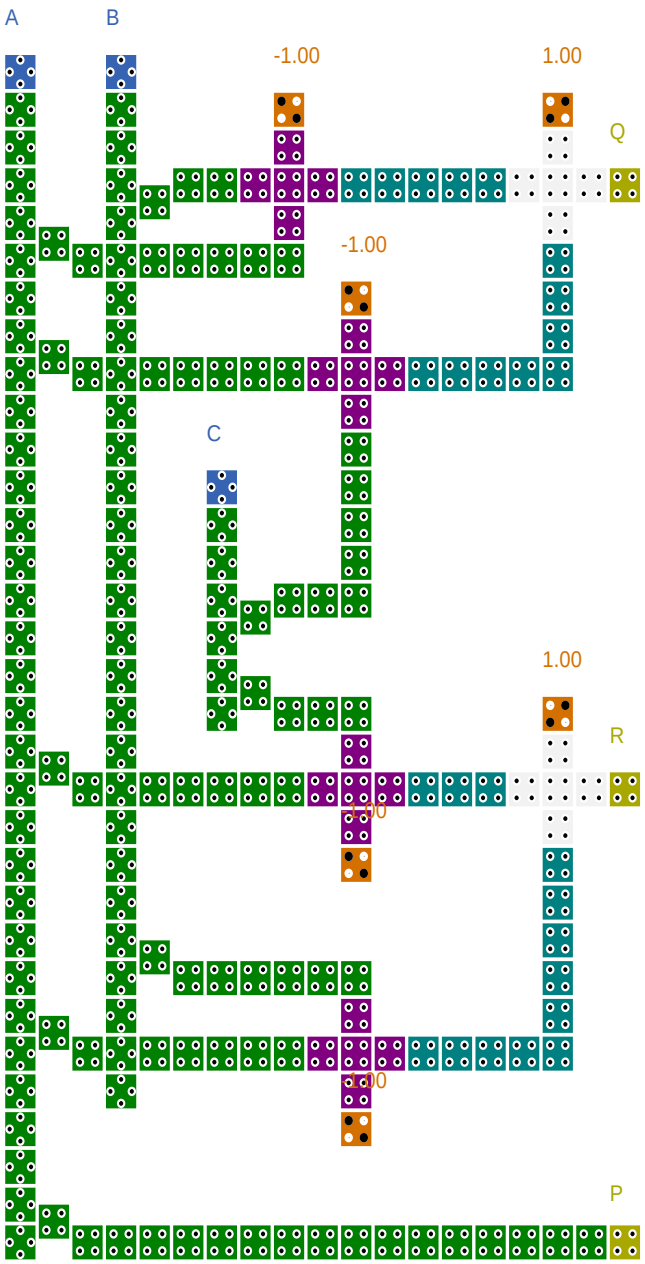


Figure 2.12: Fredkin gate QCA implementation

2.3 Barrel Shifter

Barrel shifter is a functional logic circuit with any designed inputs and same number of outputs [29]. The function of a barrel shifter is to shift the inputs information in a certain order to left or right for any possible bits. A well designed barrel shifter is able to implement bidirectional information shifting. Researchers have proposed various barrel shifter designs. Among these designs, logarithmic barrel shifter is one of the most popular designs since it saves area for the circuit and it is simple to implement. As it is shown in Figure 2.13 [36], a barrel shifter with q stages and n inputs is called an (n, q) barrel shifter. Microscopically into each stage, the p^{th} stage shifts the information for 2^{q-p} bits where $q = \log_2 n$. With a control signal which is used for either executing or shutting down the shifting mode. The shifting mode can be executed when the control signal is at logic 1 and can be shut down when the control signal is at logic 0. Hence, the information can be shifted from 0 bit (all control signals set to logic 0) to $n - 1$ bits (all control signals set to logic 1). Logical right shifter, logical left shifter, right rotator, left rotator, arithmetic left shifter and arithmetic right shifter are the different functions that can be implemented using the barrel shifter. Suppose there is an $(8, 3)$ barrel shifter with the inputs as $a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0$, the six functions are shown in Table 2.4 [19].

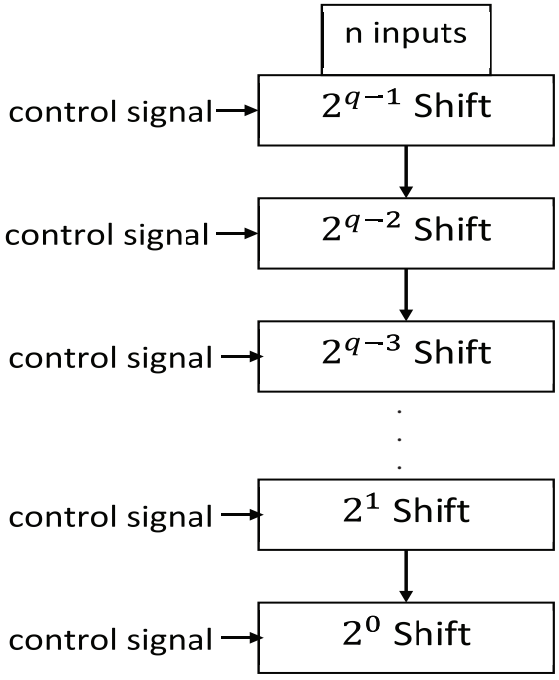


Figure 2.13: (n, k) barrel shifter block diagram

Table 2.4: Operation functions and outputs

Operation functions	Outputs
Logical right shifer	$0, 0, 0, a_7, a_6, a_5, a_4, a_3$
Right rotator	$a_2, a_1, a_0, a_7, a_6, a_5, a_4, a_3$
Logical left shifter	$a_4, a_3, a_2, a_1, a_0, 0, 0, 0$
Left rotator	$a_4, a_3, a_2, a_1, a_0, a_7, a_6, a_5$
Arithmetic right shifter	$a_7, a_7, a_7, a_7, a_6, a_5, a_4, a_3$
Arithmetic left shifter	$a_7, a_3, a_2, a_1, a_0, 0, 0, 0$

1. Logical right/left shifters: A logical right/left shifter is functioned to make any required bits of displacement of the inputs while change the same bits of the first or last bits to 0. Table 2.4 shows an example for 3 bits shifting of an 8-inputs barrel shifters. After a 3 bits logical right shifting, the outputs are changed to a serial data of $0, 0, 0, a_7, a_6, a_5, a_4, a_3$. The inputs are right translated for 3 bits while the first 3 bits are changed to 0. One the contrary, after a 3 bits logical left shifting, the outputs are changed to $a_4, a_3, a_2, a_1, a_0, 0, 0, 0$, which makes the inputs left translated for 3 bits while change the last 3 bits to 0.

2. Right/left rotators: Compared to logical shifters, rotators generally do the function of displacement. After a 3 bits right rotation, the outputs are changed to $a_2, a_1, a_0, a_7, a_6, a_5, a_4, a_3$. The last 3 bits are rotated into the front. After a 3 bits left rotation, the outputs are changed to $a_4, a_3, a_2, a_1, a_0, a_7, a_6, a_5$. The first 3 bits are rotated to the back of the serial data.

3. Arithmetic shifters: Different from logical shifters, arithmetic right shifters always keep the leftmost shifting bits as same as the sign bit, which is i_7 in this example. Arithmetic left shifters always keep the leftmost 1 bit as sign bit, while the remaining bits are doing the function of inputs logical left shifting.

2.4 Related Work

Reversible logic has attracted many meaningful attempts in proposed works such as [10–12, 15, 23, 24, 44]. Minimizing the garbage outputs and ancilla inputs is the common goal for the researchers to optimize the reversible logic circuits designs. We observed that some new technologies have been proposed and have contributed to reducing the ancilla inputs and garbage outputs [25, 31, 37, 40, 40, 43]. Some of the practical designs are flip-flops, fast fourier transform, perfect shuffle, etc [2, 7]. Some researches also included binary calculations such as half adders, full adders, ripple carry adders, carry look-ahead adder, etc.[6, 9, 16, 28, 38, 39]. Researchers have proposed new designs of ripple carry adder without ancilla inputs and optimized the operation delay in [34, 35]. We also observed several recent designs of reversible barrel shifters[17, 19, 26]. Among these designs, Mitra design [26] is the newest which was proposed in March 2015. However, there is still space to improve these designs. This thesis has done the work to optimize the barrel shifter designs including right rotator, logical right shifter, arithmetic right shifter, universal right shifter and unvisal bidirectional shifter by reducing the number of ancilla inputs and garbage outputs. What's more, many of the current barrel shifter designs focused on the universal

bidirectional shifters. This thesis work covers all 6 functions of the barrel shifters (right/left rotator, logical right/left shifter and arithmetic right/left shifter)and optimized them in a satisfying range compared to the most recent design. Also, a combined circuit that can implement all the six functions has been proposed in this thesis work.

Chapter 3

Super Conservative Reversible Logic Gate

3.1 Introduction to SCRL Gate

As we have discussed in previous chapter (2.2), Fredkin gate is a 3 by 3 reversible logic gate with one control input, two data inputs and three outputs. Recently, researchers have proposed a new $n \times n$ reversible logic gate named Super Conservative Reversible Logic Gate (SCRL gate). The SCRL gate has n inputs that includes one control bit input and $n-1$ data inputs [36]. An $n \times n$ SCRL gate is shown in Figure 3.1.

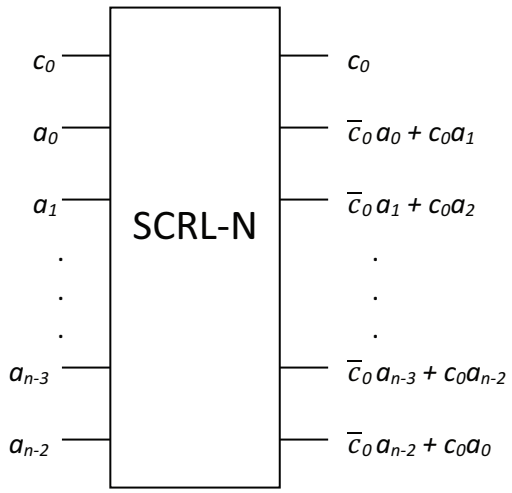


Figure 3.1: $n \times n$ SCRL gate

Define the n inputs of SCRL gate as $c_0, a_0, a_1, a_2, ...a_{n-2}$, where c_0 is the control bit input and a_0 to a_{n-2} are the normal inputs. The SCRL gate outputs are defined as $o_0, o_1, o_2, ...o_{n-1}$. The outputs function can be defined as:

$$o_0 = c_0 \tag{3.1}$$

$$o_i = \overline{c}_0 a_i + c_0 a_j \tag{3.2}$$

$$o_{n-1} = \overline{c}_0 a_{n-2} + c_0 a_0 \tag{3.3}$$

where a_i is the i^{th} input, a_j is the j^{th} input, $i \leq n-3$, $j = i+1$. The outputs depend on the control input c_0. When $c_0 = 0$, the outputs are $c_0, a_0, a_1, a_2 ... a_{n-2}$. When $c_0 = 1$, the outputs are $c_0, a_1, a_2 ... a_{n-2}, a_0$. However, the outputs function is not unique and there can be more than one version of the SCRL gates. The diversity of the output functions in SCRL gate allows it to implement different functions in reversible computing. Under a special condition where $n = 3$, the 3×3 SCRL gate is similar to Fredkin gate. However, the 3×3 Fredkin gate is functionally limited compared to the generalized $n \times n$ SCRL gate.

3.2 SCRL Gate Implementation in Field Coupled Nanocomputing

QCA computing is ultra low power in nature. It is an ideal technology to implement the SCRL gate. A QCA design of SCRL-n gate is shown in Figure 3.2. Table 3.1 is the truth table for a 4×4 SCRL gate. The function equations are:

$$o_0 = c_0 \tag{3.4}$$
$$o_1 = \overline{c}_0 a_0 + c_0 a_1 \tag{3.5}$$
$$o_2 = \overline{c}_0 a_1 + c_0 a_2 \tag{3.6}$$
$$o_3 = \overline{c}_0 a_2 + c_0 a_0 \tag{3.7}$$

If $c_0 = 0$, the outputs are c_0, a_0, a_1, a_2. Else if $c_0 = 1$, the outputs are c_0, a_1, a_2, a_0. This function is called barrel rotator, which will be discussed in the next chapter. A 4×4 SCRL gate is shown in Figure 3.3. Figure 3.4 is a QCA layout of 4×4 SCRL gate.

Table 3.1: Truth table for 4×4 SCRL gate

c_0	a_0	a_1	a_2	o_0	o_1	o_2	o_3
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	1
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	1
0	1	1	0	0	1	1	0
0	1	1	1	0	1	1	1
1	0	0	0	1	0	0	0
1	0	0	1	1	0	1	0
1	0	1	0	1	1	0	0
1	0	1	1	1	1	1	0
1	1	0	0	1	0	0	1
1	1	0	1	1	0	1	1
1	1	1	0	1	1	0	1
1	1	1	1	1	1	1	1

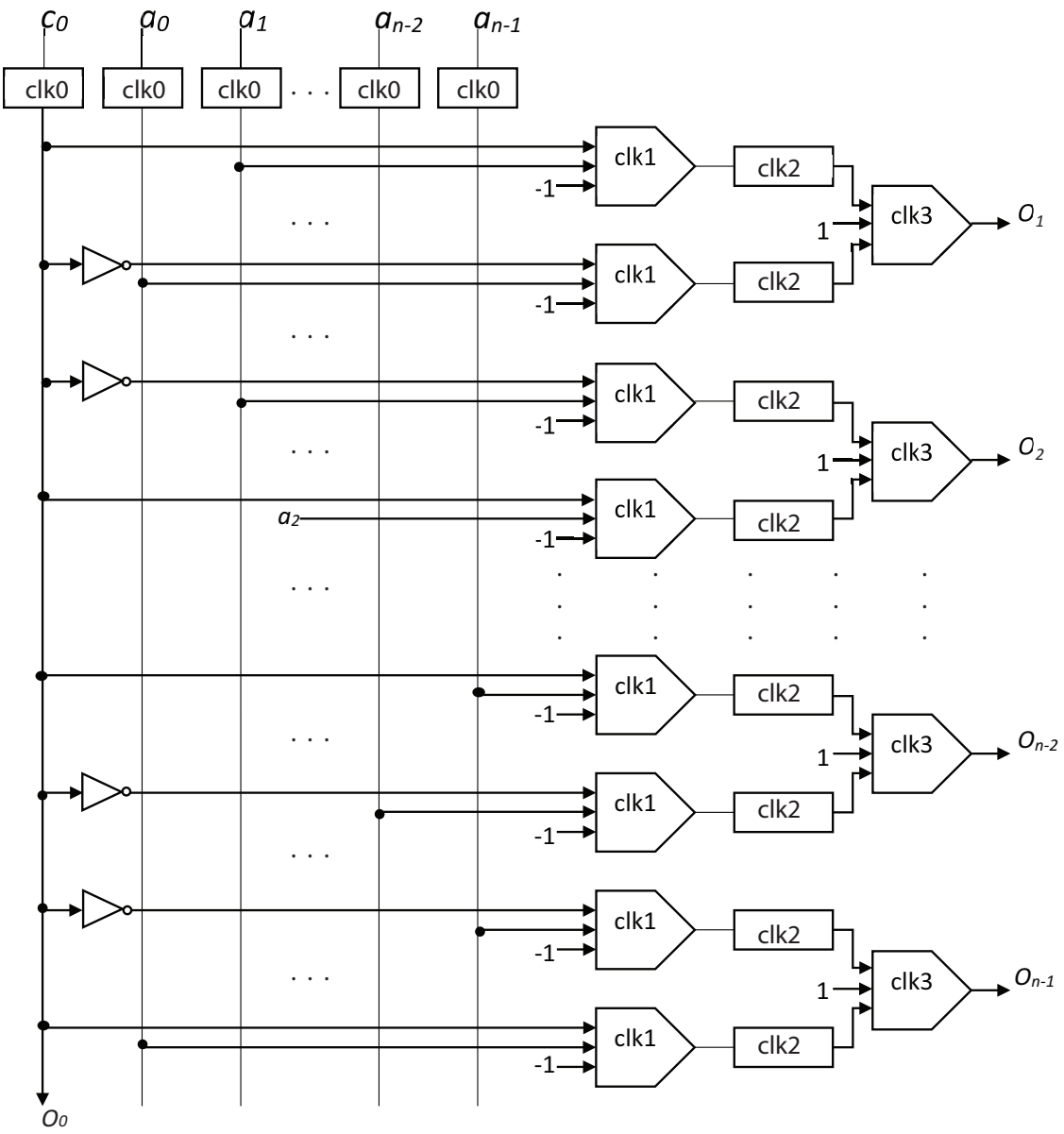


Figure 3.2: QCA design of SCRL-n gate in four-phase clocking scheme

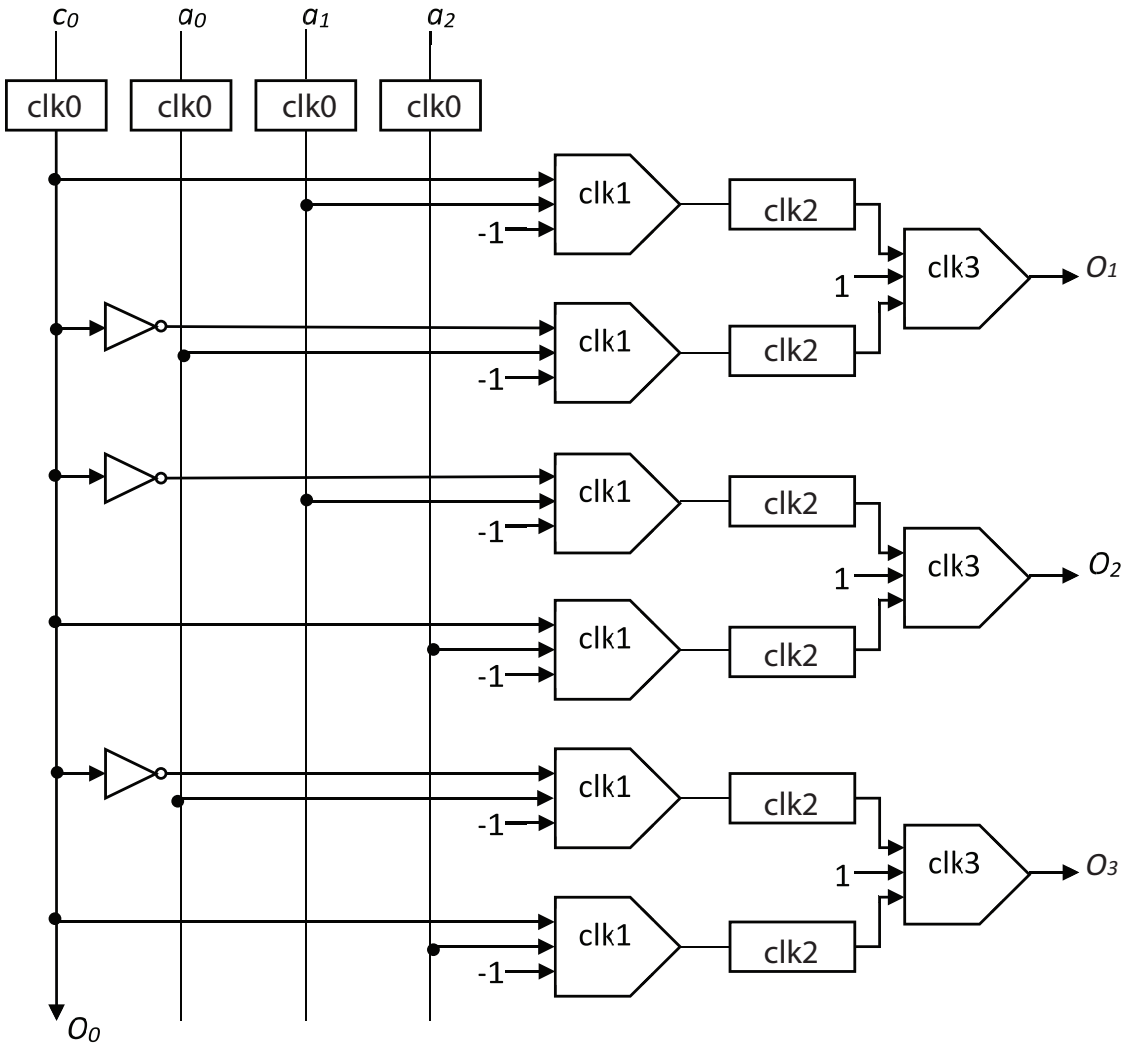


Figure 3.3: QCA design of 4×4 SCRL gate in four-phase clocking scheme

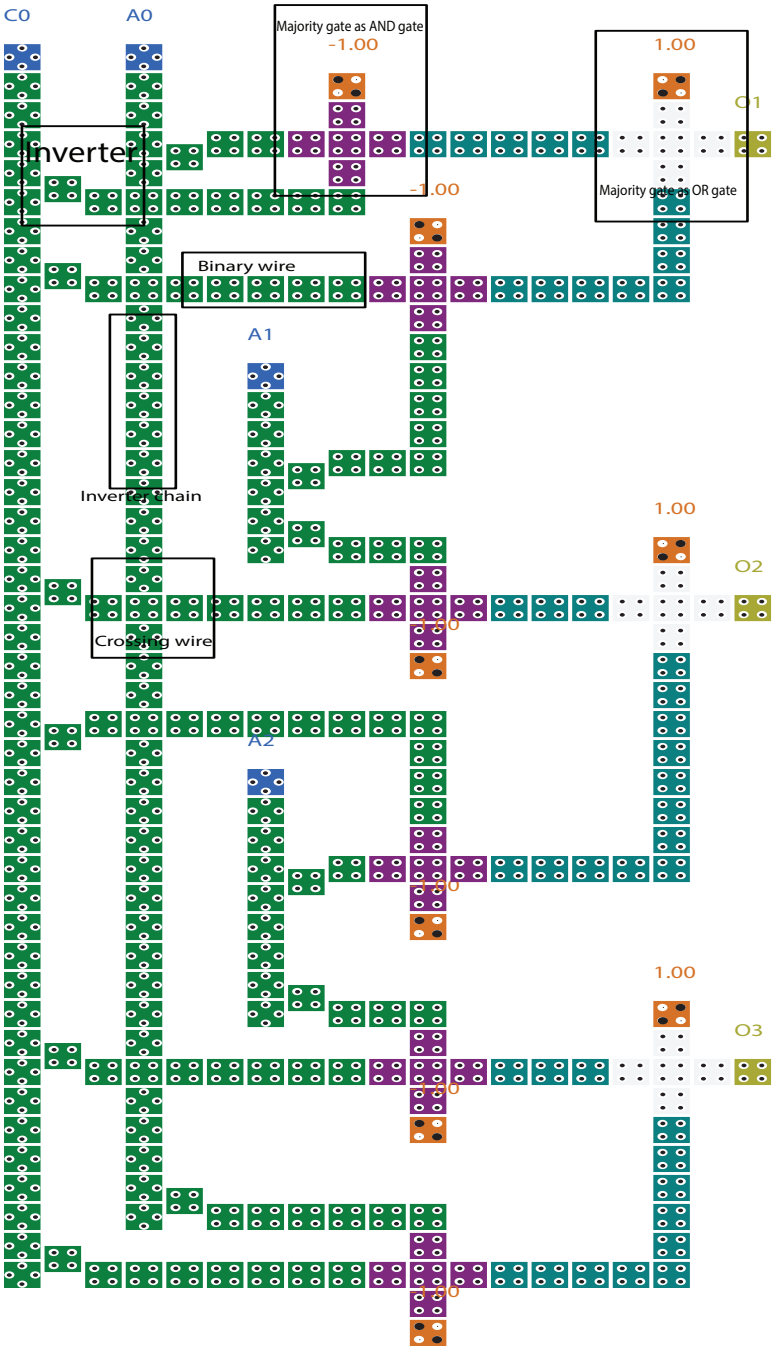


Figure 3.4: 4×4 SCRL gate

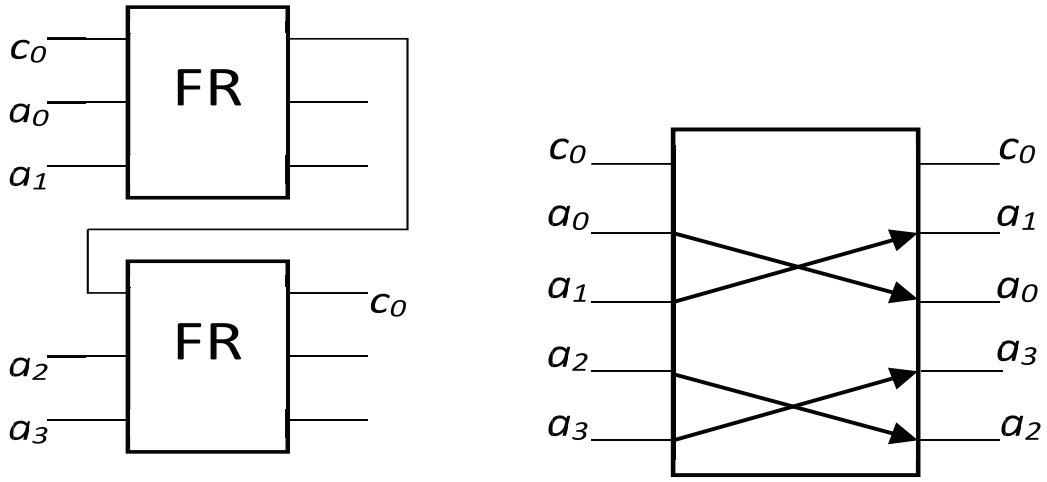
As it is shown, the two cells with misalignment perform as an inverter. The purple crossing wires are majority voters as AND gates and the white ones are majority voters as OR gates. The binary wire, inverter chain and the crossing wire are also shown in Figure 3.4. Table 3.2 is a cost analysis of the SCRL gate.

Table 3.2: Cost analysis of the SCRL gate

	3 inputs	4 inputs	5 inputs	. . .	n inputs
Inverters	2	3	4	. . .	$n-1$
Majority voters	6	9	12	. . .	$3 \times (n-1)$

3.3 Comparison with Fredkin Gate

Fredkin gate has a limited functionality compared to the SCRL gate. Since a Fredkin gate has 1 control signal and 2 inputs, there are always 1 control signal and $2 \times n$ data inputs. If n Fredkin gates are cascaded in series, the total inputs are $2n + 1$. Hence, it is not possible to implement SCRL gates of even inputs using Fredkin gates. The odd inputs SCRL gates using Fredkin gates are also limited in functionality. They can only swap the inputs in pairs while the proposed SCRL gate is able to swap the inputs globally. Figure 3.5(a) shows the Fredkin gate based SCRL gate design and Figure 3.5(b) shows the swapping operation. It can be observed that the Fredkin gate is limited to producing only the odd inputs. Furthermore, the proposed SCRL gate is able to swap the inputs globally. On the other hand, since cascading Fredkin gates in series will require transmitting the control signal, it brings another problem that the operation delay can be high with large number of inputs. However the control signal in the proposed SCRL gate does not need to be propagated serially hence the delay will be constant. This shows that the SCRL gate is functionally more powerful than the Fredkin gate [36].



(a) Fredkin based design of 5×5 SCRL gate

(b) Swapping by Fredkin gate based SCRL gate

Figure 3.5: Fredkin based SCRL gate

Table 3.3 is a comparison of 5×5 Fredkin based SCRL gate and the proposed SCRL gate. The proposed SCRL design reduced the total cells by 101 and reduce the area by $0.27um^2$. Hence the SCRL gate has benefits of implementing in QCA compared to the Fredkin gate.

Table 3.3: A comparison of 5×5 Fredkin based SCRL gate and proposed SCRL gate

	Fredkin	Proposed
Inverters	4	4
Majority voters	12	12
Clock zones	4	4
Total cells	474	373
Area	$0.75um^2$	$0.48um^2$

Chapter 4

Barrel Shifter Designs

4.1 Design Methodology of Right Rotator and Implementation in SCRL Gates

A previous irreversible right rotator design is using $2:1$ multiplexers for switching inputs to achieve rotation. Referring to Chapter 2.1, an SCRL gate is similar to a multiplexer and a normal SCRL gate can rotate the inputs in a certain order with a preset outputs function. In the existing literature [17, 19, 33], barrel shifters are designed with the Fredkin and Feynman gates. Using the Fredkin and the Feynman gates produces considerable *QCA cost (the implementation in QCA costly)*, the amount of garbage and ancilla bits. Since SCRL gate can swap any of the two inputs, the reversible barrel shifter including the reversible rotator has less QCA cost, garbage outputs, and ancilla inputs. Hence, SCRL gate is an ideal platform to design a barrel shifter.

As discussed previously, a logical right rotator function is to right rotate the bits depending on the control signals. Table 2.4 shows an example of an $(8,3)$ reversible right rotator. For an (n,q) reversible right rotator, suppose the inputs are defined as $a_{n-1}, a_{n-2}, ...a_2, a_1, a_0$, it can be designed with q stages, where $q = \log_2 n$. For the p^{th} stage, with a control signal b_{q-p}, it can do either no rotation (with control signal set to 0) or 2^{q-p} bits rotation (with control signal set to 1). Also, define the final outputs as $O_{q,n-1}, O_{q,n-2}, ...O_{q,2}, O_{q,1}, O_{q,0}$. Correspondingly, the p^{th} stage will have the outputs as $O_{p,m}$, where $m = 0, 1, 2.....n-1$. The methodology that can implement the reversible right rotator is as follows:

1. Stage 1
   For the first stage of an (n,q) right rotator, the inputs are a_{n-1},a_{n-2},..... a_2, a_1, a_0, and the outputs have the equation of:

$$O_{1,m} = \begin{cases} \overline{b}_{(q-1)} a_m + b_{(q-1)} a_{(m+2^{q-1}-n)} & \text{if } \; m \geq n - 2^{q-1} \\ \\ \overline{b}_{(q-1)} a_m + b_{(q-1)} a_{(m+2^{q-1})} & \text{if } \; m < n - 2^{q-1} \end{cases} \tag{4.1}$$

where b_{q-1} is the control signal of the first stage, and m is the m^{th} input or output in this stage. If $b_{q-1} = 0$, the outputs of the first stage will keep the same as the inputs:

$$O_{1,m} = a_m \tag{4.2}$$

else if $b_{q-1} = 1$, the outputs of the first stage will rotate for 2^{q-1} bits to the right. That is:

$$O_{1,m} = \begin{cases} a_{m+2^{q-1}-n} & \text{if } \ m \geq n - 2^{q-1} \\ \\ a_{m+2^{q-1}} & \text{if } \ m < n - 2^{q-1} \end{cases} \tag{4.3}$$

2. Stage p $(1 \leq p \leq q)$
For the p^{th} stage of an (n, q) right rotator, the inputs are the outputs of the $(p-1)^{th}$ stage, $O_{p-1,n-1}, O_{p-1,n-2}, ...O_{p-1,2}, O_{p-1,1}, O_{p-1,0}$. The outputs of the p^th stage are:

$$O_{p,m} = \begin{cases} \overline{b}_{q-p} O_{p-1,m} + b_{q-p} O_{p-1,m+2^{q-p}-n} & \text{if } \ m \geq n - 2^{q-p} \\ \\ \overline{b}_{q-p} O_{p-1,m} + b_{q-p} O_{p-1,m+2^{q-p}} & \text{if } \ m < n - 2^{q-p} \end{cases} \tag{4.4}$$

where b_{q-p} is the control signal of the p^th stage, and m is the m^{th} input or output in this stage. If $b_{q-p} = 0$, the outputs of the first stage will keep the same as the inputs:

$$O_{p,m} = O_{p-1,m} \tag{4.5}$$

else if $b_{q-p} = 1$, the outputs of the first stage will rotate for 2^{q-p} bits to the right. That is:

$$O_{p,m} = \begin{cases} O_{p-1,m+2^{q-p}-n} & \text{if } \ m \geq n - 2^{q-p} \\ \\ O_{p-1,m-2^{q-p}} & \text{if } \ m < n - 2^{q-p} \end{cases} \tag{4.6}$$

Then at the outputs side of the q^{th} stage (the last stage), the final outputs $O_{q,n-1}$, $O_{q,n-2}$, ... $O_{q,2}$, $O_{q,1}$, $O_{q,0}$ can be rotated from 0 to $n-1$ bits. Using the SCRL gate, each stage of the (n, q) reversible right rotator needs one $(n+1) \times (n+1)$ SCRL gate. Thus in total, the number of $n \times n$ SCRL gates is n.

Throughout the designs of all the 6 functions in the reversible barrel shifter, an $(8,3)$ logic circuit is primarily designed, thereafter the methodology is extended to design (n,q) logic circuit. Figure 4.1 is an example of $(8,3)$ reversible right rotator in 9×9 SCRL gates. An $(8,3)$ right rotator has 3 rotation stages. The first stage will either right rotate the inputs $a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0$ for 4 bits with the control signal $b_2 = 1$ or keep them unchanged with $b_2 = 0$. For convenience, the outputs from the first stage, which are also the inputs of the second stage, are defined as $k_7, k_6, k_5, k_4, k_3, k_2, k_1, k_0$. Stage 2 will either rotate its inputs for 2 bits with $b_1 = 1$ or keep them unchanged with $b_1 = 0$. The outputs from stage 2, which are also the inputs of the third stage, are defined as $j_7, j_6, j_5, j_4, j_3, j_2, j_1, j_0$. Stage 3 will either rotate its inputs for 1 bits with $b_0 = 1$ or keep them unchanged with $b_0 = 0$. The final outputs are defined as $O_7, O_6, O_5, O_4, O_3, O_2, O_1, O_0$. To implement the rotation in SCRL gates, the following equations are defined for each stage, respectively:

$$k_i = \begin{cases} \overline{b}_2 a_i + b_2 a_{i-4} & \text{if } \ i \geq 4 \\ \\ \overline{b}_2 a_i + b_2 a_{i+4} & \text{if } \ i < 4 \end{cases} \tag{4.7}$$

$$j_i = \begin{cases} \overline{b}_1 a_i + b_1 a_{i-6} & \text{if } \ i \geq 6 \\ \\ \overline{b}_1 a_i + b_1 a_{i+2} & \text{if } \ i < 6 \end{cases} \tag{4.8}$$

$$O_i = \begin{cases} \overline{b}_0 a_i + b_0 a_{i-7} & \text{if } \ i \geq 7 \\ \\ \overline{b}_0 a_i + b_0 a_{i+1} & \text{if } \ i < 7 \end{cases} \tag{4.9}$$

In this way, a reversible $(8,3)$ right rotator is designed using three 9×9 SCRL gates. Table 4.1 shows all possible rotating bits and the corresponding states of the 3 control signals. It is interesting that the serial binary number of the control signals are corresponding to the decimal number of the rotating bits. Compared to a previous design with Fredkin gates and Feynman gates, [17, 19, 33] this design with SCRL gates uses less QCA gates which considerably minimize the transmission delay. Also the SCRL design has no garbage outputs which extremely improve the efficiency.

Table 4.1: Rotating bits and corresponding states of control signals

b_2	b_1	b_0	Rotating Bits
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

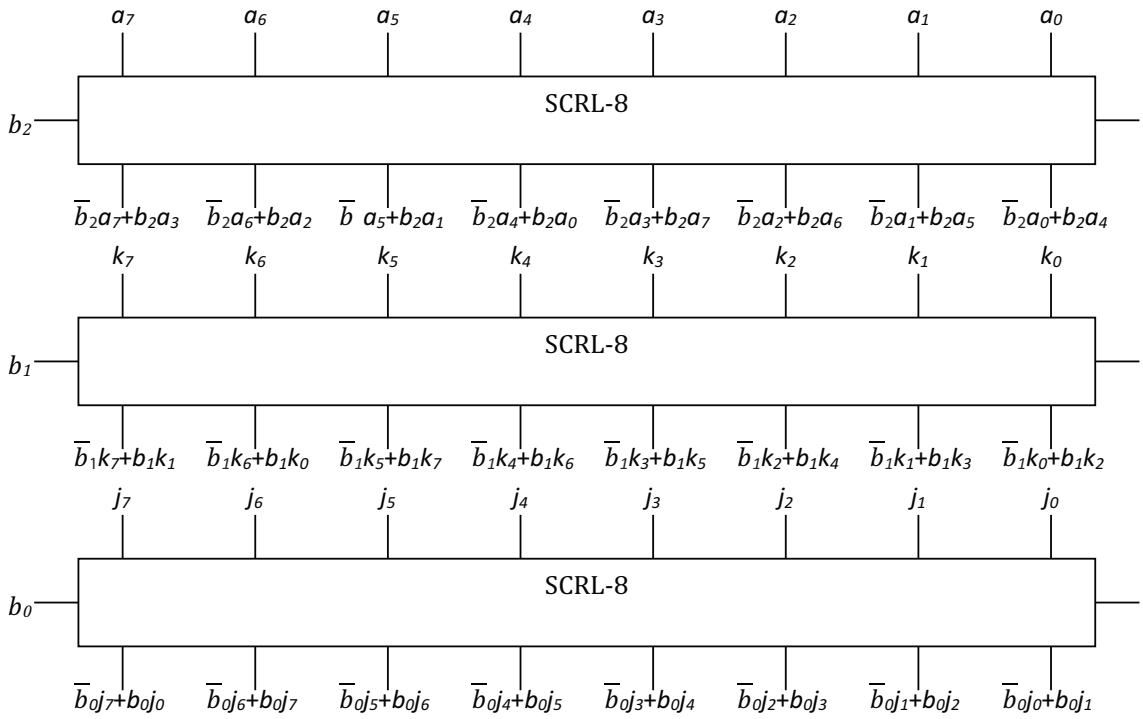


Figure 4.1: Reversible $(8, 3)$ SCRL right rotator

Simulations in Verilog HDL has been made to verify that the design methodology is correct. Each simulation applies different clock cycles for different inputs with a_7 having the shortest clock cycle and a_0 having the longest clock cycle so that it can clearly show the results whether the barrel shifter operates correctly or not. In all the simulation of reversible barrel shifter 3 bits rotation or shift operation is assumed. Figure 4.2 is the simulation result for SCRL right rotator.

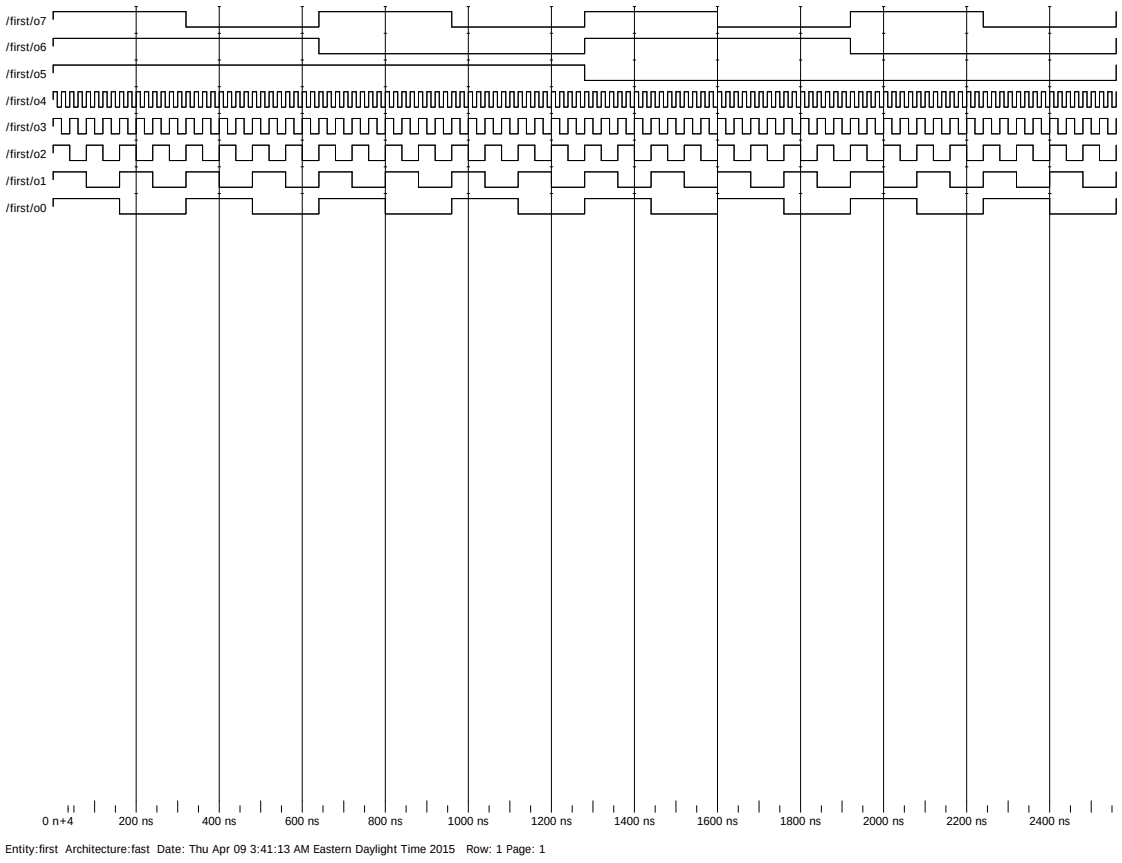


Figure 4.2: Simulation result of SCRL right rotator

4.2 Design Methodology of Logical Right Shifter and Implementation in SCRL Gates

A logical right shifter has the function of shifting the inputs to the right for n bits and set the leftmost n bits to 0. Table 2.4 shows an example of 3 bits logical right shifting, which yields the outputs $0, 0, 0, a_7, a_6, a_5, a_4, a_3$ from $a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0$. Designing an (n, q) reversible logical right shifter having the inputs of $a_{n-1}, a_{n-2}, ...a_2, a_1, a_0$ requires q stages of shifting, where $q = \log_2 n$. Each shifting stages is operating with a control signal b_{q-p}, where $1 \leq p \leq n$. Thus with b_{q-p} set to 0, the shifting mode is off so that the p^{th} shifting stage will not operate a logical right shift. With the control signal b_{q-p} set to 1, the right shifting mode is on and the p^{th} shifting stage will operate a 2^{q-p} bits logical right shifting. Some existing reversible designs of logical right shifters are using Fredkin gates as reversible multiplexers and Feynman gates as reversible data copiers. The use of Fredkin gates and Feynman gates will produce a large numbers of garbage outputs and ancilla inputs. Since SCRL gate can swap any of the two inputs, the reversible barrel shifter including the reversible logical right shifter has less QCA cost, garbage outputs, and ancilla inputs. Hence, SCRL gate is an ideal platform to design a barrel shifter.
The difference between logical right shifter and right rotator is that a logical right shifter

will set the leftmost shifted bits to 0 while a right rotator will not. As a result, it requires several selecters to select an input from 0 and any other original inputs. For example, the p^{th} stage requires a total number of 2^{q-p} selecters to achieve a 2^{q-p} bits right shifting. In this design, Fredkin gates are used to construct the selection unit so that the design will keep reversible. Figure 4.3 is an example for an $(8, 3)$ selection unit of logical right shifter.

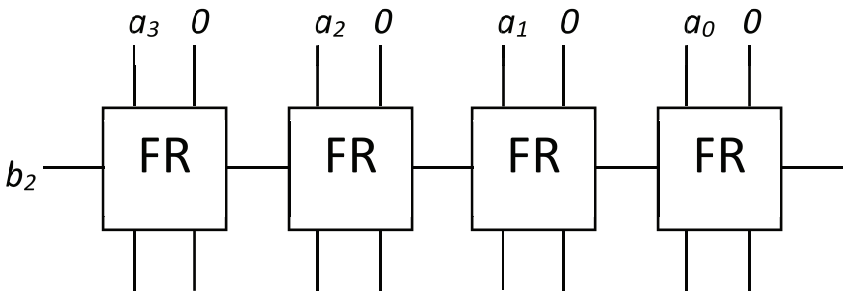


Figure 4.3: Selection unit of an $(8, 3)$ logical right shifter

1. Stage 1
   The first stage has the inputs of a selected n bits data from $0, a_{n-1}, a_{n-2}, ...a_2, a_1$, and a control signal b_{q-1}. The first stage requires a total number of 2^{q-1} Fredkin gates to select the input and 1 $n \times n$ SCRL gate. Define the final outputs as $O_{q,n-1}, O_{q,n-2}, ...O_{q,2}, O_{q,1}, O_{q,0}$. Correspondingly, the p^{th} stage has the outputs expressed by $O_{p,m}$, where m is the m^{th} output of this stage. Thus for stage 1, define the selected inputs as $s_{n-1-2^{q-1}}, ...s_2, s_1, s_0$. They have the following equation:

$$s_m = \overline{b}_{q-1} a_m \quad \text{for} \quad 0 \leq m \leq n - 1 - 2^{q-1} \tag{4.10}$$

   where b_{q-1} is the control bit for all the Fredkin gate. When b_{q-1} is set to 0, $s_m = a_m$, the actual inputs will keep unchanged as the primary inputs $a_{n-1}, a_{n-2}, ...a_2, a_1$. If b_{q-1} is set to 1, then $s_m = 0$, it allows this SCRL design of logical right shifter to set the leftmost shifted bits as 0. Then the actual inputs are $s_{n-1-2^{q-1}}$, ... $s_2, s_1, s_0, a_{n-1}, a_{n-2}, ...a_{n-2^{q-1}}$. The outputs have the following equation:

$$O_{1,m} = \begin{cases} \overline{b}_{q-1} a_m + b_{q-1} s_{m+2^{q-1}-n} & \text{if} \quad m \geq n - 2^{q-1} \\ \\ \overline{b}_{q-1} s_m + b_{q-1} a_{m+2^{q-1}} & \text{if} \quad m < n - 2^{q-1} \end{cases} \tag{4.11}$$

   where b_{q-1} is the control signal of the first stage, and m is the m^{th} input or output in this stage. If $b_{q-1} = 0$, the outputs of the first stage will keep the same as the inputs:

$$O_{1,m} = \begin{cases} a_m & \text{if } m \geq n - 2^{q-1} \\ \\ s_m & \text{if } m < n - 2^{q-1} \end{cases} \tag{4.12}$$

else if $b_{q-1} = 1$, the outputs of the first stage will rotate for 2^{q-1} bits to the right. That is:

$$O_{1,m} = \begin{cases} s_{m+2^{q-1}-n} & \text{if } m \geq n - 2^{q-1} \\ \\ a_{m+2^{q-1}} & \text{if } m < n - 2^{q-1} \end{cases} \tag{4.13}$$

2. Stage p $(1 \leq p \leq q)$
At the p^{th} stage of an SCRL logical right shifter, the primary inputs are $O_{p-1,n-1}$, $O_{p-1,n-2}$, ... $O_{p-1,2}, O_{p-1,1}, O_{p-1,0}$, which are from the previous stage. The p^{th} stage requires the total number of 2^{q-p} Fredkin gates as selecters, allowing the stage to select the actual inputs from 0 and the rightmost 2^{q-p} primary inputs, $O_{p-1,2^{q-p}-1}, O_{p-1,2^{q-p}-2}$, ... $O_{p-1,2}, O_{p-1,1}, O_{p-1,0}$. Now define the selected inputs as $s_{2^{q-p}-1}, s_{2^{q-p}-2}$, ...$s_2, s_1, s_0$. The selected inputs have the following equation:

$$s_m = \overline{b}_{q-p} O_{p-1,m} \quad \text{for} \quad 0 \leq m \leq n - 1 - 2^{q-p} \tag{4.14}$$

When b_{q-p} is set to 0, $s_m = O_{p-1,m}$, the actual inputs will keep unchanged as the primary inputs $O_{p-1,n-1}, O_{p-1,n-2}$, ... $O_{p-1,2}, O_{p-1,1}, O_{p-1,0}$. If b_{q-p} is set to 1, then $s_m = 0$, it allows this SCRL design of logical right shifter to set the leftmost shifted bits as 0. Then the actual inputs are $s_{n-1-2^{q-p}}$, ... s_2, s_1, s_0, $O_{p-1,n-1}, O_{p-1,n-2}$, ...$O_{p-1,n-2^{q-p}}$. Then the outputs equation is as the following:

$$O_{p,m} = \begin{cases} \overline{b}_{q-p} O_{p-1,m} + b_{q-p} s_{m+2^{q-p}-n} & \text{if } m \geq n - 2^{q-p} \\ \\ \overline{b}_{q-p} s_m + b_{q-p} O_{p-1,m+2^{q-p}} & \text{if } m < n - 2^{q-p} \end{cases} \tag{4.15}$$

where b_{q-p} is the control signal of the p^{th} stage, and m is the m^{th} input or output in this stage. If $b_{q-p} = 0$, the outputs of the p^{th} stage will keep the same as the inputs:

$$O_{p,m} = \begin{cases} O_{p-1,m} & \text{if } m \geq n - 2^{q-p} \\ \\ s_m & \text{if } m < n - 2^{q-p} \end{cases} \tag{4.16}$$

else if $b_{q-p} = 1$, the outputs of the p^{th} stage will rotate for 2^{q-p} bits to the right. That is:

$$O_{p,m} = \begin{cases} s_{m+2^{q-p}-n} & \text{if } \ m \geq n - 2^{q-p} \\ \\ O_{p-1,m+2^{q-p}} & \text{if } \ m < n - 2^{q-p} \end{cases} \tag{4.17}$$

Therefore, at the final outputs side of the logical right shifter, the primary inputs can be right shifted from 0 to $n-1$ bits according to the different settings of the control bits b_m. In SCRL design of an (n, q) logical right shifters, it requires q SCRL gates, $n-1$ Fredkin gates. As it is shown in Figure 4.4, an $(8, 3)$ logical right shifter has 3 shifting stages with the primary inputs as $a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0$. The corresponding final outputs are $O_7, O_6, O_5, O_4, O_3, O_2, O_1, O_0$. To simplify the design, define the outputs for the first stage as $k_7, k_6, k_5, k_4, k_3, k_2, k_1, k_0$, define the outputs for stage 2 as $j_7, j_6, j_5, j_4, j_3, j_2, j_1, j_0$. And to separate the selected inputs from each stage, define the selected input for stage 1 as s_3, s_2, s_1, s_0, the selected inputs for stage 2 as x_1, x_0, the selected inputs for stage 3 as y_0. The selected inputs are selected by the following equtions:

$$s_m = \overline{b}_2 a_m \ \text{ for } \ 0 \leq m \leq 3 \tag{4.18}$$

$$x_m = \overline{b}_1 k_m \ \text{ for } \ 0 \leq m \leq 1 \tag{4.19}$$

$$y_m = \overline{b}_0 j_m \ \text{ for } \ m = 0 \tag{4.20}$$

The first stage will either right shift the actual inputs $a_7, a_6, a_5, s_4, s_3, s_2, s_1, s_0$ for 4 bits with the control signal $b_2 = 1$ or keep them unchanged with $b_2 = 0$. Stage 2 will either right shift its inputs for 2 bits with $b_1 = 1$ or keep them unchanged with $b_1 = 0$. Stage 3 will either right shift its inputs for 1 bits with $b_0 = 1$ or keep them unchanged with $b_0 = 0$. For each stage, the outputs are shown in the following equations:

$$k_i = \begin{cases} \overline{b}_2 a_i + b_2 s_{i-4} & \text{if } \ i \geq 4 \\ \\ \overline{b}_2 s_i + b_2 a_{i+4} & \text{if } \ i < 4 \end{cases} \tag{4.21}$$

$$j_i = \begin{cases} \overline{b}_1 a_i + b_1 x_{i-6} & \text{if } \ i \geq 6 \\ \\ \overline{b}_1 x_i + b_1 a_{i+2} & \text{if } \ i < 6 \end{cases} \tag{4.22}$$

$$O_i = \begin{cases} \bar{b}_0 a_i + b_0 y_{i-7} & \text{if } \ i \geq 7 \\ \\ \bar{b}_0 y_i + b_0 a_{i+1} & \text{if } \ i < 7 \end{cases} \tag{4.23}$$

In this way, a reversible $(8, 3)$ logical right shifter can be worked out using 3 SCRL gates and 7 Fredkin gates. The use of 7 Fredkin gates produces 7 garbage outputs in this reversible logical right shifter design, which has reduced much more garbage outputs than the reversible designs in existing literature. Also, the operation delay is optimized since this design use less 2×2 logic gates. Table 4.2 shows the possible right shift operations with the different control signal values. And Figure 4.5 is the simulation result of SCRL logical right shifter.

Table 4.2: Shifting bits and corresponding states of control signals

b_2	b_1	b_0	Shifting Bits
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

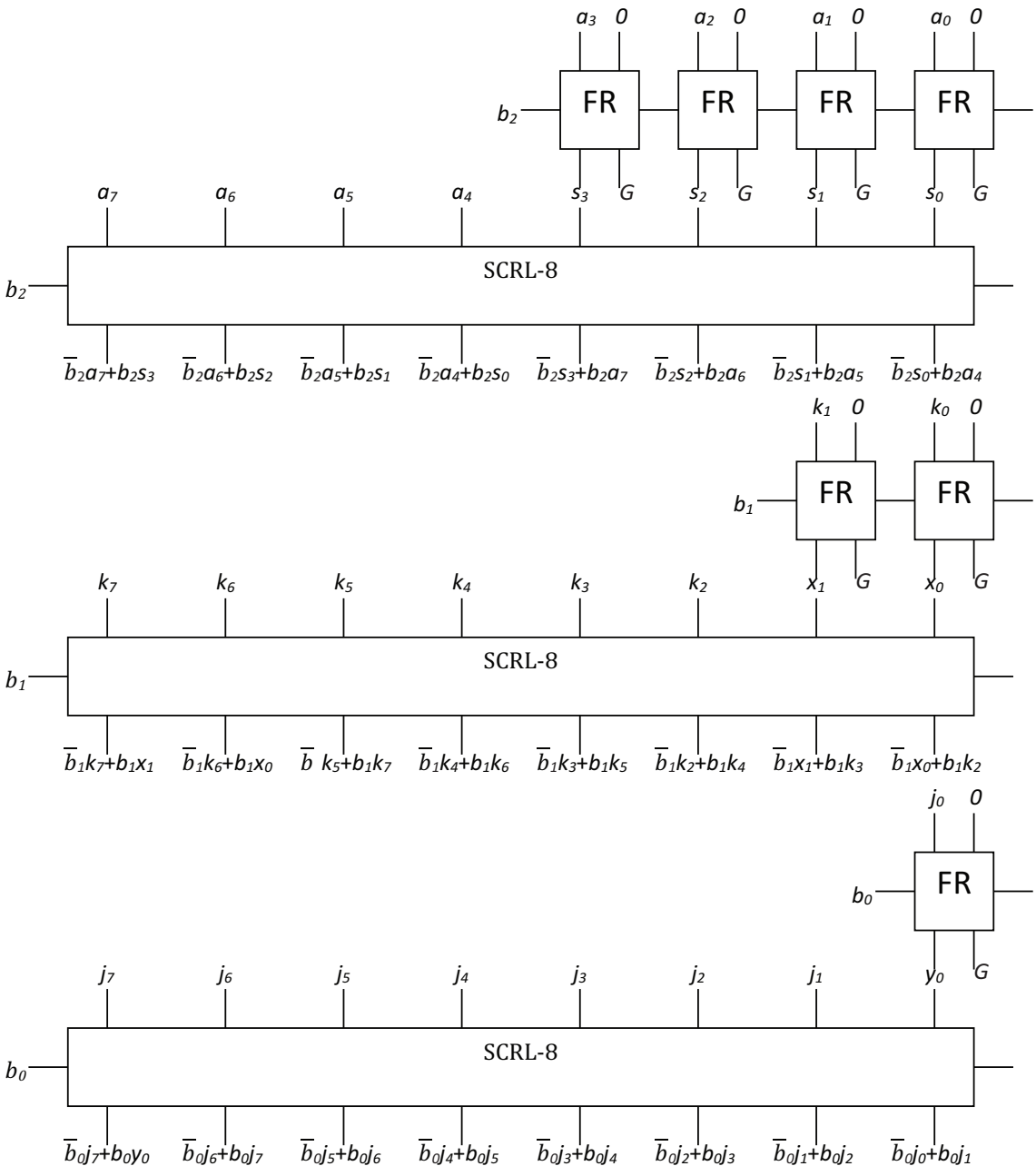


Figure 4.4: $(8, 3)$ SCRL logical right shifter (FR: Frdkin gate, G: garbage outputs)

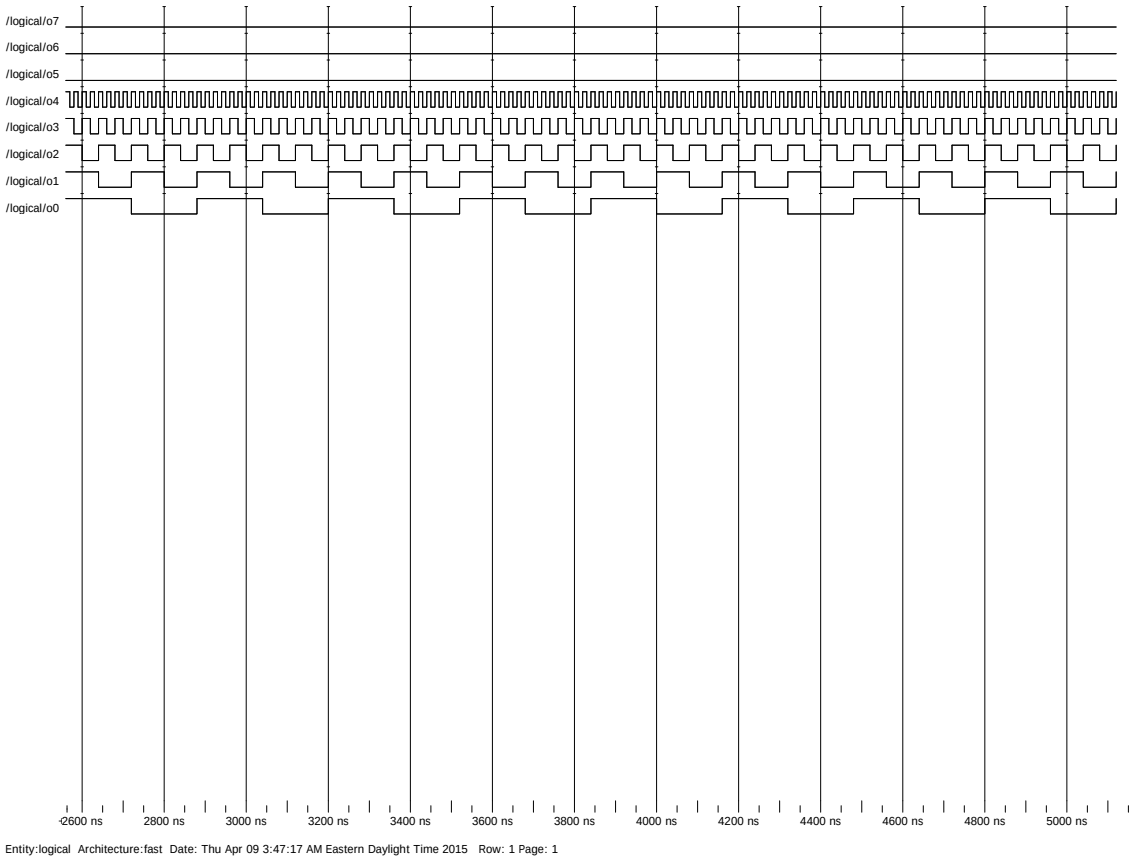


Figure 4.5: Simulation result of SCRL logical right shifter

4.3 Design Methodology of Arithmetic Right Shifter and Implementation in SCRL Gates

An arithmetic right shifter has the function to shift the inputs to the right for n bits and always keep the leftmost n bits as same as the sign bit a_{n-1}. Table 2.4 shows an example of 3 bits arithmetic right shifting, it changes the inputs $a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0$ into the outputs $a_7, a_7, a_7, a_7, a_6, a_5, a_4, a_3$. In order to design an (n, q) reversible arithmetic right shifter with the initial inputs as a_{n-1}, a_{n-2},..... a_2, a_1, a_0 in SCRL gates. It requires q stages of shifting operations, where $q = \log_2 n$. Each one of the q shifting stages is operating with a control signal b_{q-p}, where p is the p^{th} stage, $p = 1, 2.....q$. Thus with b_{q-p} set to 0, the shifting mode is off so that the p^{th} shifting stage will not shift the inputs data. With the control signal b_{q-p} set to 1, the right shifting mode is excited and now the p^{th} shifting stage will operate a 2^{q-p} bits right shifting. Different form the logical right shifter, the arithmetic right shifter always keep the leftmost shifting bits as same as the sign bit a_{n-1} instead of 0. Therefore, in the SCRL design of the arithmetic right shifter, the main difference from the logical right shifter is that the arithmetic right shifter will use Fredkin gates to select from an inputs ofa_{n-1} and the remaining initial inputs, while the logical shifter selects from 0 and the initial inputs. Another difference is that in the design of the SCRL arithmetic

right shifter, there will be a certain number of Feynman gates used in each stage to copy the sign bit a_{n-1}. Since Feynman gates do not have any garbage outputs, the amount of garbage outputs is the same in reversible SCRL logical right shifter. Figure 4.6 shows the operation of the copy and selection of the first stage in an $(8, 3)$ arithmetic right shifter. As discussed in Chapter 2, when input B of a Feynman gate is at 0, this Feynman gate performs as a copier. The two outputs of this Feynman gate are both the same as the first input. The copy-selection unit shown in Figure 4.6 uses the Feynman gates to achieve the copy function and then pass the copied sign bit to the Fredkin gate to be selected from. The rightmost output of the Feynman gate is not in use in as shown, but actually it will be used in the next stage. The selection part of the arithmetic right shifter will either pass the sign bit (when the control signal is at 1) or the initial inputs (when the control bit is at 0) to the shifting operation unit.

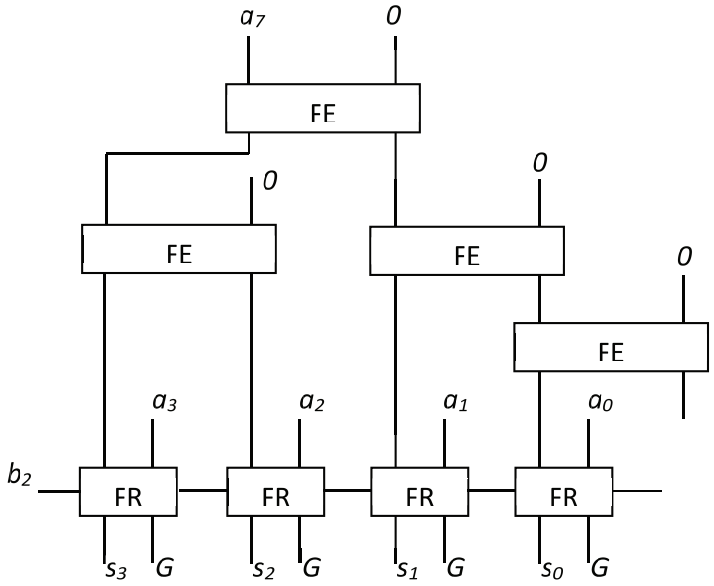


Figure 4.6: Example of copy-selection unit (FE: Feynman gate, FR: Fredkin gate, G: garbage outputs)

The previous reversible design in [19] is using all Fredkin gates and Feynman gates. This will produce a large numbers of garbage outputs. A number of n garbage outputs will be yielded for each stage. From another point of view, the two-bits-in-a-group way of switching data requires much more time passing the control signals. In this design using less Fredkin gates and Feynman gates, a great improvement in reducing garbage outputs and minimizing operation delay can be achieved.

1. Stage 1
   As usual, define the initial inputs of the first stage of an arithmetic right shifter as $a_{n-1}, a_{n-2}, \ldots\ldots\ a_2, a_1, a_0$ with a control signal b_{q-1}. Since there are 2^{q-1} selected inputs in the first stage, then define the selected inputs as $s_{n-1-2^{q-1}}, \ldots\ldots\ s_2, s_1, s_0$. Hence, to select the 2^{q-1} bits of inputs, a total number of 2^{q-1} Fredkin gates are

required. And to make copies of the sign bit a_{n-1} for the Fredkin gates, a total number of 2^{q-1} Feynman gates are also required. If the final outpus are defined as $O_{q,n-1}, O_{q,n-2}$, ... $O_{q,2}, O_{q,1}, O_{q,0}$, correspondingly, the p^{th} stage will have the outputs expressed by $O_{p,m}$, where $0 \leq m \leq n-1$. For $b_{q-1} = 0$, the selection unit will select a serial inputs of $a_{n-1-2^{q-1}}$, ... a_2, a_1, a_0, and the shifting unit will not be exited thus the first stage of the arithmetic right shifter will not perform shifts. For $b_{q-1} = 1$, the selection unit will select a serial inputs of $s_{n-1-2^{q-1}}$, ... s_2, s_1, s_0, and the shifting unit is turned on to make a 2^{q-1} bits right shift. In this way, the arithmetic right shifter performs an arithmetic right shift. The selection inputs have the following equation:

$$s_m = \overline{b}_{q-1} a_m + b_{q-1} a_{n-1} \quad \text{for} \quad 0 \leq m \leq n-1-2^{q-1} \tag{4.24}$$

When b_{q-1} is set to 0, $s_m = a_m$, the actual inputs will keep unchanged as the primary inputs a_{n-1}, a_{n-2}, ... a_2, a_1. If b_{q-1} is set to 1, then $s_m = a_{n-1}$, it allows this SCRL design of logical right shifter to set the leftmost shifted bits as the sign bit a_{n-1}. Then the actual inputs are $s_{n-1-2^{q-1}}$, ... $s_2, s_1, s_0, a_{n-1}, a_{n-2}$, ... $a_{n-2^{q-1}}$. The outputs have the following equation:

$$O_{1,m} = \begin{cases} \overline{b}_{q-1} a_m + b_{q-1} s_{m+2^{q-1}-n} & \text{if} \quad m \geq n-2^{q-1} \\ \\ \overline{b}_{q-1} s_m + b_{q-1} a_{m+2^{q-1}} & \text{if} \quad m < n-2^{q-1} \end{cases} \tag{4.25}$$

If $b_{q-1} = 0$, the outputs of the first stage will keep the same as the inputs:

$$O_{1,m} = \begin{cases} a_m & \text{if} \quad m \geq n-2^{q-1} \\ \\ s_m & \text{if} \quad m < n-2^{q-1} \end{cases} \tag{4.26}$$

else if $b_{q-1} = 1$, the outputs of the first stage will rotate for 2^{q-1} bits to the right. That is:

$$O_{1,m} = \begin{cases} s_{m+2^{q-1}-n} & \text{if} \quad m \geq n-2^{q-1} \\ \\ a_{m+2^{q-1}} & \text{if} \quad m < n-2^{q-1} \end{cases} \tag{4.27}$$

2. Stage p $(1 \leq p \leq q)$
   At the p^{th} stage of an SCRL logical right shifter, the initial inputs are the outputs from the previous stage $O_{p-1,n-1}$, $O_{p-1,n-2}$, ... $O_{p-1,2}, O_{p-1,1}, O_{p-1,0}$. The p^{th} stage requires the total number of 2^{q-p} Fredkin gates as selecters, and 2^{q-p} Feynman gates

to achieve the function to select the actual inputs from the sign bit a_{n-1} and the rightmost 2^{q-p} initial inputs, $O_{p-1,2^{q-p}-1}, O_{p-1,2^{q-p}-2}$, ... $O_{p-1,2}, O_{p-1,1}, O_{p-1,0}$. If the selected inputs are defined as $s_{2^{q-p}-1}, s_{2^{q-p}-2}$, ... s_2, s_1, s_0, they have the following equation:

$$s_m = \overline{b}_{q-p} O_{p-1,m} + b_{q-p} a_{n-1} \quad \text{for} \quad 0 \leq m \leq n-1-2^{q-p} \tag{4.28}$$

When b_{q-p} is set to 0, $s_m = O_{p-1,m}$, the actual inputs will keep unchanged as the primary inputs $O_{p-1,n-1}, O_{p-1,n-2}$, ... $O_{p-1,2}, O_{p-1,1}$. If b_{q-p} is set to 1, then $s_m = a_{n-1}$. Therefore the leftmost shifted bits can be set to either the initial inputs or the sign bit a_{n-1}. Then the actual inputs are $s_{n-1-2^{q-p}}$, ... s_2, s_1, s_0, $O_{p-1,n-1}, O_{p-1,n-2}, ...O_{p-1,n-2^{q-p}}$. The outputs equation can be expressed as the following:

$$O_{p,m} = \begin{cases} \overline{b}_{q-p} O_{p-1,m} + b_{q-p} s_{m+2^{q-p}-n} & \text{if} \quad m \geq n-2^{q-p} \\ \\ \overline{b}_{q-p} s_m + b_{q-p} O_{p-1,m+2^{q-p}} & \text{if} \quad m < n-2^{q-p} \end{cases} \tag{4.29}$$

If $b_{q-p} = 0$, the outputs of the p^{th} stage will keep the same as the inputs:

$$O_{p,m} = \begin{cases} O_{p-1,m} & \text{if} \quad m \geq n-2^{q-p} \\ \\ s_m & \text{if} \quad m < n-2^{q-p} \end{cases} \tag{4.30}$$

else if $b_{q-p} = 1$, the outputs of the p^{th} stage will right shift for 2^{q-p} bits to the right. That is:

$$O_{p,m} = \begin{cases} s_{m+2^{q-p}-n} & \text{if} \quad m \geq n-2^{q-p} \\ \\ O_{p-1,m+2^{q-p}} & \text{if} \quad m < n-2^{q-p} \end{cases} \tag{4.31}$$

As a result, at the final outputs side of the arithmetic right shifter, the initial inputs can be right shifted from 0 to $n-1$ bits according to the different settings of the control bits b_m. It is also mentioned in the previous section that if considering all the different states of the combinations of the control bits as binary numbers, the corresponding decimal numbers are the right shifting bits from 0 to $n-1$. In SCRL design of an (n, q) arithmetic right shifters, it requires q SCRL gates, $n-1$ Fredkin gates and $n-1$ Feynman gates. To make it intuitional, a design of $(8, 3)$ SCRL-based arithmetic right shifter is shown in Figure 4.7. It has 3 shifting stages with the initial inputs as $a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0$. The corresponding final outputs are $O_7, O_6, O_5, O_4, O_3, O_2, O_1, O_0$. To simplify the expressions,

define the outputs for the first stage as $k_7, k_6, k_5, k_4, k_3, k_2, k_1, k_0$, and define the outputs for stage 2 as $j_7, j_6, j_5, j_4, j_3, j_2, j_1, j_0$. And also to separate the selected inputs from each stage, define the selected input for stage 1 as s_3, s_2, s_1, s_0, the selected inputs for stage 2 as x_1, x_0, the selected inputs for stage 3 as y_0. The selected inputs are selected by the following equations:

$$s_m = \overline{b}_2 a_m + b_2 a_7 \quad \text{for} \quad 0 \leq m \leq 3 \tag{4.32}$$

$$x_m = \overline{b}_1 k_m + b_1 a_7 \quad \text{for} \quad 0 \leq m \leq 1 \tag{4.33}$$

$$y_m = \overline{b}_0 j_m + b_0 a_7 \quad \text{for} \quad m = 0 \tag{4.34}$$

The first stage shift unit will either right shift the actual inputs $a_7, a_6, a_5, s_4, s_3, s_2, s_1, s_0$ for 4 bits with the control signal $b_2 = 1$ while it keeps the leftmost 4 bits of outputs as same as the sign bit a_7 or keep them unchanged with $b_2 = 0$. The shift unit of stage 2 will either right shift its inputs for 2 bits with $b_1 = 1$ while it keeps the left most 2 bits of outputs as same as the sign bit a_7 or keep them unchanged with $b_1 = 0$. At last the shift unit of stage 3 will either right shift its inputs for 1 bits with $b_0 = 1$ while it keeps the leftmost 1 bit of output as same as the sign bit a_7 or keep them unchanged with $b_0 = 0$. For each stage, the outputs are shown in the following equations:

$$k_i = \begin{cases} \overline{b}_2 a_i + b_2 s_{i-4} & \text{if} \quad i \geq 4 \\ \\ \overline{b}_2 s_i + b_2 a_{i+4} & \text{if} \quad i < 4 \end{cases} \tag{4.35}$$

$$j_i = \begin{cases} \overline{b}_1 a_i + b_1 x_{i-6} & \text{if} \quad i \geq 6 \\ \\ \overline{b}_1 x_i + b_1 a_{i+2} & \text{if} \quad i < 6 \end{cases} \tag{4.36}$$

$$O_i = \begin{cases} \overline{b}_0 a_i + b_0 y_{i-7} & \text{if} \quad i \geq 7 \\ \\ \overline{b}_0 y_i + b_0 a_{i+1} & \text{if} \quad i < 7 \end{cases} \tag{4.37}$$

Thus, a reversible $(8, 3)$ arithmetic right shifter function is achieved. As it is shown in Figure 4.7, this design totally yields 7 garbage outputs, which has reduced much more garbage outputs than the previous reversible designs. This design applies 7 Fredkin gates and 7 Feynman gates in total. The reduced requirement of the $2 : 2$ logic gates also contributes to the optimization of the operation delay. Table 4.3 shows the possible right shift operations and the different combinations of the control signals. And Figure 4.8 is the simulation result of SCRL arithmetic right shifter.

Table 4.3: Possible operations

b_2	b_1	b_0	Shifting Bits
0	0	0	$a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0$
0	0	1	$a_7, a_7, a_6, a_5, a_4, a_3, a_2, a_1$
0	1	0	$a_7, a_7, a_7, a_6, a_5, a_4, a_3, a_2$
0	1	1	$a_7, a_7, a_7, a_7, a_6, a_5, a_4, a_3$
1	0	0	$a_7, a_7, a_7, a_7, a_7, a_6, a_5, a_4$
1	0	1	$a_7, a_7, a_7, a_7, a_7, a_7, a_6, a_5$
1	1	0	$a_7, a_7, a_7, a_7, a_7, a_7, a_7, a_6$
1	1	1	$a_7, a_7, a_7, a_7, a_7, a_7, a_7, a_7$

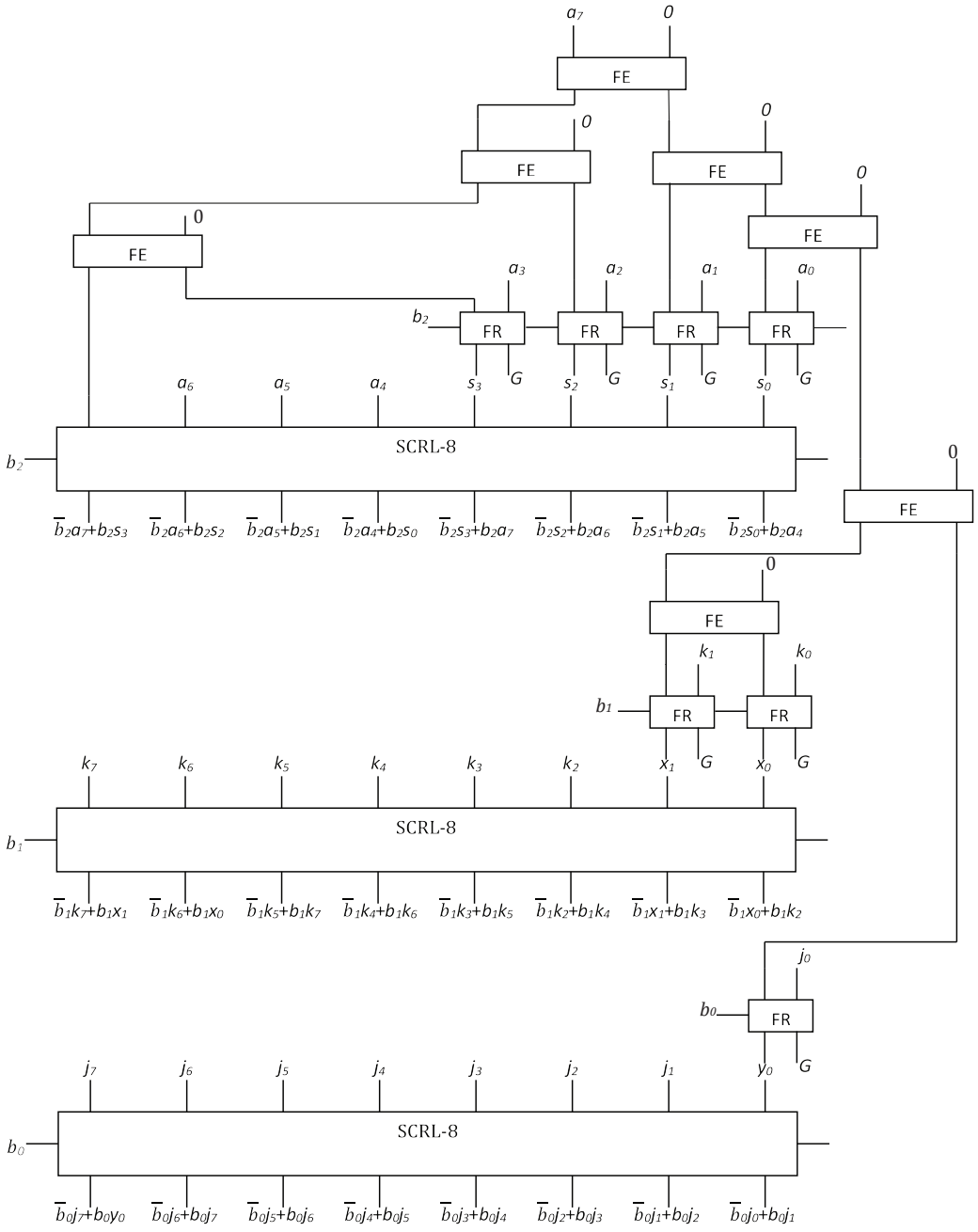


Figure 4.7: $(8, 3)$ SCRL arithmetic right shifter (FR: Frdkin gate, FE: Feynman gate, G: garbage outputs)

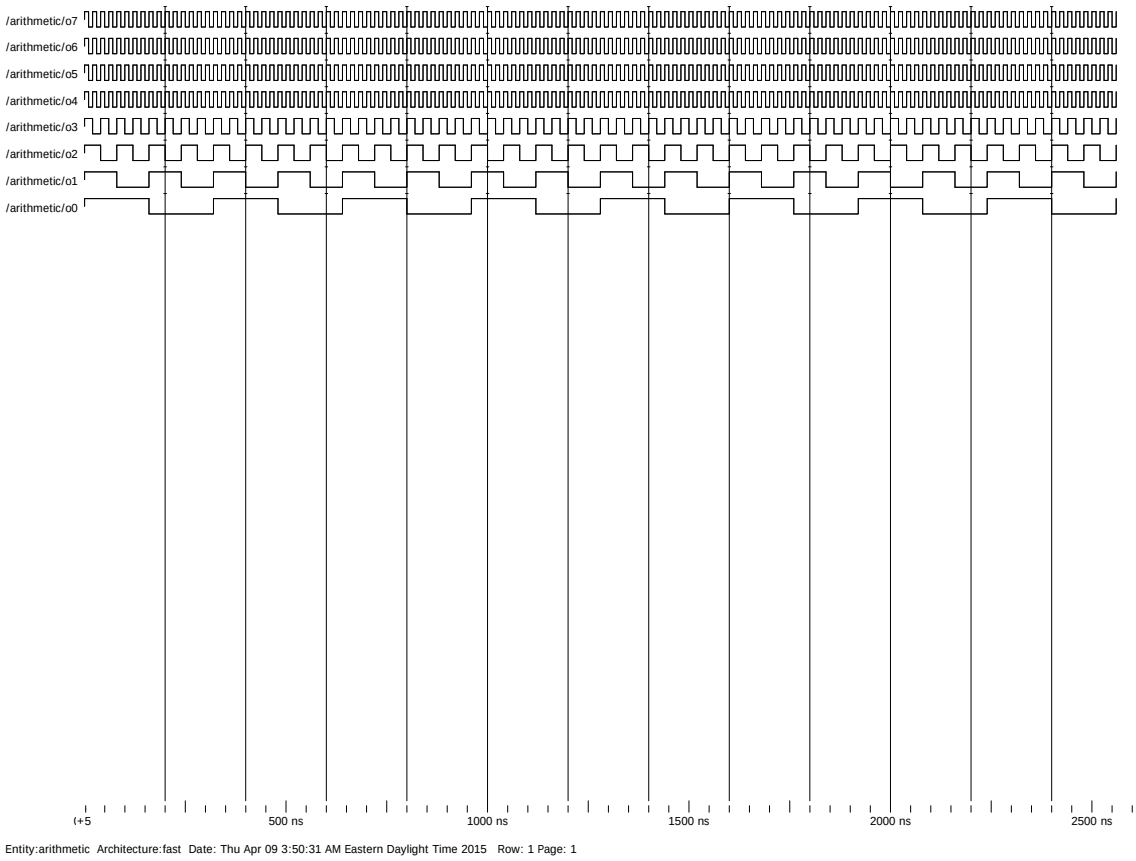


Figure 4.8: Simulation result of SCRL arithmetic right shifter

4.4 Design Methodology of Universal Right Shifter and Implementation in SCRL Gates

Looking back to the previous three sections of the final implementations, it is not difficult to find out that there are similarities in the designs. The rotation or shift units are very similar to each other. A universal right shifter combines the similarities and develop another function unit to achieve the three different functions: right rotation, logical right shift and arithmetic right shift. The operations are shown in Table 2.4. The new function unit is capable to deal command to make a decision to complete a right rotation or a logical right shift or an arithmetic right shift. Thus it requires some Fredkin gates to achieve the selection function, and some Feynman gates to copy the selected outputs to transmit to the shifting units. Similarly to the previous designs, for an (n, q) universal right shifter, there are 2^{q-1} shifting stages of the shifting unit. In Stage 1, the rightmost 2^{q-1} bits of inputs are involved in the selection unit and the same bits of inputs may be shifted after the selection. Different form the previous selection units of the logical right shifter and the arithmetic right shifter, the selection unit of the universal right shifter will select the inputs for two times, first select from 0 and the sign bit a_{n-1} to decide whether the universal right shifter is a logical right shifter (select 0 as the inputs) or an arithmetic right shifter (select a_{n-1} as

the inputs). Then in the next selection, the selection unit will select from the initial inputs $a_{2^{q-1}-1}, a_{2^{q-1}-2}$, ...$a_2, a_1, a_0$ and the outputs of the previous selection. Then as a result the selection unit is functioned to complete a selection from the three different functions, right rotation, logical right shift and arithmetic right shift. The control signals c_0, b_{q-1} are responsible for selecting the operations. c_0 controls the selection from logical right shift and arithmetic shift while t_{q-1} controls the selection of the right rotation and the previous selected function. Table 4.4 shows the different operations selected by the different states of the two control signals above.

Table 4.4: Different operations of universal right shifter

c_0	t_{q-1}	Operations
0	0	Right rotation
0	1	Arithmetic
1	0	Right rotation
1	1	Logical right shift

1. Selection unit
   There are two parts of the selection unit. The first part is responsible for selecting from the sign bit a_{n-1} and 0 so that the logic circuit is capable to select an operation of logical right shift or an arithmetic right shift. Define the output of the first part of the selection unit as F, it has the following equation:

$$F = \bar{c}_0 a_{n-1} \tag{4.38}$$

   in this way the Fredkin gate selects the input a_{n-1} when $c_0 = 1$ so that the first part of the selection unit selects the operation mode of arithmetic right shift. In another way, the Fredkin gate selects the input 0 when $c_0 = 0$, so that the logic circuit selects the operation mode of logical right shift. Then the second selection part of the selection unit will select from the initial inputs $a_{2^{q-1}-1}, a_{2^{q-1}-2}$, ...$a_2, a_1, a_0$ and the primarily selected input F. It requires 2^{q-1} Feynman gates to make copies of F for the second selection part of the unit. The final selected inputs $s_{2^{q-1}-1}, s_{2^{q-1}-2}$, ... s_2, s_1, s_0 are selected by:

$$s_m = \bar{t}_{q-1} a_m + t_{q-1} F \quad \text{for} \quad m = 0, 1, 2, ...2^{q-1} - 1 \tag{4.39}$$

   When $t_{q-1} = 0$, the selection unit selects the inputs as it is selected in by the first Fredkin gate so that the logic circuit performs as either a logical right shifter or an arithmetic right shifter depending on the value of F. When $t_{q-1} = 1$, the selection unit selects the inputs as the initial inputs so that the logic circuit performs an operation of a right rotator. To make a more visualized explanation, an example for a selection unit of an $(8, 3)$ universal right shifter is shown in Figure 4.9.

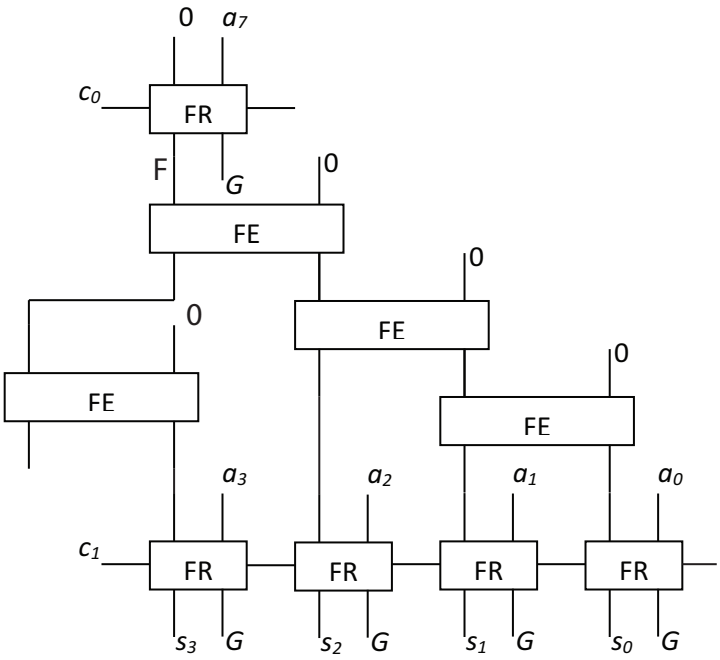


Figure 4.9: Selection unit of $(8, 3)$ universal right shifter (FR: Fredkin gate, FE: Feynman gate, G: Garbage outputs)

F is the output of the first Fredkin gate. It has the equation as:

$$F = \bar{c}_0 a_7 \tag{4.40}$$

For $c_0 = 0$, $F = 0$, the first Fredkin gate selects the input 0 and for $c_0 = 1$, $F = a_7$, the first Fredkin gate selects the input a_7, which is the sign bit. Therefore, the first selection process of the selection unit achieved the function to make a selection from logical right shift or arithmetic right shift. Then the selected input F is propagated to the next Feynman gate, this Feynman gate together with the other three Feynman gates yields a total number of 5 copies of F. Four out of them will be used in the next progress of the selection unit to complete the overall selection of the three operations and the other one will be used for the next stage selection. The second progress of the selection unit will select the inputs from F and the initial inputs a_3, a_2, a_1, a_0. The final selected inputs are defined as s_3, s_2, s_1, s_0. For $m = 0, 1, 2, 3$ the final selected inputs have the following equation:

$$s_m = \bar{t}_2 a_m + t_2 F \tag{4.41}$$

Thus for $t_2 = 0$, $s_m = F$, the selection unit then selects either the logical right shift or the arithmetic right shift depending on the the value of F. For $t_2 = 1$, $s_m = a_m$, then the selection unit selects the operation of right rotation.

2. Stage 1
   For the first stage, the inputs are the selected inputs by the selection unit a_{n-1}, a_{n-2}, ... $a_{2^{q-1}}$, $s_{2^{q-1}-1}, s_{2^{q-1}-2}$, ... s_2, s_1, s_0. The outputs of the first stage are defined as $O_{1,m}$:

$$O_{1,m} = \begin{cases} \overline{b}_{q-1}a_m + b_{q-1}s_{m+2^{q-1}-n} & \text{if } \ m \geq 2^{q-1}-1 \\ \\ \overline{b}_{q-1}s_m + b_{q-1}a_{m+2^{q-1}} & \text{if } \ m < 2^{q-1}-1 \end{cases} \tag{4.42}$$

b_{q-1} is the control signal of the first stage. The outputs are:

$$O_{1,m} = \begin{cases} a_m & \text{if } \ m \geq n-2^{q-1} \\ \\ s_m & \text{if } \ m < n-2^{q-1} \end{cases} \tag{4.43}$$

If $b_{q-1} = 0$. Else if $b_{q-1} = 1$, the outputs of the first stage will rotate for 2^{q-1} bits to the right. That is:

$$O_{1,m} = \begin{cases} s_{m+2^{q-1}-n} & \text{if } \ m \geq 2^{q-1}-1 \\ \\ a_{m+2^{q-1}} & \text{if } \ m < 2^{q-1}-1 \end{cases} \tag{4.44}$$

3. Stage p $(1 \leq p \leq q)$
   For the p^{th} stage of the design of a universal right shifter, the initial inputs are the outputs from the previous stage $O_{p-1,n-1}$, $O_{p-1,n-2}$, ... $O_{p-1,2}, O_{p-1,1}, O_{p-1,0}$. Since the primarily selected input F can be applied as a global variable and can be copied by Feynman gates, the p^{th} stage does not require a double selection unit as the first stage. The selection unit of the p^{th} stage $(p \neq 1)$ will only select from F and the inputs O_m, where $0 \leq m \leq n-2^{q-p}$. It requires 2^{q-p} Fredkin gates and 2^{q-p} Feynman gates to achieve the selection. The selected inputs are defined as $s_{2^{q-p}-1}, s_{2^{q-p}-2}$, ... s_2, s_1, s_0:
   $$s_m = \overline{t}_{q-p}O_{p-1,m} + t_{q-p}F \ \text{ for } \ 0 \leq m \leq n-1-2^{q-p} \tag{4.45}$$
   When t_{q-p} is set to 0, $s_m = O_{p-1,m}$, the actual inputs will keep unchanged as the previous outputs $O_{p-1,n-1}, O_{p-1,n-2}$, ... $O_{p-1,2}, O_{p-1,1}$. If t_{q-p} is set to 1, then $s_m = F$. So that there are three possibilities for the leftmost 2^{q-p} bits: 0, a_{n-1}, $O_{p-1,m}$. The outputs are defined as:

$$O_{p,m} = \begin{cases} \overline{b}_{q-p}O_{p-1,m} + b_{q-p}s_{m+2^{q-p}-n} & \text{if } \ m \geq n-2^{q-p} \\ \\ \overline{b}_{q-p}s_m + b_{q-p}O_{p-1,m+2^{q-p}} & \text{if } \ m < n-2^{q-p} \end{cases} \tag{4.46}$$

If $b_{q-p} = 0$, the outputs of the p^{th} stage will keep the same as the inputs:

$$O_{p,m} = \begin{cases} O_{p-1,m} & \text{if } \ m \geq n - 2^{q-p} \\ \\ s_m & \text{if } \ m < n - 2^{q-p} \end{cases} \tag{4.47}$$

else if $b_{q-p} = 1$, the outputs of the p^{th} stage will right shift for 2^{q-p} bits to the right. That is:

$$O_{p,m} = \begin{cases} s_{m+2^{q-p}-n} & \text{if } \ m \geq n - 2^{q-p} \\ \\ O_{p-1,m+2^{q-p}} & \text{if } \ m < n - 2^{q-p} \end{cases} \tag{4.48}$$

Thus at the final outputs side, as it is shown in Table 4.4, the three different operations can be realized. And for different states of the control signals fro each stage, the universal right shifter can complete the three operations for any bits from 0 to $n-1$ according to the different settings of all the control signals. To make it visualized, a design of $(8, 3)$ SCRL-based universal right shifter is shown in Figure 4.10. It has 3 shifting or rotating stages with the initial inputs as $a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0$. The corresponding final outputs are $O_7, O_6, O_5, O_4, O_3, O_2, O_1, O_0$. To simplify the expressions and equations, the outputs for the first stage are defined as $k_7, k_6, k_5, k_4, k_3, k_2, k_1, k_0$, and the outputs for stage 2 are defined as $j_7, j_6, j_5, j_4, j_3, j_2, j_1, j_0$. And also to separate the selected inputs from each stage, define the selected input for stage 1 as s_3, s_2, s_1, s_0, the selected inputs for stage 2 as x_1, x_0, the selected inputs for stage 3 as y_0. The selected inputs are selected by the following equations:

$$s_i = \bar{t}_2 a_i + t_2 F \ \text{ for } \ 0 \leq m \leq 3 \tag{4.49}$$

$$x_m = \bar{t}_1 k_m + t_1 F \ \text{ for } \ 0 \leq m \leq 1 \tag{4.50}$$

$$y_m = \bar{t}_0 j_m + t_0 F \ \text{ for } \ m = 0 \tag{4.51}$$

The first stage shift unit will either right shift the initial inputs $a_7, a_6, a_5, s_4, s_3, s_2, s_1, s_0$ for 4 bits with the control signal $b_2 = 1$ or keep them unchanged with $b_2 = 0$. The shift unit of stage 2 will either right shift its inputs for 2 bits with $b_1 = 1$ or keep them unchanged with $b_1 = 0$. At last the shift unit of stage 3 will either right shift its inputs for 1 bits with $b_0 = 1$ or keep them unchanged with $b_0 = 0$. For each stage, the outputs are defined as:

$$k_i = \begin{cases} \bar{b}_2 a_i + b_2 s_{i-4} & \text{if } \ i \geq 4 \\ \\ \bar{b}_2 s_i + b_2 a_{i+4} & \text{if } \ i < 4 \end{cases} \tag{4.52}$$

$$j_i = \begin{cases} \overline{b}_1 a_i + b_1 x_{i-6} & \text{if } \; i \geq 6 \\ \\ \overline{b}_1 x_i + b_1 a_{i+2} & \text{if } \; i < 6 \end{cases} \tag{4.53}$$

$$O_i = \begin{cases} \overline{b}_0 a_i + b_0 y_{i-7} & \text{if } \; i \geq 7 \\ \\ \overline{b}_0 y_i + b_0 a_{i+1} & \text{if } \; i < 7 \end{cases} \tag{4.54}$$

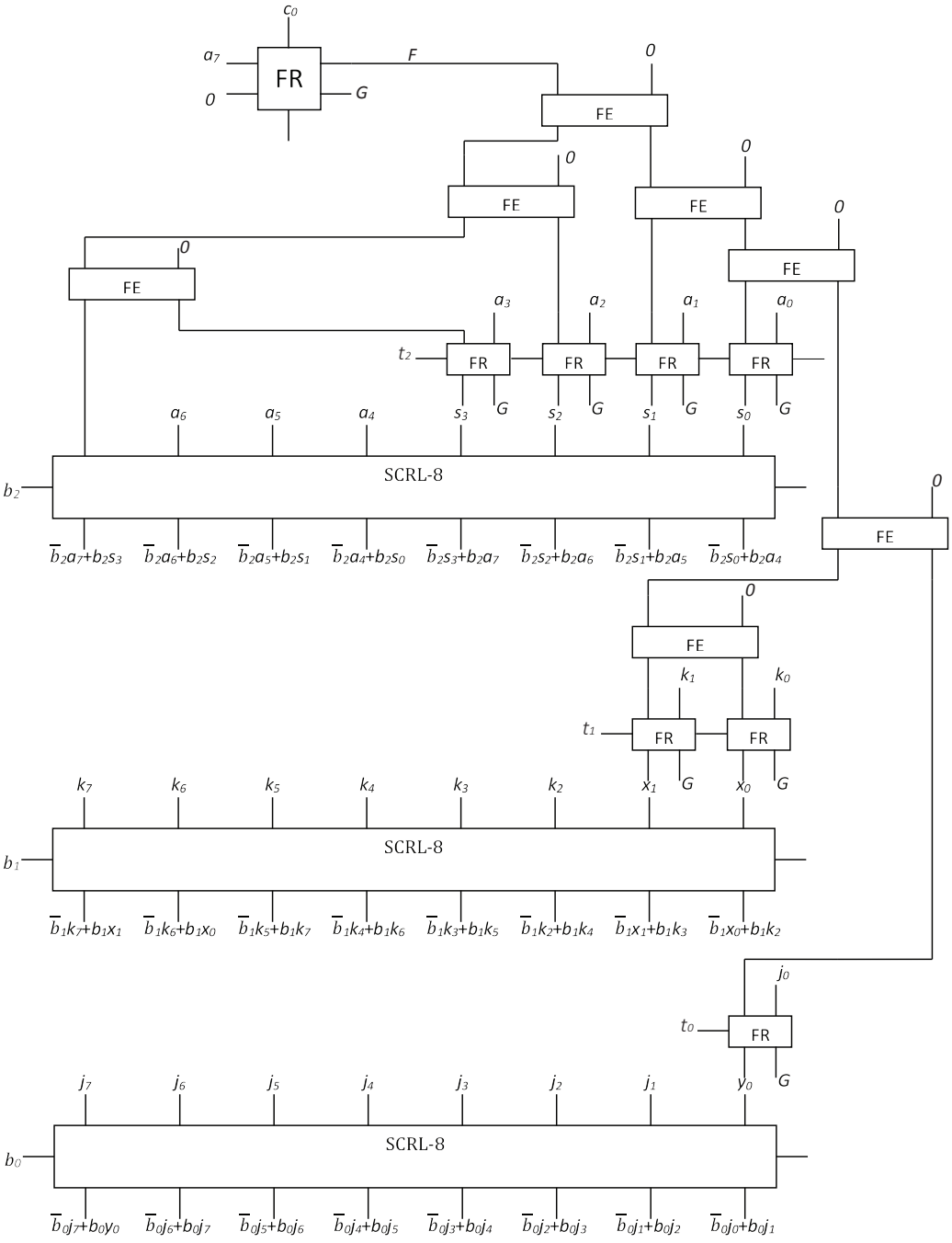


Figure 4.10: An SCRL $(8, 3)$ universal right shifter (FR: Fredkin gate, FE: Feynman gate, G: garbage outputs)

Table 4.5 is a list of all possible bits data movement of three different operations.

Table 4.5: Operations of all possible bits in $(8, 3)$ universal right shifter

t_2	t_1	t_0	c_0	b_2	b_1	b_0	Operations
0	0	0	0	0	0	0	0 bit arithmetic right shift
0	0	1	0	0	0	1	1 bit arithmetic right shift
0	1	0	0	0	1	0	2 bit arithmetic right shift
0	1	1	0	0	1	1	3 bit arithmetic right shift
1	0	0	0	1	0	0	4 bit arithmetic right shift
1	0	1	0	1	0	1	5 bit arithmetic right shift
1	1	0	0	1	1	0	6 bit arithmetic right shift
1	1	1	0	1	1	1	7 bit arithmetic right shift
0	0	0	1	0	0	0	0 bit logical right shift
0	0	1	1	0	0	1	1 bit logical right shift
0	1	0	1	0	1	0	2 bit logical right shift
0	1	1	1	0	1	1	3 bit logical right shift
1	0	0	1	1	0	0	4 bit logical right shift
1	0	1	1	1	0	1	5 bit logical right shift
1	1	0	1	1	1	0	6 bit logical right shift
1	1	1	1	1	1	1	7 bit logical right shift
0	0	0	/	0	0	0	0 bit right rotation
0	0	0	/	0	0	1	1 bit right rotation
0	0	0	/	0	1	0	2 bit right rotation
0	0	0	/	0	1	1	3 bit right rotation
0	0	0	/	1	0	0	4 bit right rotation
0	0	0	/	1	0	1	5 bit right rotation
0	0	0	/	1	1	0	6 bit right rotation
0	0	0	/	1	1	1	7 bit right rotation

Thus if the control signal $c_0 = 0$, $t_2 + t_1 + t_0 \neq 0$ the universal performs as an arithmetic right shifter and if $c_0 = 1$, $t_2 + t_1 + t_0 \neq 0$ it performs as a logical right shifter. If $t_2 + t_1 + t_0 = 0$, no matter what state c_0 is at, the universal right shifter will perform as a right rotator. And Figure 4.11 is the simulation result of SCRL universal right shifter.

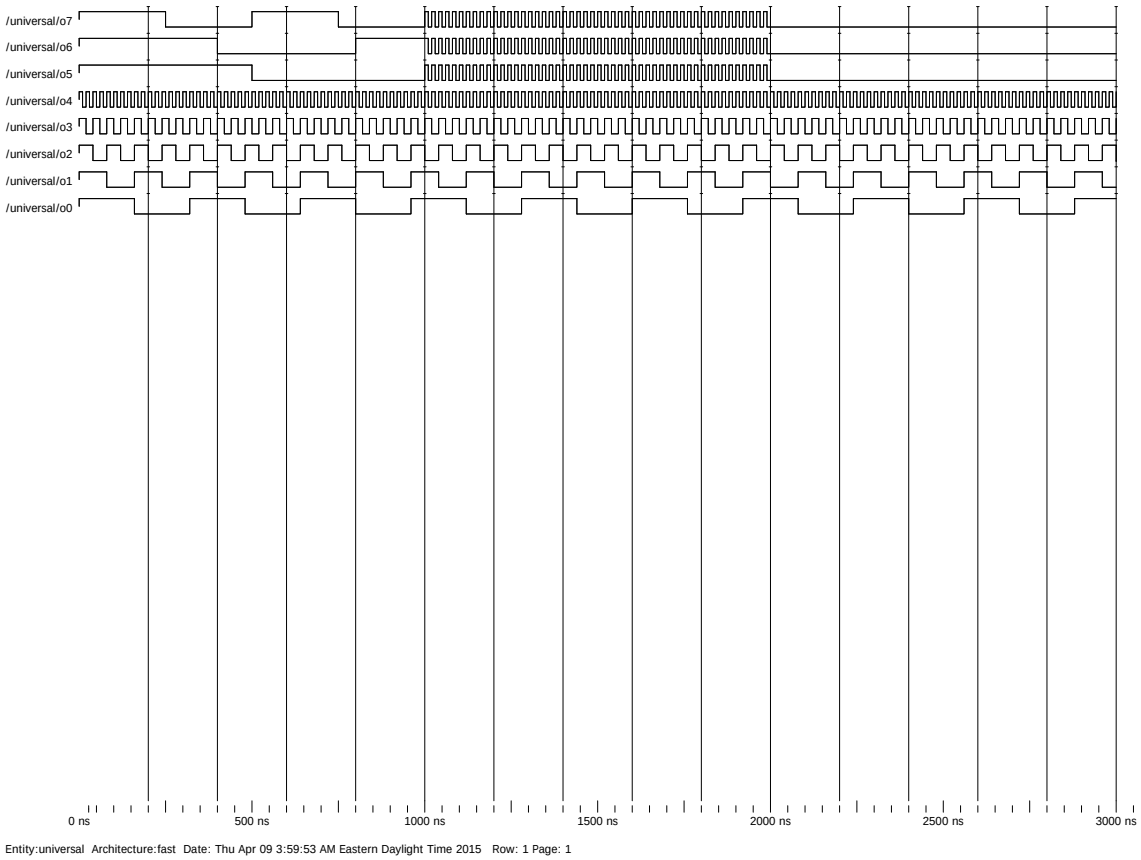


Figure 4.11: Simulation result of SCRL universal right shifter

4.5 Design Methodology of Universal Bidirectional Shifter and Implementation in SCRL Gates

In the previous sections, the involved designs are exclusively right directional. Recall that in Table 2.4, left directional operations are also mentioned. However, the only difference between the two directional rotators is the operating direction. And there is the same difference between the two directional operations of logical shifter. For the arithmetic right shifter and the arithmetic left shifter, there is another difference other than the shifting direction. The left shifted bits of data are replaced by 0 while the leftmost bit is kept as the sign bit. Designing a logic circuit that is capable of operating all the 6 functions in Table 2.4 is another goal of this thesis. The main purpose is to design a unit that can realize a order reverse operation. Thinking that if the order of the initial inputs are reversed and then reversed back after the right directional operation unit, the final output will be just the same as the ones of left directional operation except the arithmetic shift. It is not difficult to find out that ignoring the first bit of data, the arithmetic left shifter is operating an logical left shifter. So that it is executable to include the bidirectional arithmetic shifter in the double-reverse design universal shifter by adding another Fredkin gate to the second reverse operation to make a selection from the sign bit a_{n-1} and the output of the last function

operation stage f_0. Figure 4.12 is an example for a reverse unit of a universal shifter.

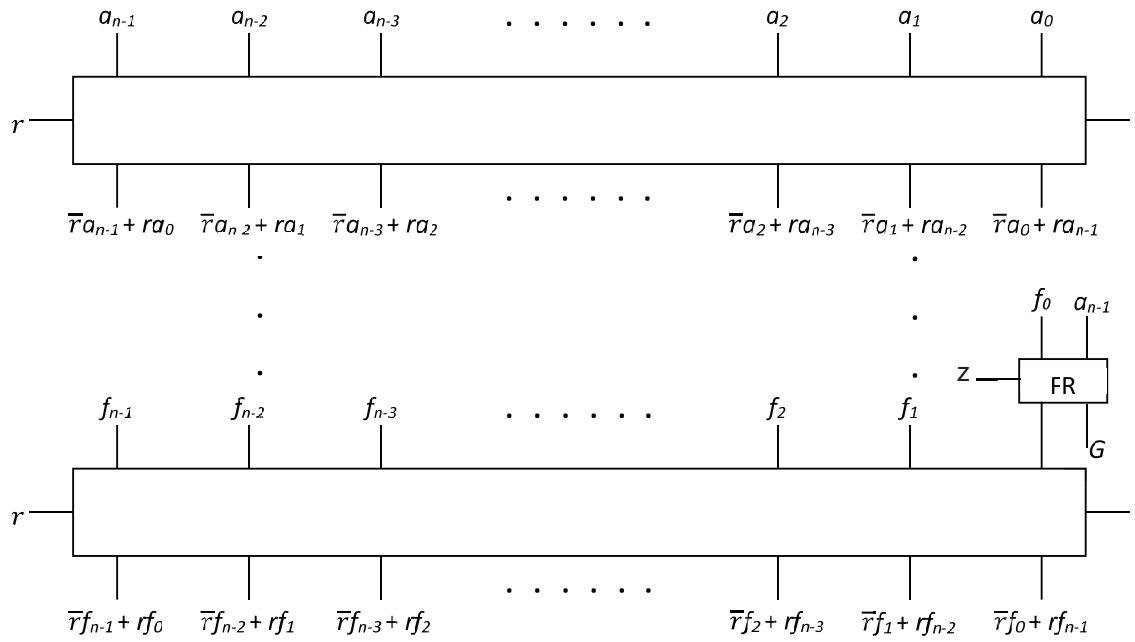


Figure 4.12: Reverse unit

The variable r is the control signal of the reverse unit. Suppose the inputs to the reverse unit are $a_{n-1}, a_{n-2}, a_{n-3}$, ... a_2, a_1, a_0, with the control signal $r = 1$, the first reverse operation will reverse them into the order of a_0, a_1, a_2, ... $a_{n-3}, a_{n-2}, a_{n-1}$ and the second reverse operation will reverse the data from the last operation stage $f_{n-1}, f_{n-2}, f_{n-3}$, ... f_2, f_1, f_0 to f_0, f_1, f_2, ... $f_{n-3}, f_{n-2}, f_{n-1}$. With $r = 0$, the reverse unit will not operate reverse function. Thus the logic circuit is enabled to make bidirectional operations. When $r = 1$, it performs a left operation and when $r = 0$, it performs a right operation. Table 4.6 is a list of the different operations with different states of the control signals. $T = t_{q-1} + t_{q-2} + ... + t_1 + t_0$.

Table 4.6: Operations of the universal bidirectional shifter

z	r	T	c_0	Operations
0	0	0	0	Right rotation
0	0	0	1	Right rotation
0	0	1	0	Arithmetic right shift
0	0	1	1	Logical right shift
0	1	0	0	Left rotation
0	1	0	1	Left rotation
0	1	1	1	Logical left shift
1	1	1	1	Arithmetic left shift

The control signal z is a switch to turn on the arithmetic left shift mode. It always work

at with $r = 1$, $t_2 + t_1 + t_0 = 1$, $c_0 = 1$, which is a logical right shift in the previous function operation.

1. Right rotation
   When the control bits are at the state of all 0s, the control signal r will not turn the reverse unit on, so that the logic circuit is a right directional operator. And then the selection unit selects a operation of right rotator with $T = 0$. The contol signals will always select the initial inputs. As a result, the universal bidirectional shifter performs right rotate operation. Suppose there is a serial inputs a_{n-1}, a_{n-2}, ... a_2, a_1, a_0. After the first part of the reverse unit, this serial inputs will keep unchanged since the reverse unit is off ($r = 0$). The selection unit selects an operation of right rotation, so that for each stage, the inputs will always be the same as the outputs from the upper stage. The rotating bit is controlled by b_{q-1}, which is discussed in the previous designs.

2. Arithmetic right shift
   While $T = 1$, $T = t_{q-1} + t_{q-2} + ... + t_1 + t_0$, it means that at least one of t_{q-1}, t_{q-2}, ... t_1, t_0 is set to 1. So that at least one of the shifting stages will have the input from the first Fredkin gate which select the input form 0 and the sign bit a_{n-1}. Then with $c_0 = 0$, the first Fredkin gate in the selection unit selects the input as the sign bit a_{n-1}. Thus the universal bidirectional shifter performs as an arithmetic right shifter.

3. Logical right shit
   As it is mentioned previously, when $T = 1$, at least one of the shifting stages will obtain the input from the first Fredkin gate which will be either 0 or a_{n-1}. At $c_0 = 1$, the first selection part of the selection unit will select 0 as the input. Then controlled by b_{q-1}, at least one of the shifting operation stage will have a certain number of inputs as 0. Therefore, the universal bidirectional shifter performs as a logical right shifter.

4. Left rotation
   To make a left rotation, it is required that $r = 1$ so that the reverse unit is turn on to make a left directional rotation. If the initial inputs are a_{n-1}, a_{n-2}, ... a_2, a_1, a_0, after the first part of the reverse unit, they are reversed to a_0, a_1, a_2, ... a_{n-2}, a_{n-1}. Suppose that the logic circuit will perform a 2 bits left rotation, then after the operation unit, the outputs are $a_{n-2}, a_{n-1}, a_0, a_1$, ... a_{n-4}, a_{n-3}. After reversed by the second part of the reverse unit, the final outputs are a_{n-3}, a_{n-4}, ... $a_1, a_0, a_{n-2}, a_{n-1}$. Thus the circuit operates a 2 bits left rotation.

5. Logical left shift
   For a logical left shift, the direction control signal is required to be set to 1. Similarly to left rotation, the operation unit should perform a logical right shift so that $T = 1$ and $c_0 = 1$. For the same inputs as left rotation, suppose the logic circuit is going to perform a 2 bits logical left shift, they are reversed by the first part of the reverse unit. The operation unit yields the outputs as $0, 0, a_0, a_1$, ... a_{n-4}, a_{n-3}. At last the reverse unit yields the final outputs as a_{n-3}, a_{n-4}, ... $a_1, a_0, 0, 0$. The logic circuit then realizes an operation of 2 bits logical left shift.

6. Arithmetic left shift
   Arithmetic left shift is different from left rotation and logical left shift. The operation

unit is not an arithmetic right shifter but a logical right shifter so that there are 0s yielded from the operation unit. For an arithmetic left shift, control signal z is required to be set to 1 so that at prior to the last reverse, the Fredkin gate selects the fist last bit, which will be reversed to the first after the reverse, as the sign bit a_{n-1}. Again, suppose the logic circuit is going to perform a 2 bits arithmetic left shift with the same inputs, after reversed by the first part of the reverse unit, the operation unit performs a logical right shift and yields the outputs as $0, 0, a_0, a_1,$ ..... a_{n-4}, a_{n-3}. Then the z controlled Fredkin gate selects the last bits as a_{n-1}. After the second reverse, the final outputs are $a_{n-1}, a_{n-4}, a_{n-5},$ ..... $a_1, a_0, 0, 0$. Thus the universal bidirectional shifter realizes a 2 bits arithmetic left shift.

Figure 4.13 is an example of an $(8, 3)$ universal bidirectional shifter. The initial inputs are $i_7, i_6, i_5, i_4, i_3, i_2, i_1, i_0$. The first reverse yields $a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0$. And the operation unit yields $f_7, f_6, f_5, f_4, f_3, f_2, f_1, f_0$. The z controlled Fredkin yields h. And Figure 4.14 is the simulation result of SCRL universal bidirectional shifter.

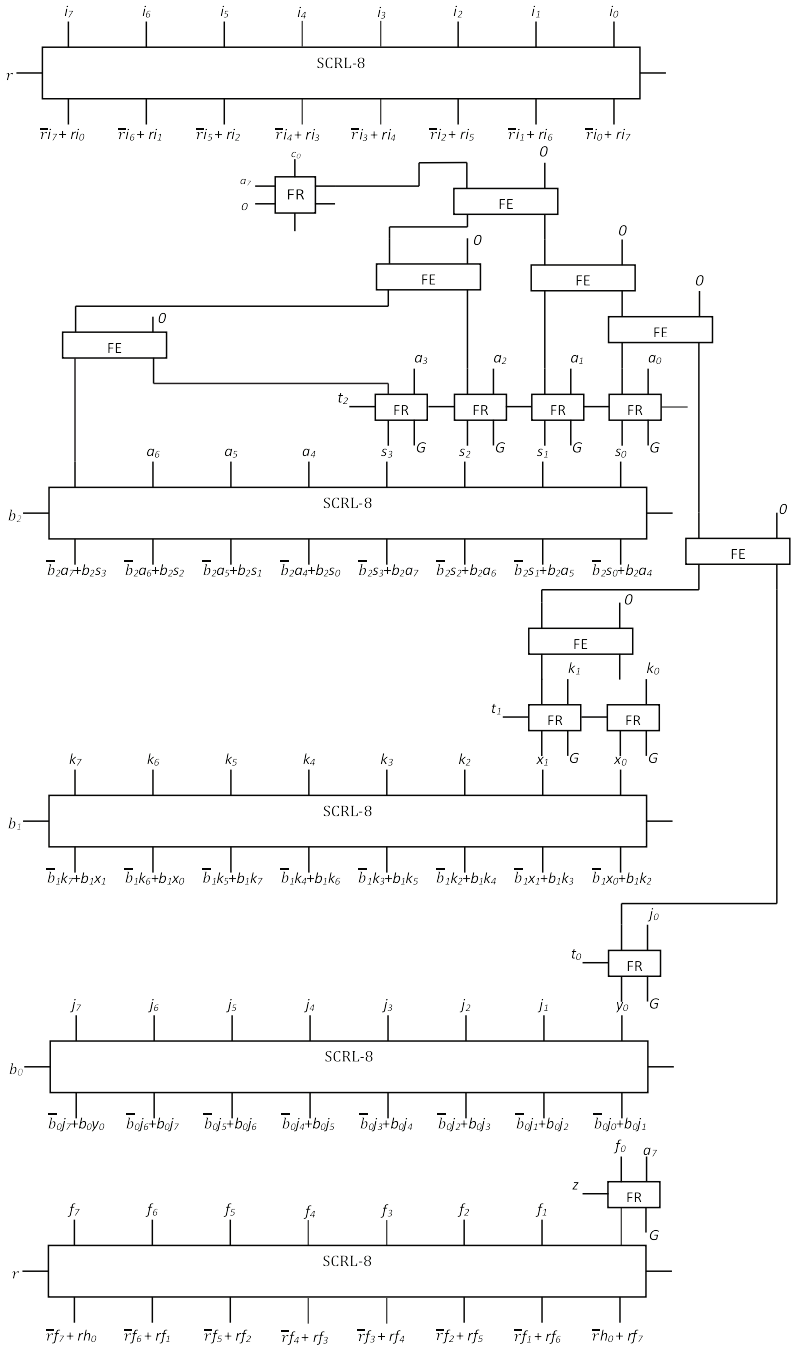


Figure 4.13: An $(8, 3)$ universal bidirectional shifter (FR: Fredkin gate, FE: Feynman gate, G:garbage outputs)

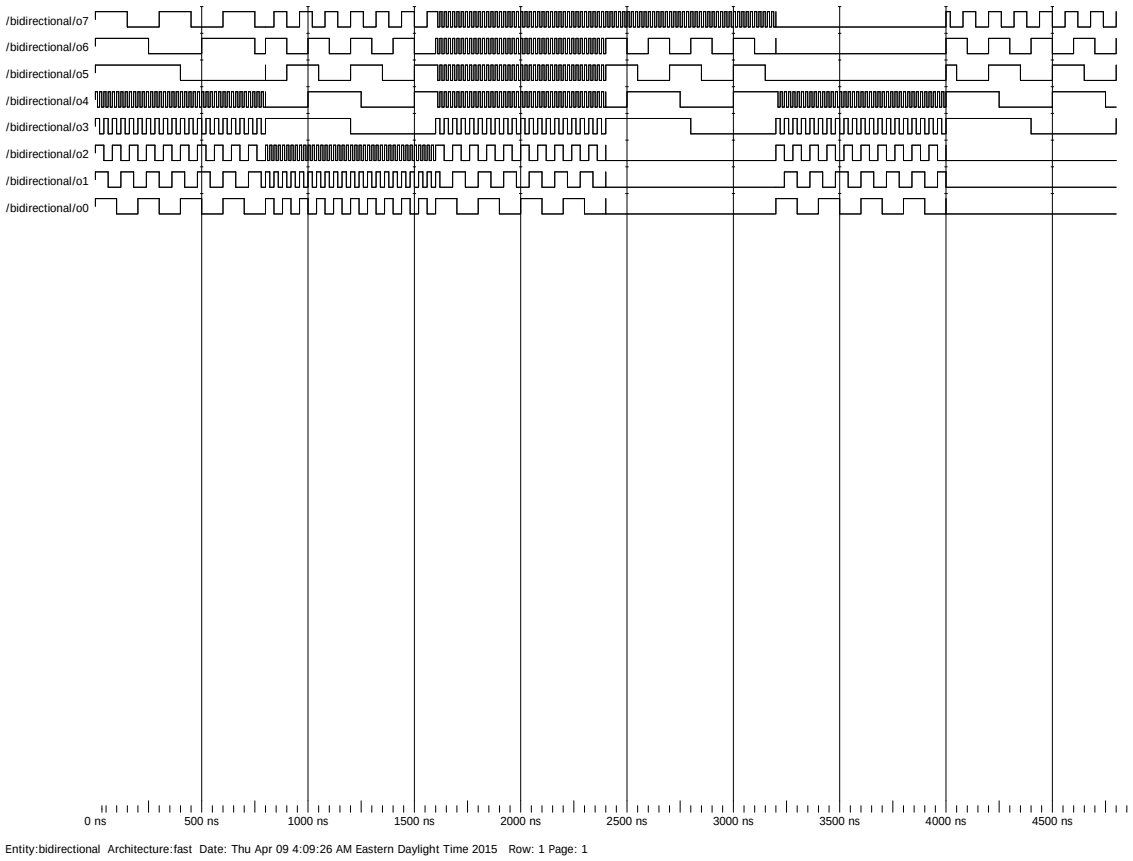


Figure 4.14: Simulation result of SCRL universal bidirectional shifter

Chapter 5

Evaluations

In this thesis, we have evaluated the proposed designs of the SCRL-based reversible barrel shifter in terms of the amount of garbage and ancilla bits and the qca cost.

5.1 Garbage Outputs

Garbage outputs is one of primary metric that needs to be optimized when designing the reversible logic circuits. Garbage outputs are the outputs that are needed for reversibility but are not responsible for any useful operations.

5.1.1 Garbage Outputs in SCRL Reversible Right Rotator

As it is shown in Figure 4.1, the SCRL-based reversible right rotator do not yield any garbage outputs. *Hence, the SCRL reversible right rotator is a circuit that is free of garbage outputs.* To the best of our knowledge, the existing design in [33] is the most efficient design of reversible right rotator. Hence, we have compared our proposed SCRL-based right rotator with the design in [33]. Figure 5.1 plots of the number of the garbage outputs in [33].

Table 5.1 is a comparison of the garbage outputs between the SCRL-based design of reversible right rotator and the existing design in [33].

Table 5.1: Garbage outputs in right rotators

	$(4,2)$	$(8,3)$	$(16,4)$	$(32,5)$	$(64,6)$	. . .	(n,q)
SCRL design	0	0	0	0	0	. . .	0
Shamsujjoha design	2	3	4	5	6	. . .	q
Improvement ratio S%	100	100	100	100	100	. . .	N/A
Improvement ratio S% is the improvement ratio compared to Shamsujjoha design [33].							

The comparison results shows that the proposed SCRL-based design of right rotator is a great improvement compared to the design in [33].

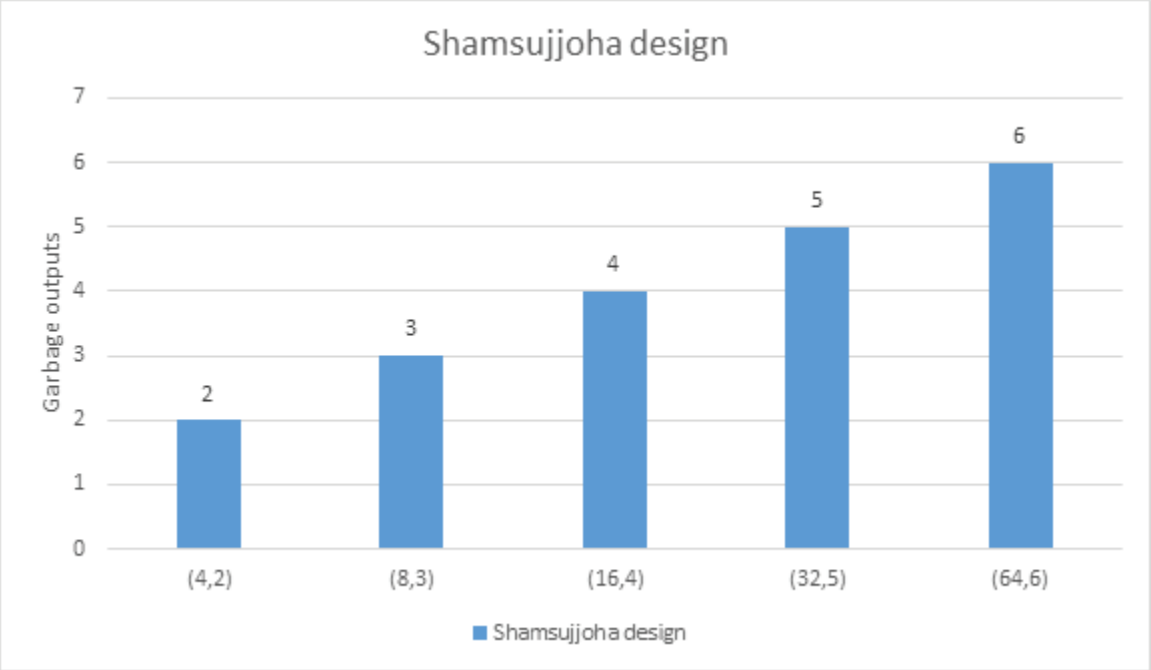


Figure 5.1: Garbage outputs in reversible right rotators

5.1.2 Garbage Outputs in SCRL Reversible Logical Right shifter

As mentioned in the design methodology of the SCRL-based reversible logical right shifter, for the p^th of an (n, q) circuit, a number of 2^{q-p} Fredkin gates are required to form the selection unit and each Fredkin gate will produce one garbage output. We have compared the proposed SCRL-based reversible logical right shifter with the existing reversible designs in [19, 33]. Table 5.2 is a comparison of the garbage outputs of different designs, SCRL design, Kotiyal design and Shamsujjoha design [19, 33].

Table 5.2: Garbage outputs in logical right shifters

	$(4,2)$	$(8,3)$	$(16,4)$	$(32,5)$	$(64,6)$	. . .	(n,q)
SCRL design	3	7	15	31	63	. . .	n-1
Kotiyal design	10	27	68	165	390	. . .	$(n+1) \times q$
Shamsujjoha design	8	24	64	160	384	. . .	$n \times q$
Improvement ratio K%	70.00	74.07	77.94	81.12	83.84	. . .	N/A
Improvement ratio S%	62.50	70.83	76.56	80.62	83.59	. . .	N/A

Figure 5.2 is a chart that shows the increasing trend of garbage outputs as n, q increases.

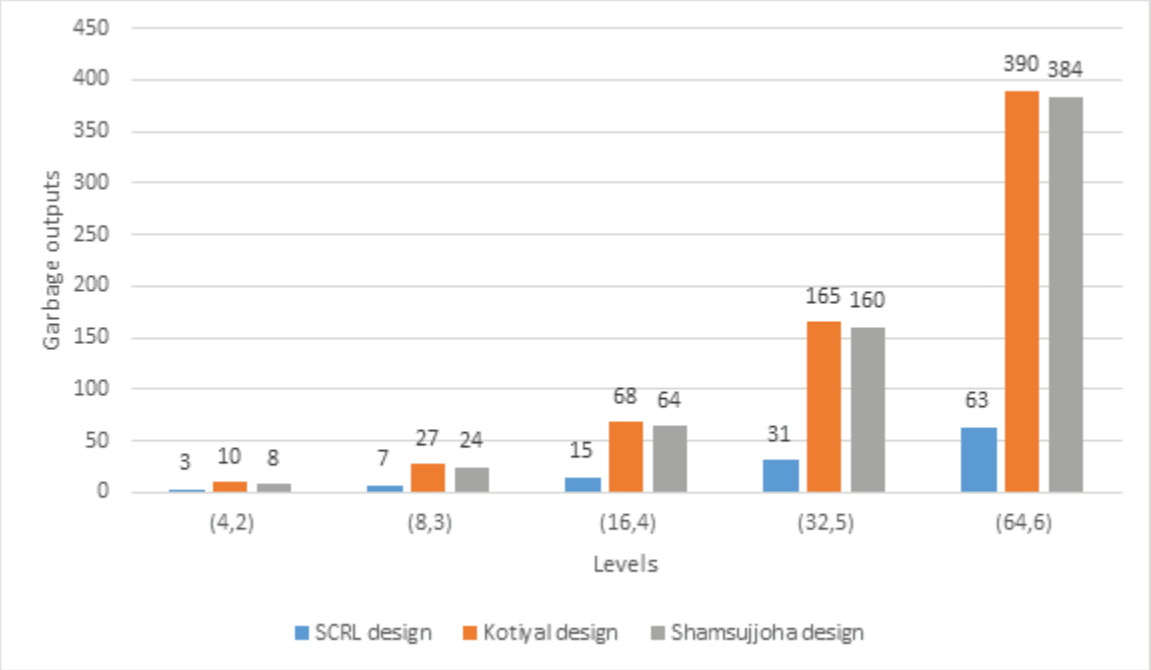


Figure 5.2: Garbage outputs in reversible logical right shifters

5.1.3 Garbage Outputs in SCRL Reversible Arithmetic Right Shifter

The SCRL-based design of reversible arithmetic right shifter requires the same number of Fredkin gates as the logical right shifter. Since Feynman gates do not yields any garbage outputs, the SCRL reversible arithmetic right shifter has the same bits of garbage outputs as the logical right shifter. Table 5.3 shows the garbage outputs of the arithmetic right shifter in a detailed to stage measurement.

Table 5.3: Garbage outputs in arithmetic right shifter

	$n = 4$	$n = 8$	$n = 16$	$n = 32$	$n = 64$
$k = 2$	3	6	12	24	48
$k = 3$	NA	7	14	28	56
$k = 4$	NA	NA	15	30	60
$k = 5$	NA	NA	NA	31	62
$k = 6$	NA	NA	NA	NA	63

5.1.4 Garbage Outputs in SCRL Reversible Universal Right Shifter

The SCRL-based design of reversible universal right shifter requires an additional Fredkin gate to construct the three variables selection unit. Thus for the whole circuit, it yields one more garbage outputs. As a result, there are n Fredkin gates in a SCRL reversible universal right shifter and meanwhile n garbage outputs are yielded in total. We have the comparisons between SCRL-based design and the existing design in [19]. Table 5.4 is a

comparison of the garbage outputs in reversible universal right shifters in SCRL design and existing Kotiyal design [19].

Table 5.4: Garbage outputs in universal right shifters

	$(4,2)$	$(8,3)$	$(16,4)$	$(32,5)$	$(64,6)$	. . .	(n,q)
SCRL design	4	8	16	32	64	. . .	n
Kotiyal design	16	37	86	199	456	. . .	$(n+1) \times q + n + 2$
Impv ratio K%	75.00	78.38	81.40	83.92	85.96	. . .	N/A
Improvement ratio K% is the improvement ratio compared to Kotiyal design [19].							

Figure 5.3 has a direct view of the increasing trend of garbage outputs with the increasing n, q values.

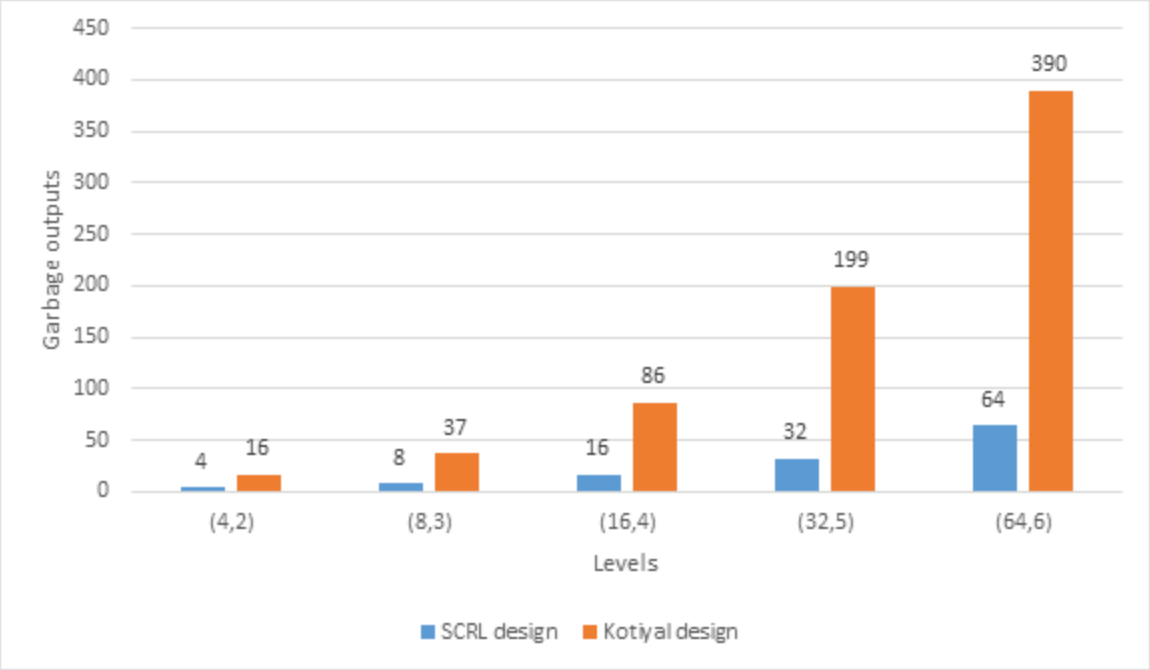


Figure 5.3: Garbage outputs in reversible universal right shifters

5.1.5 Garbage Outputs in SCRL Reversible Universal Bidirectional Shifter

The SCRL-based reversible universal bidirectional shifter has an additional reverse unit for direction reversing. To make an arithmetic left shift, an additional Fredkin gate is required at the second reverse of the reverse unit as shown in Figure 4.13. Hence, the SCRL-based reversible universal bidirectional shifter has $(n+1)$ garbage outputs in total. We have compared the SCRL-based universal bidirectional shifter with the existing designs in [17, 19, 26]. Table 5.5 shows the comparison of garbage outputs of the SCRL design and some other existing designs [17, 19, 26] (M is the Mitra design [26], K is the Kotiyal design [19] and H is the Hosseininia design [17]).

Table 5.5: Garbage outputs in universal bidirectional shifter designs

	$(4,2)$	$(8,3)$	$(16,4)$	$(32,5)$	$(64,5)$
SCRL design	5	9	17	33	65
Kotiyal design	17	38	87	200	457
Mitra design	11	17	27	45	79
Hosseininia design	32	74	164	358	776
Impv ratio K%	70.59	76.32	80.46	83.50	85.78
Impv ratio M%	54.54	47.06	37.04	26.67	17.72
Impv ratio H%	84.38	87.84	89.63	90.78	91.62

Figure 5.4 has a direct view of the increasing trend of garbage outputs with the increasing n, q values.

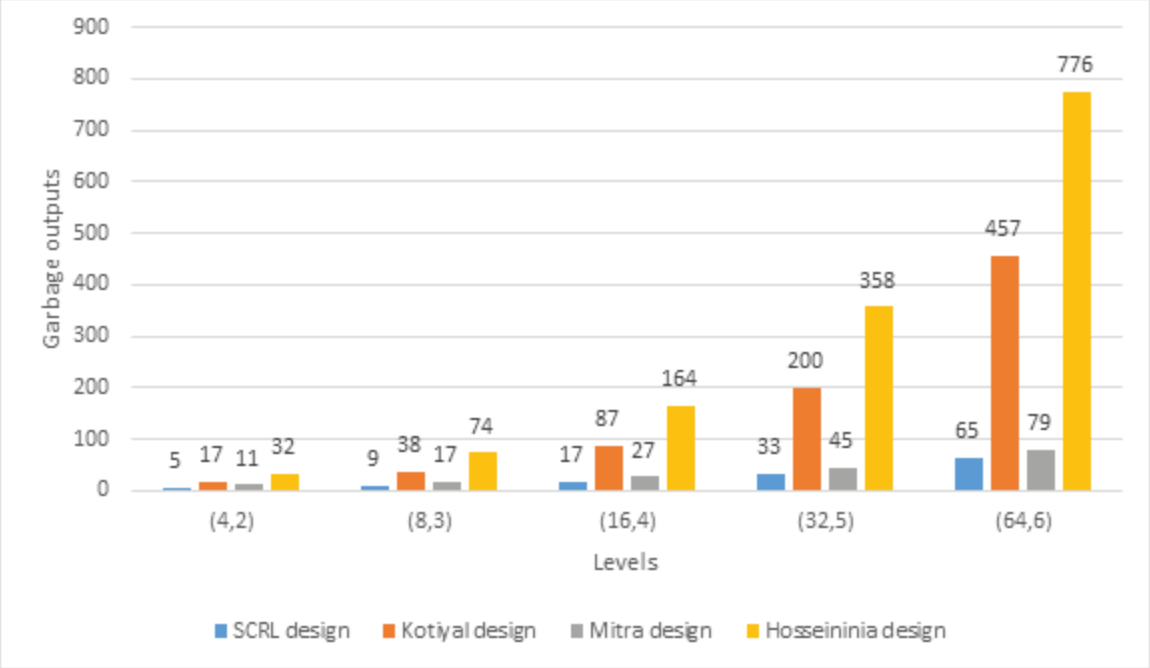


Figure 5.4: Garbage outputs in reversible universal bidirectional shifters

5.2 Comparison of Garbage Outputs

Table 5.1 shows that the SCRL design reversible right rotator does not yield any garbage outputs so that the improvement ratios of the Shamsujjoha design are all 100%. Thus the SCRL design reversible right rotaor is perfect from the garbage outputs evaluation. Table 5.2 is a comparison table of garbage outputs for reversible logical right shifters. Improvement ratio K% means the improvement ratio of Kotiyal design and Improvement ratio S% means the impovement ratio of Shamsujjoha design. The improvement ratios show that the SCRL design has improved the number of garbage outputs in a decent extent. And the ratio grows as the inputs level grows. It grows from 70.00% at $(4,2)$ inputs level to 83.84% at $(64,6)$

inputs level for Kotiyal design. And it grows from 62.50% at $(4, 2)$ inputs level to 83.59% at $(64, 6)$ inputs level for Shamsujjoha design. So that it is even more efficient when designing extensive inputs logical right shifters. Table 5.4 shows that the improvement ratio also grows with the growing inputs level of the reversible universal right shifter. It means that for the reversible universal right shifter, the SCRL design has also improved quite a lot. Table 5.5 is the most important comparison since there are multiple designs in reversible universal bidirectional shifters. The Kotiyal and Hosseininia improvement ratio show that the SCRL design has improved the garbage outputs. Even for Mitra design, which is the new 2015 design, the SCRL design has also reduced the garbage outputs. Even though the improvement ratio of Mitra design is decreasing, the garbage outputs of SCRL design will always be less than those of Mitra design as it is indicated in the (n, q) inputs level that $n + 2q + 3$ is larger than $n + 1$.

5.3 Ancilla Inputs

Ancilla inputs refer to the constant inputs in the logic circuits. The amount of ancilla inputs is another important criteria to evaluate the reversible logic. A large number of ancilla inputs is always a limitation to fabricate a quantum computer with large number of inputs.

5.3.1 Ancilla Inputs in SCRL Reversible Right Rotator

The SCRL-based reversible right rotator are built by all SCRL gates. All the inputs are the given variables and control signals that are not considered as ancilla inputs. That is to say that there are no ancilla inputs in SCRL-based reversible right rotator.

5.3.2 Ancilla Inputs in SCRL Reversible Logical Right Shifter

A logical right shifter requires copies of 0s to replace the shifted bits. As shown in Figure 4.3, the selection unit built by Fredkin gates requires 0s as the constant inputs. As a result, each Fredkin gates in use is a source of an ancilla input. In an (n, q) SCRL-based reversible logical right shifter, a number of $n - 1$ ancilla inputs are produced. We have also compared the SCRL-based design of reversible logical right shifter with the existing design in [19]. Table 5.6 shows the number of ancilla inputs at different (n, q) values in SCRL and another existing Kotiyal design [19].

Table 5.6: Ancilla inputs in logical right shifters

	$(4, 2)$	$(8, 3)$	$(16, 4)$	$(32, 5)$	$(64, 6)$	. . .	(n, q)
SCRL design	3	7	15	31	63	. . .	n-1
Kotiyal design	8	24	64	160	384	. . .	$n \times q$
Improvement ratio K%	62.50	70.83	76.56	80.625	83.59	. . .	N/A
Improvement ratio K% is the improvement ratio compared to Kotiyal design [19].							

Figure 5.5 has a direct view of the increasing trend of ancilla inputs with the increasing n, q values.

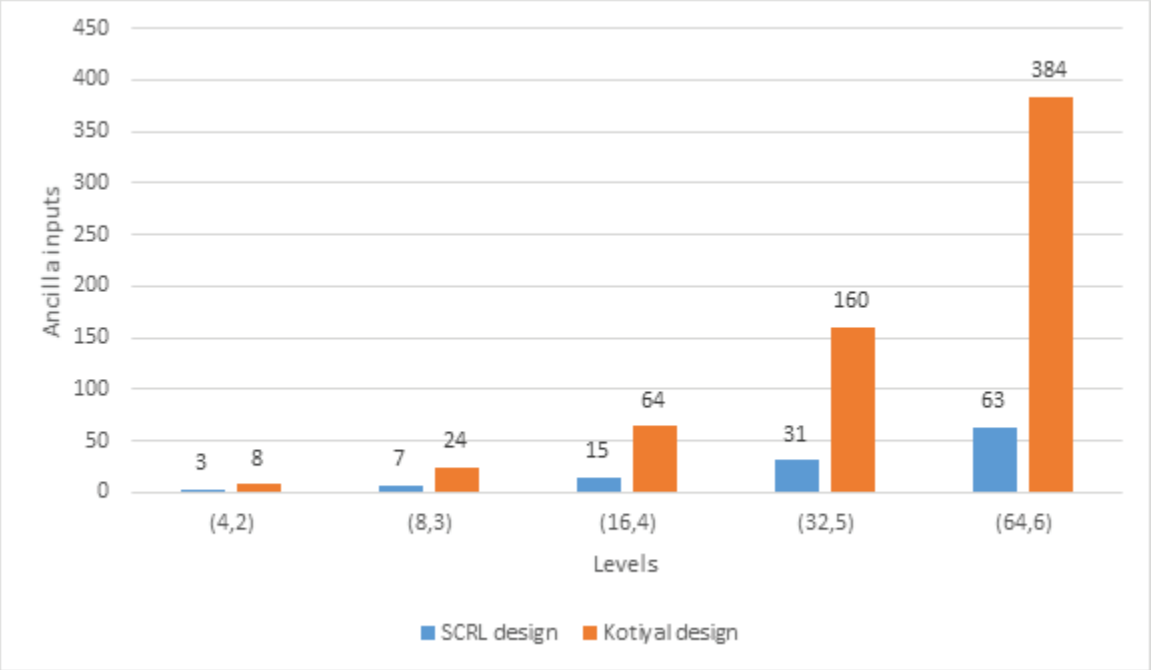


Figure 5.5: Ancilla inputs in reversible logical right shifters

5.3.3 Ancilla Inputs in SCRL Reversible Arithmetic Right Shifter

In the SCRL-based design of reversible arithmetic right shifter, the source of ancilla inputs is the Feynman gate. In order to realize an arithmetic right shift, the logic circuit requires copies of the sign bit. A Feynman gate performs a copy function when the second input is set to 0. The constant 0s are the ancilla inputs and each Feynman gate in use produces one ancilla input. The number of Feynman gates in an (n, q) SCRL arithmetic right shifter is $n - 1$. As a result, there are $(n - 1)$ ancilla inputs. Table 5.7 shows the numbers of ancilla inputs at different (n, q) values.

Table 5.7: Ancilla inputs in arithmetic right shifters

	$(4, 2)$	$(8, 3)$	$(16, 4)$	$(32, 5)$	$(64, 6)$	. . .	(n, q)
SCRL design	3	7	15	31	63	. . .	n-1

5.3.4 Ancilla Inputs in SCRL Reversible Universal Right Shifter

The quantity of Feynman gates applied in the SCRL design of an (n, q) reversible universal right shifter is $n - 1$. And the first Fredkin gate in the selection unit is used to select the data from 0 and a_{n-1}, where 0 is a constant input. Hence, there are n ancilla inputs in SCRL-based reversible universal right shifter in total. We have compared the SCRL-based design of reversible universal right shifter with the existing design in [19]. Table 5.8 shows the ancilla inputs comparison between the SCRL and Kotiyal design [19].

Table 5.8: Ancilla inputs in universal right shifters

	$(4,2)$	$(8,3)$	$(16,4)$	$(32,5)$	$(64,6)$	. . .	(n,q)
SCRL design	4	8	16	32	64	. . .	n
Kotiyal design	12	32	80	192	448	. . .	$n \times q + 2^q$
Improvement ratio K%	66.67	75.00	80.00	83.33	85.71	. . .	N/A
Improvement ratio K% is the improvement ratio compared to Kotiyal design [19].							

Figure 5.6 has a direct view of the increasing trend of ancilla inputs with the increasing n, q values.

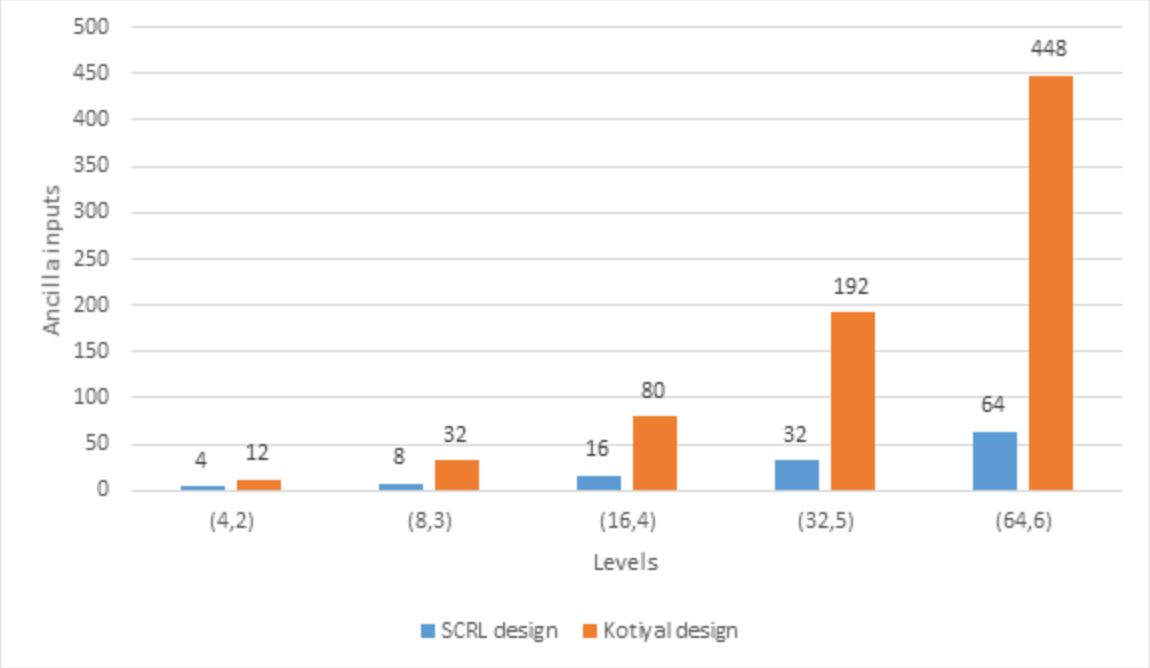


Figure 5.6: Ancilla inputs in reversible universal right shifters

5.3.5 Ancilla Inputs in SCRL Reversible Universal Bidirectional Shifter

Since the additional Fredkin gate in the reverse unit is used for selecting data from the sign bit and the previous last bit of outputs, it does not require an ancilla input. As the result, the total number of ancilla inputs in SCRL-based reversible universal bidirectional shifter are the same as the number of ancilla inputs in universal right shifter. We have compared the SCRL-based design of reversible universal bidirectional shifter with the existing designs in [17, 19, 26]. Table 5.9 shows the number of ancilla inputs in the SCRL design and some other existing designs of reversible universal bidirectional shifters [17, 19, 26](M is the Mitra design [26], K is the Kotiyal design [19] and H is the Hosseininia design [17]).

Table 5.9: Ancilla inputs in universal bidirectional shifters

	$(4,2)$	$(8,3)$	$(16,4)$	$(32,5)$	$(64,6)$
SCRL design	4	8	16	32	64
Kotiyal design	13	33	81	193	449
Mitra design	6	11	20	37	70
Hosseininia design	28	68	156	348	764
Improvement ratio K%	69.23	75.76	80.24	83.42	85.75
Improvement ratio M%	33.33	27.27	20.00	13.51	8.57
Improvement ratio H%	85.71	88.24	89.74	90.80	91.62

Figure 5.7 has a direct view of the increasing trend of ancilla inputs with the increasing n, q values.

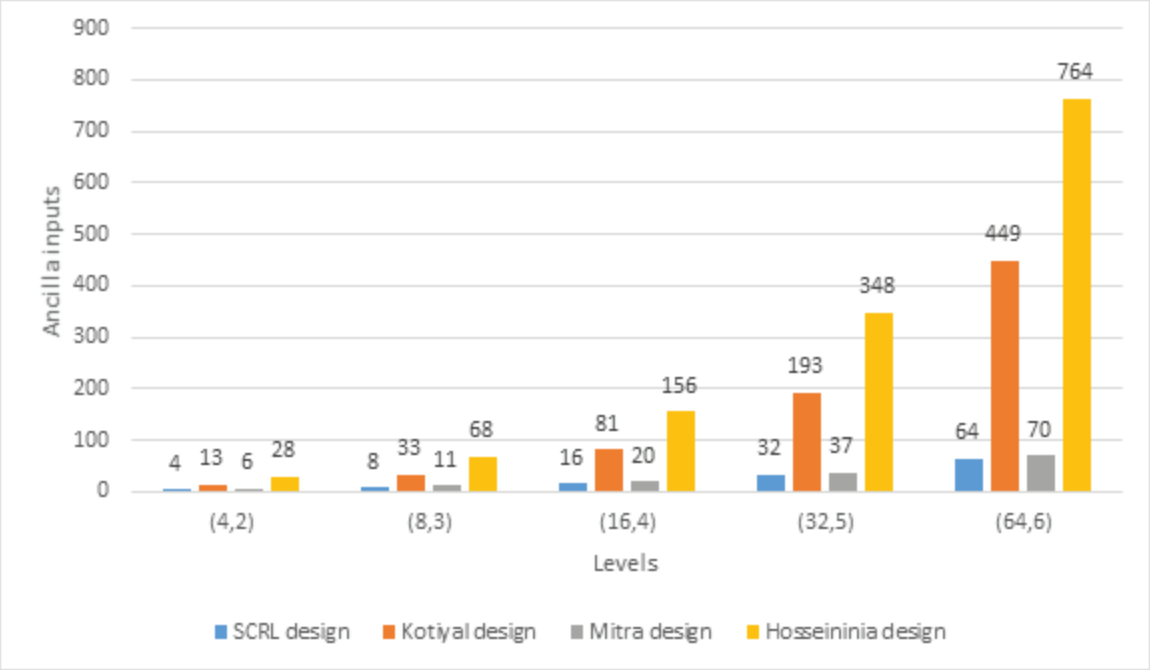


Figure 5.7: Ancilla inputs in reversible bidirectional right shifters

5.4 Comparison of Ancilla Inputs

Table 5.6 shows that the SCRL design reversible logical right shifter has improved quite a lot in ancilla inputs. It saves 62.50% ancilla inputs at $(4,2)$ inputs level and this ratio grows to 83.59% at $(64,6)$ inputs level. Table 5.8 is a comparison table of ancilla inputs for reversible universal right shifters. Improvement ratio K% means the improvement ratio of Kotiyal design . The improvement ratios show that the SCRL design has reduced the number of ancilla inputs in a decent number. And the ratio grows as the inputs level grows. It grows from 66.67% at $(4,2)$ inputs level to 85.71% at $(64,6)$ inputs level for Kotiyal design. So that it is even more efficient when designing extensive inputs universal right

shifters. Table 5.9 is the most important comparison since there are multiple designs in reversible universal bidirectional shifters. The Kotiyal and Hosseininia improvement ratio show that the SCRL design has reduced the number of ancilla inputs. Even for Mitra design, which is the new 2015 design, the SCRL design has also reduced the ancilla inputs decently. Even though the improvement ratio of Mitra design is decreasing, the garbage outputs of SCRL design will always be less than those of Mitra design as it is indicated in the (n, q) inputs level that $n + q$ is larger than n.

5.5 QCA Cost Evaluation

We have also evaluated the cost of implementing the reversible barrel shifter in QCA nanotechnology. The QCA cost consists of two parameters: (i) the amount of inverters used in the design, (ii) the amount of majority voters used in the design. We have compared the QCA cost of the SCRL-based reversible barrel shifters with the existing design of reversible barrel shifters.

5.5.1 Right Rotator Cost

An (n, q) reversible SCRL-based right rotator requires q SCRL gates and each $n \times n$ SCRL gate requires n inverters and $3 \times n$ majority voters. Thus an (n, q) reversible SCRL right rotator requires a total number of $n \times q$ inverters and $3 \times n \times q$ majority voters. Each Fredkin gate uses 2 inverters and 6 majority voters. We have compared the SCRL-based design of reversible right rotator with the design in [33]. Table 5.10 shows the cost evaluation of SCRL right rotator. Figure 5.8 and Figure 5.9 are the comparisons of the inverters and majority voters of the SCRL-based design of reversible right rotaor with the design in [33].

Table 5.10: Cost evaluation of right rotator

	$(4, 2)$	$(8, 3)$	$(16, 4)$	$(32, 5)$	$(64, 6)$	. . .	(n, q)
Inverters	8	24	64	160	384	. . .	$n \times q$
Majority voters	24	72	192	480	1152		$3 \times n \times q$

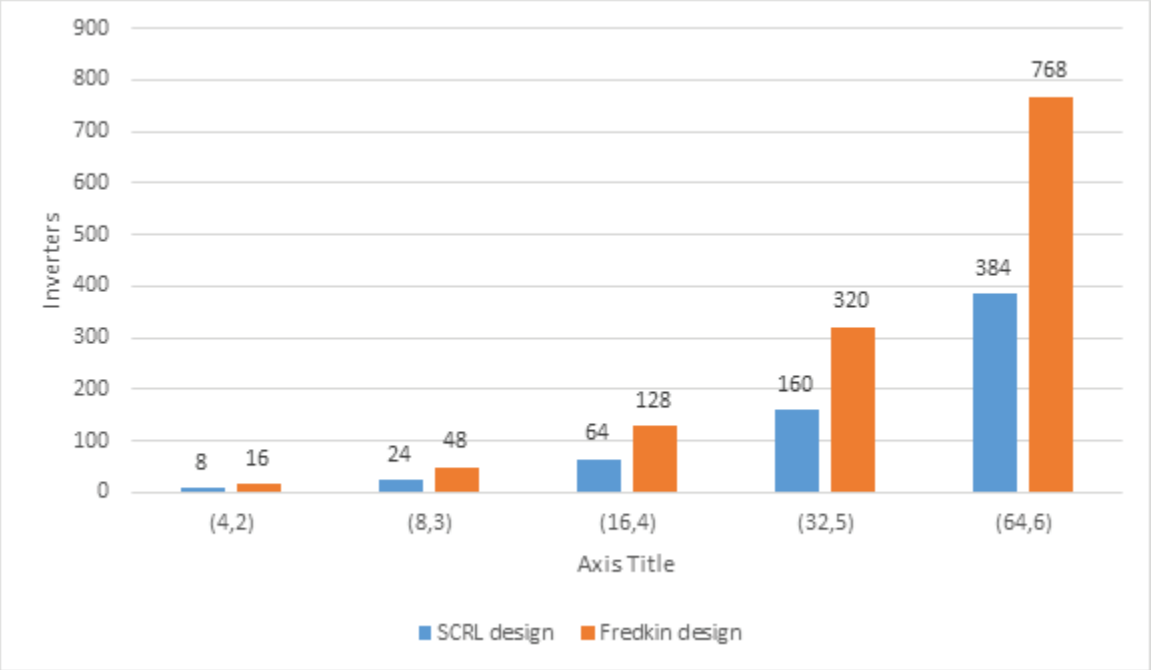


Figure 5.8: Inverter cost comparison of right rotator

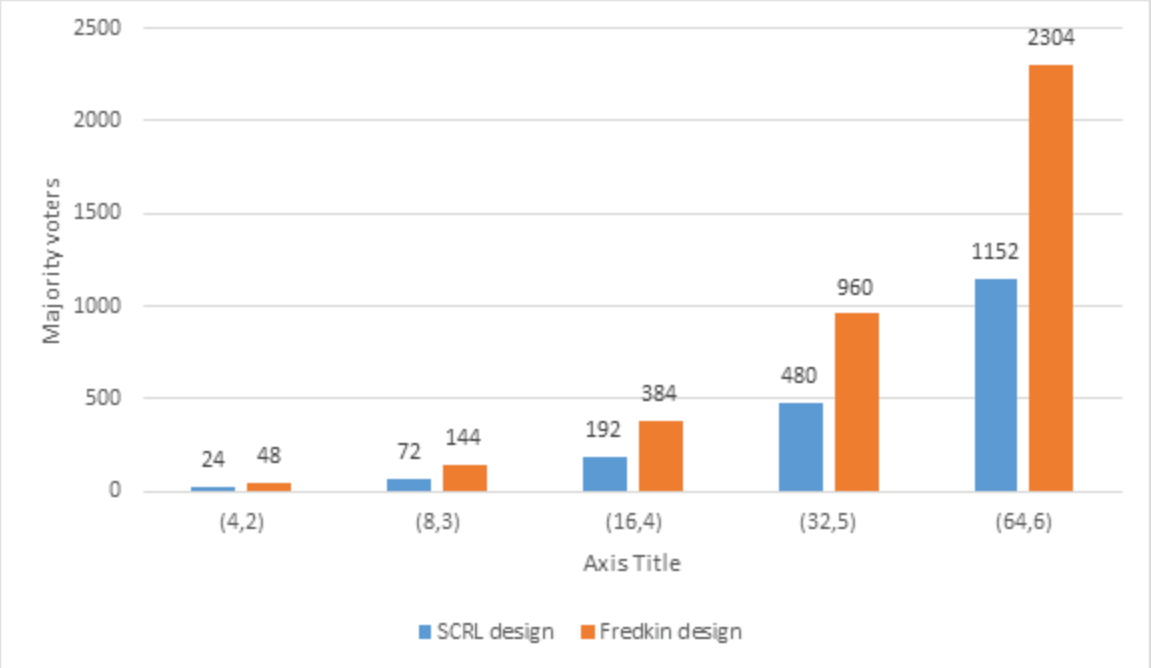


Figure 5.9: Majority voters comparison of right rotator

5.5.2 Logical Right Shifter Cost

An (n, q) reversible SCRL logical right shifter requires q SCRL gates and $n-1$ Fredkin gates. Each $n \times n$ SCRL gate requires n inverters and $3 \times n$ majority voters. Each Fredkin gate is using 2 inverters and 6 majority voters. Thus an (n, q) reversible SCRL right rotator requires a total number of $n \times q + 2 \times (n-1)$ inverters and $3 \times n \times q + 6 \times (n-1)$ majority voters. Each Feynman gate requires 2 inverters and 3 majority voters. We have compared the SCRL-based design of reversible logical right shifter with the designs in [19, 33]. Table 5.11 shows the cost evaluation of SCRL logical right shifter. Figure 5.10 and Figure 5.11 are the comparisons of the inverters and majority voters [19, 33].

Table 5.11: Cost evaluation of logical right shifter

	$(4,2)$	$(8,3)$	$(16,4)$	$(32,5)$	$(64,6)$	(n,q)
Inverters	14	38	94	222	510	$n \times q + 2 \times (n-1)$
Majority voters	42	114	282	666	1530	$3 \times n \times q + 6 \times (n-1)$

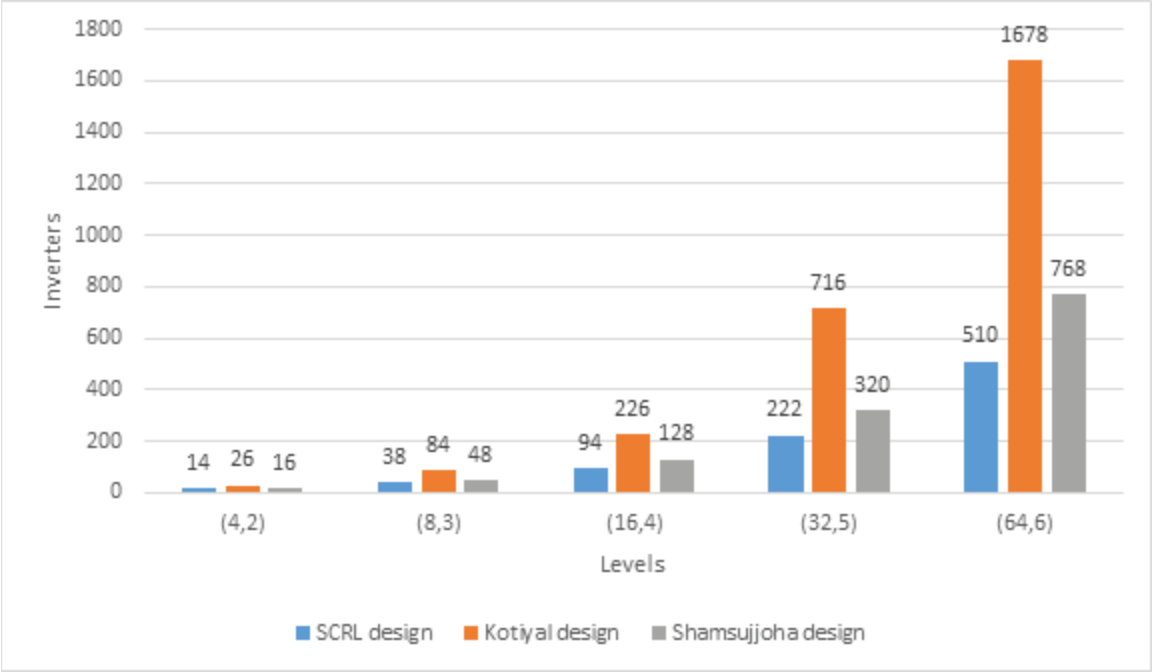


Figure 5.10: Inverter cost comparison of logical right shifter

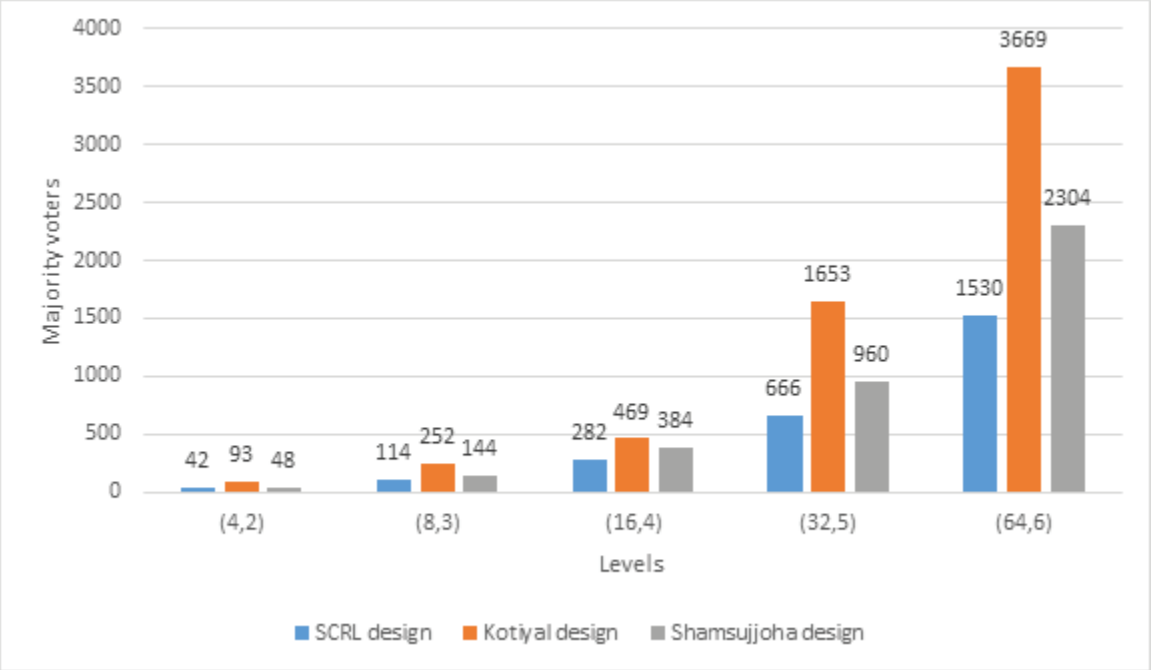


Figure 5.11: Majority voter cost comparison of logical right shifter

5.5.3 Arithmetic Right Shifter Cost

An (n, q) reversible SCRL arithmetic right shifter requires q SCRL gates, $n-1$ Fredkin gates and $n-1$ Feynman gates. The existing designs do not include the arithmetic right shifter, so there is no comparison in this part. Each Feynman gate has 2 inverters and 3 majority voters. Each Fredkin gate has 2 inverters and 6 majority voters. Thus there are $4 \times (n-1) + n \times q$ inverters and $9 \times (n-1) + 3 \times n \times q$ majority voters in an (n, q) SCRL arithmetic right shifter. Table 5.12 shows the cost for different levels.

Table 5.12: Cost evaluation of arithmetic right shifter

	$(4,2)$	$(8,3)$	$(16,4)$	$(32,5)$	$(64,6)$
Inverters	20	52	124	284	636
Majority voters	51	135	327	759	1719

5.5.4 Universal Right Shifter Cost

An (n, q) reversible SCRL-based reversible universal right shifter requires the same number of SCRL gates and Feynman gates as arithmetic right shifter. And it requires one more Fredkin gate than arithmetic right shifter. Thus the number of inverters that an (n, q) SCRL universal right shifter requires is $4 \times (n-1) + n \times q + 2$. And the number of majority voters is $9 \times (n-1) + 3 \times n \times q + 6$. The existing design of reversible universal right shifter in [19] requires $n \times (q+1) + 2$ Fredkin gates and $(q+1) \times n - 1$ Feynman gates. Table 5.13 shows the cost evaluation of SCRL universal right shifter. Figure 5.12 and Figure 5.13 are the comparisons of the inverters and majority voters between the two designs [19].

Table 5.13: Cost evaluation of universal right shifter

	$(4,2)$	$(8,3)$	$(16,4)$	$(32,5)$	$(64,6)$	. . .	(n,q)
Inverters	22	54	126	286	638	. . .	$4 \times (n-1) + n \times q + 2$
Majority voters	57	141	333	765	1725	. . .	$9 \times (n-1) + 3 \times n \times q + 6$

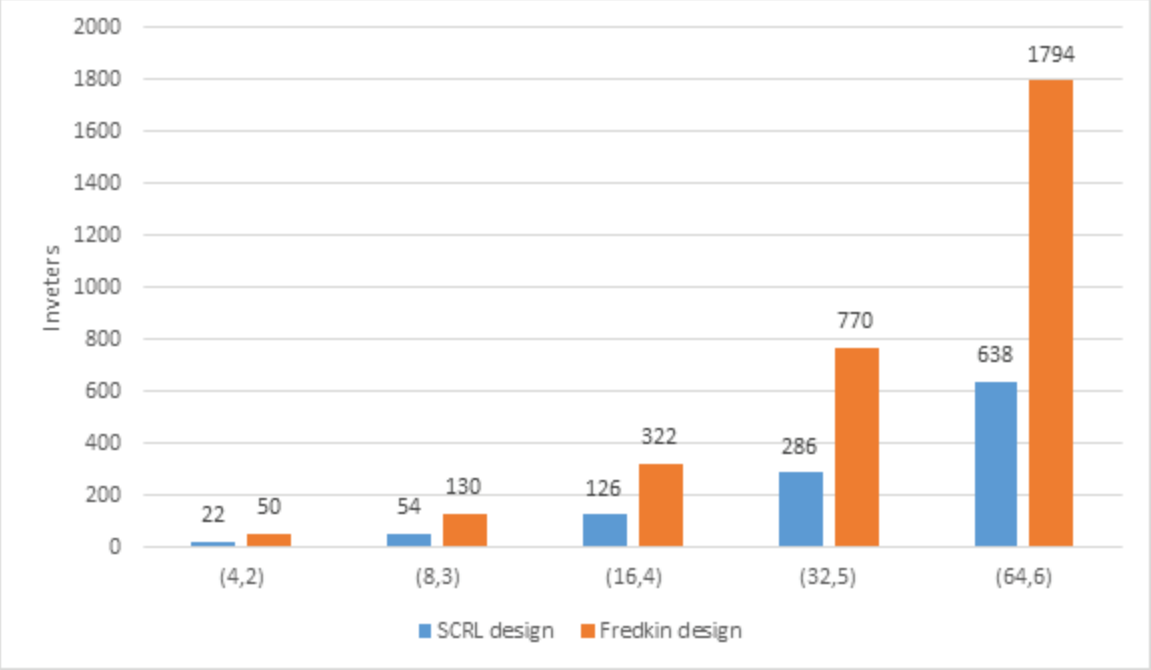


Figure 5.12: Inverter cost comparison of universal right shifter

Table 5.14: Cost evaluation of universal bidirectional shifter

	(4, 2)	(8, 3)	(16, 4)	(32, 5)	(64, 6)
Inverters	28	72	160	352	768
Majority voters	87	195	435	963	2115
M is the Mitra design, K is the Kotiyal design and H is the Hosseininia design					

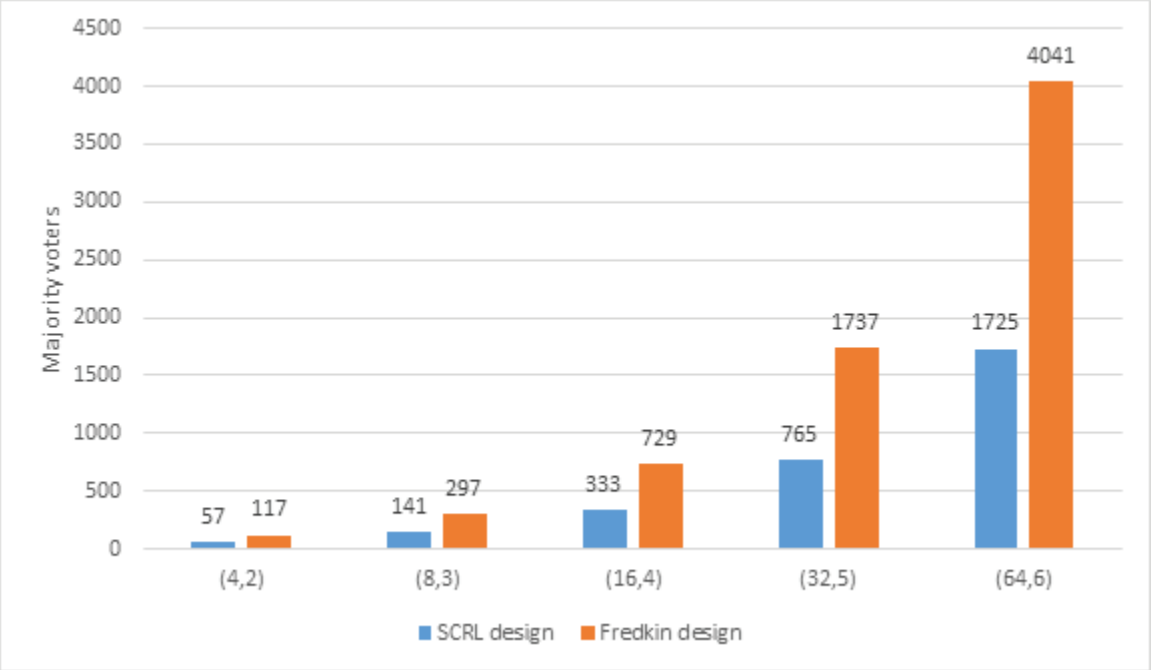


Figure 5.13: Majority voter cost comparison of universal right shifter

5.5.5 Universal Bidirectional Shifter Cost

An (n, q) reversible SCRL-based reversible universal bidirectional shifter requires the same number of Feynman gates as universal right shifter. And it requires 1 more Fredkin gates and 2 more SCRL gate than universal right shifter. Thus the number of inverters that an (n, q) SCRL universal bidirectional shifter requires is $4 \times (n-1) + n \times (q+2) + 4$. And the number of majority voters is $9 \times (n-1) + 3 \times n \times (q+2) + 12$. We have compared the proposed SCRL-based reversible universal bidirectional shifter with the existing designs in [17, 19, 26]. Table 5.14 shows the cost evaluation of SCRL universal bidirectional shifter. Figure 5.14 and Figure 5.15 are the comparisons of the inverters and majority voters between the different designs [17, 19, 26].

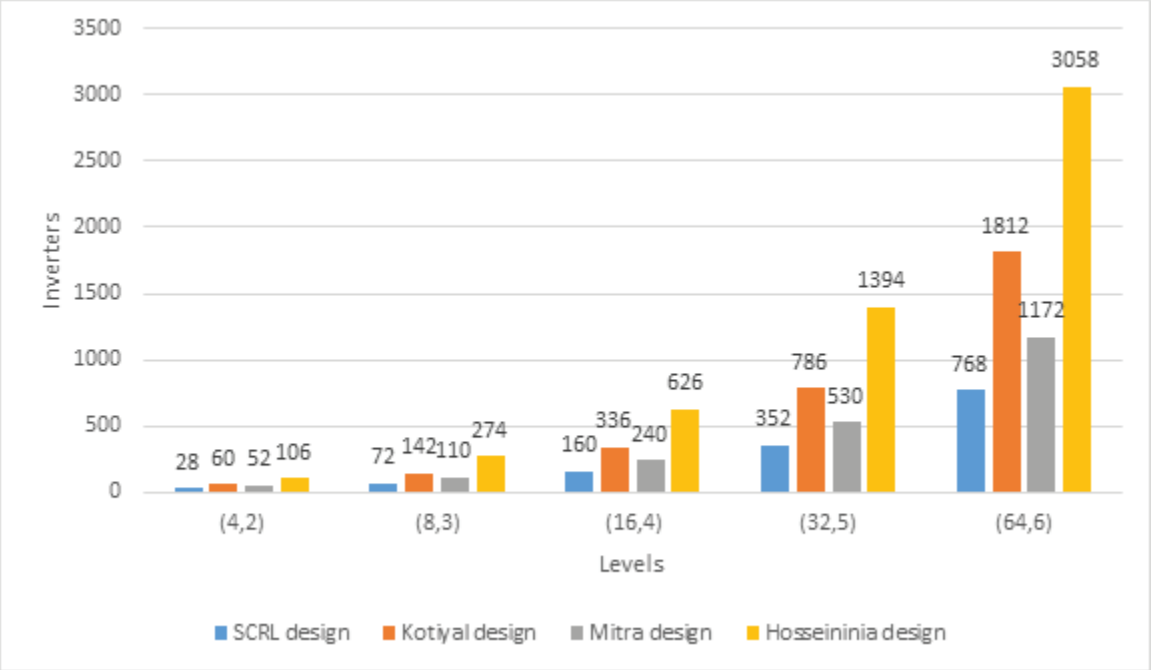


Figure 5.14: Inverter cost comparison of universal bidirectional shifter

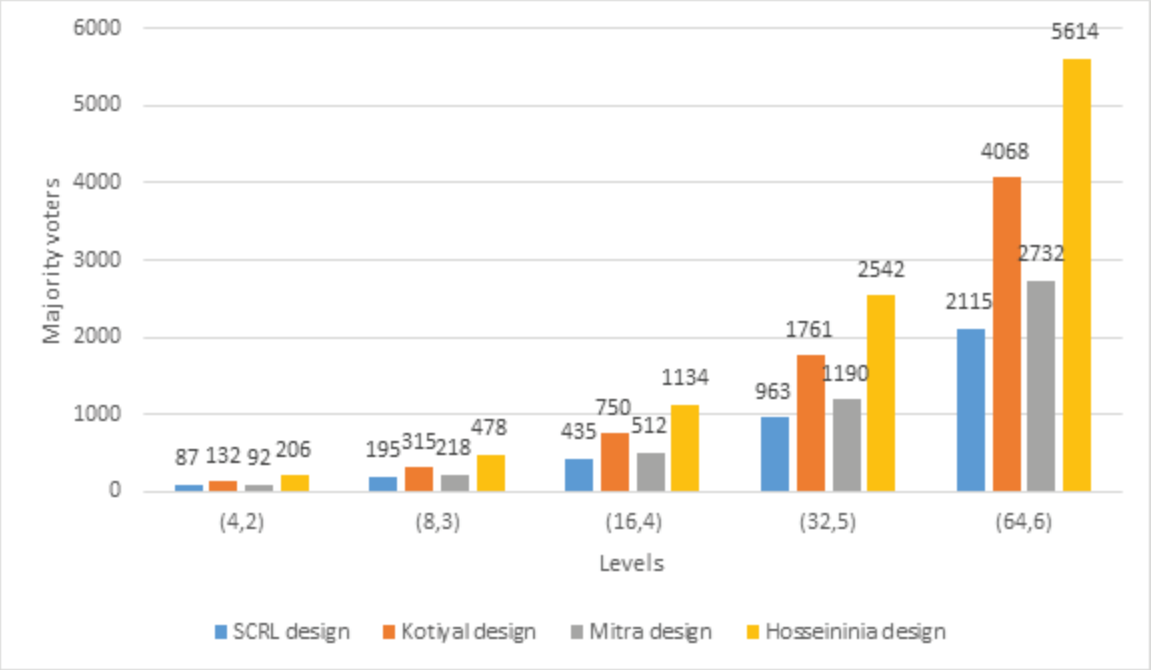


Figure 5.15: Majority voter cost comparison of universal bidirectional shifter

Chapter 6

Conclusion

In this thesis, the SCRL-based design of reversible barrel shifters are presented. The methodologies for the design of (n, q) reversible barrel shifters based on SCRL gate are also proposed. The five proposed methodologies consist of reversible right rotator, reversible logical right shifter, reversible arithmetic right shifter, reversible universal right shifter and reversible universal bidirectional shifter. All the proposed design methodologies of reversible barrel shifters based on SCRL gate have been verified by the Verilog simulation using the Modelsim software. We proved that the SCRL gate is more powerful compared to the existing Fredkin gate in implementing the logic functions. The proposed SCRL gate based designs of reversible barrel shifters are mapped in QCA computing and are compared with the existing designs according to the number of garbage and ancilla bits and the QCA cost. The proposed designs of reversible barrel shifters based on SCRL gate have shown significant improvement compared to the designs existing in literature in the metrics of garbage and ancilla bits and the QCA cost.

References

[1] Mohammad Tanvir Alam, Steven J Kurtz, Mohammad Abu Jafar Siddiq, Michael T Niemier, Gary H Bernstein, Xiaobo Sharon Hu, and Wolfgang Porod. On-chip clocking of nanomagnet logic lines and gates. *Nanotechnology, IEEE Transactions on*, 11(2):273–286, 2012.

[2] MS Alam and MA Karim. Programmable optical perfect shuffle interconnection network using fredkin gates. *Microwave and Optical Technology Letters*, 5(7):330–333, 1992.

[3] Charles H Bennett. Logical reversibility of computation. *IBM journal of Research and Development*, 17(6):525–532, 1973.

[4] Sanjukta Bhanja, Marco Ottavi, Fabrizio Lombardi, and Salvatore Pontarelli. Qca circuits for robust coplanar crossing. *Journal of Electronic Testing*, 23(2-3):193–210, 2007.

[5] Sanjukta Bhanja and Javier Pulecio. A review of magnetic cellular automata systems. In *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, pages 2373–2376. IEEE, 2011.

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

[7] E Oran Brigham and EO Brigham. *The fast Fourier transform and its applications*, volume 1. Prentice Hall Englewood Cliffs, NJ, 1988.

[8] M. Cowlishaw. Decimal arithmetic faq part 3 hardware questions. *http://speleotrove.com/decimal/decifaq3.html*, 2010.

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

[10] James Donald and Niraj K Jha. Reversible logic synthesis with fredkin and peres gates. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 4(1):2, 2008.

[11] Oleg Golubitsky, Sean M Falconer, and Dmitri Maslov. Synthesis of the optimal 4-bit reversible circuits. In *Proceedings of the 47th Design Automation Conference*, pages 653–656. ACM, 2010.

[12] Oleg Golubitsky and Dmitri Maslov. A study of optimal 4-bit reversible toffoli circuits and their synthesis. *Computers, IEEE Transactions on*, 61(9):1341–1353, 2012.

[13] Mariagrazia Graziano, Marco Vacca, and Maurizio Zamboni. *Magnetic QCA design: modeling, simulation and circuits*. INTECH Open Access Publisher, 2011.

[14] Pallav Gupta. Circuit design with quantum cellular automata. In *Nanoelectronic Circuit Design*, pages 441–477. Springer, 2011.

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

[16] Majid Haghparast, Somayyeh Jafarali Jassbi, Keivan Navi, and Omid Hashemipour. Design of a novel reversible multiplier circuit using hng gate in nanotechnology. In *World Appl. Sci. J.* Citeseer, 2008.

[17] Nayereh Hosseininia, Soudabeh Boroumand, and Majid Haghparast. Novel nanometric reversible low power bidirectional universal logarithmic barrel shifter with overflow and zero flags. *Journal of Circuits, Systems, and Computers*, 2015.

[18] William NN Hung, Xiaoyu Song, Guowu Yang, Jin Yang, and Marek Perkowski. Optimal synthesis of multiple output boolean functions using a set of quantum gates by symbolic reachability analysis. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 25(9):1652–1663, 2006.

[19] Saurabh Kotiyal. Design methodologies for reversible logic based barrel shifters. 2012.

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

[21] Weiqiang Liu, S. Srivastava, Liang Lu, M. O'Neill, and E.E. Swartzlander. Are qca cryptographic circuits resistant to power analysis attack? *Nanotechnology, IEEE Transactions on*, 11(6):1239–1251, 2012.

[22] Xiaojun Ma, Jing Huang, Cecilia Metra, and Fabrizio Lombardi. Reversible gates and testability of one dimensional arrays of molecular qca. *Journal of Electronic Testing*, 24(1-3):297–311, 2008.

[23] Dmitri Maslov and Gerhard W Dueck. Reversible cascades with minimal garbage. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 23(11):1497–1509, 2004.

[24] Dmitri Maslov, Gerhard W Dueck, and D Michael Miller. Techniques for the synthesis of reversible toffoli networks. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 12(4):42, 2007.

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

[26] Sajib Kumar Mitra and Ahsan Raja Chowdhury. Optimized logarithmic barrel shifter in reversible logic synthesis. In *VLSI Design (VLSID), 2015 28th International Conference on*, pages 441–446. IEEE, 2015.

[27] Zahra Mohammadi and Majid Mohammadi. Implementing a one-bit reversible full adder using quantum-dot cellular automata. *Quantum Information Processing*, 13(9):2127–2147, 2014.

[28] Michael Nachtigal, Himanshu Thapliyal, and Nagarajan Ranganathan. Design of a reversible floating-point adder architecture. In *Nanotechnology (IEEE-NANO), 2011 11th IEEE Conference on*, pages 451–456. IEEE, 2011.

[29] Matthew R Pillmeier, Michael J Schulte, and Eugene G Walters III. Design alternatives for barrel shifters. In *International Symposium on Optical Science and Technology*, pages 436–447. International Society for Optics and Photonics, 2002.

[30] Javier F Pulecio and Sanjukta Bhanja. Magnetic cellular automata coplanar cross wire systems. *Journal of applied physics*, 107(3):034308–034308, 2010.

[31] Jacqueline E Rice. An introduction to reversible latches. *The Computer Journal*, 51(6):700–709, 2008.

[32] Bibhash Sen, Manojit Dutta, Samik Some, and Biplab K Sikdar. Realizing reversible computing in qca framework resulting in efficient design of testable alu. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 11(3):30, 2014.

[33] Md Shamsujjoha, Hafiz Md Hasan Babu, Lafifa Jamal, and Ahsan Raja Chowdhury. Design of a fault tolerant reversible compact unidirectional barrel shifter. In *VLSI Design and 2013 12th International Conference on Embedded Systems (VLSID), 2013 26th International Conference on*, pages 103–108. IEEE, 2013.

[34] Yasuhiro Takahashi. Quantum arithmetic circuits: A survey. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, 92(5):1276–1283, 2009.

[35] Yasuhiro Takahashi and Noboru Kunihiro. A linear-size quantum circuit for addition with no ancillary qubits. *Quantum Information & Computation*, 5(6):440–448, 2005.

[36] Himanshu Thapliyal, Apeksha Bhatt, and Nagarajan Ranganathan. A new crl gate as super class of fredkin gate to design reversible quantum circuits. In *Circuits and Systems (MWSCAS), 2013 IEEE 56th International Midwest Symposium on*, pages 1067–1070. IEEE, 2013.

[37] Himanshu Thapliyal and Nagarajan Ranganathan. Reversible logic-based concurrently testable latches for molecular qca. *Nanotechnology, IEEE Transactions on*, 9(1):62–69, 2010.

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

[39] Himanshu Thapliyal and Nagarajan Ranganathan. Design of efficient reversible logic-based binary and bcd adder circuits. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 9(3):17, 2013.

[40] Himanshu Thapliyal, Nagarajan Ranganathan, and Ryan Ferreira. Design of a comparator tree based on reversible logic. In *Nanotechnology (IEEE-NANO), 2010 10th IEEE Conference on*, pages 1113–1116. IEEE, 2010.

[41] Himanshu Thapliyal, Nagarajan Ranganathan, and Saurabh Kotiyal. Reversible logic based design and test of field coupled nanocomputing circuits. In *Field-Coupled Nanocomputing*, pages 133–172. Springer, 2014.

[42] Peng Wang, Mohammed Niamat, and Srinivasa Vemuru. Minimal majority gate mapping of 4-variable functions for quantum cellular automata. In *Nanotechnology (IEEE-NANO), 2011 11th IEEE Conference on*, pages 1307–1312. IEEE, 2011.

[43] Robert Wille, Oliver Keszocze, and Rolf Drechsler. Determining the minimal number of lines for large reversible circuits. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*, pages 1–4. IEEE, 2011.

[44] Guowu Yang, Xiaoyu Song, William NN Hung, and Marek A Perkowski. Bi-directional synthesis of 4-bit reversible circuits. *The Computer Journal*, 51(2):207–215, 2008.

[45] R. Zhang, K. Walus, W. Wang, and G.A. Jullien. A method of majority logic reduction for quantum cellular automata. *Nanotechnology, IEEE Transactions on*, 3(4):443–450, 2004.

VITA

Education:

01/2014-05/2015 Master in Electrical Engineering
Dept. Electrical and Computer Engineering, University of Kentucky
Major: Electrical Engineering

08/2012- 12/2013 Bachelor of Science in Electrical Engineering
Junior and Senior Year
Dept. Electrical and Computer Engineering, University of Kentucky
Major: Electrical Engineering

09/2009- 07/2011 Bachelor of Science in Electrical Engineering
China University of Mining & Technology (Xuzhou)
Freshman and Sophomore Year
Dept. School of Information & Electrical Engineering
Major: Electrical Engineering and Automation

HONORS AND AWARDS:

2013 Be admitted as a member of national honor society ETA KAPPA NU (HKN)
2013 Fall 2012-Spring 2013 UK International Engineering Transfer Student Scholarship
2013 Be admitted as a member of the largest technical professional society IEEE
2010 Fall 2009-Spring 2010 CUMT Outstanding in Learning Progress Scholarship