Design and Implementation of A Reversible Central Processing Unit

Lafifa Jamal

Registration No: 197

Session: 2014–2015

A Thesis submitted for the degree of
Doctor of Philosophy in Computer Science and Engineering

Department of Computer Science and Engineering
University of Dhaka, Dhaka-1000, Bangladesh

April 2017

Declaration of Authorship

We declare that this thesis titled, 'Design and Implementation of a Reversible Central Processing Unit' and the works presented in it are my own. We confirm that:

- The full part of the work is done during PhD research study in University of Dhaka, Bangladesh.
- Any part of this thesis has not previously been submitted for a degree or any other qualification at this University or any other institution.
- We have consulted the published work of others with appropriate references.
- This thesis work is done by us and our contributions and enhancements from other works are clearly stated.

Signed: _____

(Lafifa Jamal)
Candidate

Countersigned: _____

(Professor Dr. Hafiz Md. Hasan Babu)
Supervisor

Abstract

Reversible logic is an emerging technology. Conventional logic dissipates more power by losing bits of information whereas reversibility recovers bit loss from the unique input-output mapping. Reversible computing spans computational models that are both forward and backward deterministic. Reversible logic circuits require some constant ancilla inputs to generate the required functions. It produces some unused outputs, called garbage outputs, to maintain the reversibility of a circuit. It is important to minimize the number of ancilla inputs, number of garbage outputs and quantum cost in the design of reversible circuits. Reversible logic has promising applications in emerging computing paradigm such as quantum computing, DNA computing, optical computing, quantum dot cellular automata etc.

The central processing unit (CPU) is regarded as the brain of a computer. It is a piece of hardware that carries out the instructions needed to run a computer program. Processor speed determines the performance of a computer. It performs the basic arithmetical, logical, and input/output operations of a computer system. Advanced conventional processors have inherently higher power dissipation. The arithmetic logic unit of CPU performs simple arithmetic and logical operations and the control unit of CPU manages the various components of the computer.

Power dissipation is the main constrain when it comes to portability. Reversible logic gates are designed as a method to reduce the energy dissipation of logic circuits based on Launder′s concept. Central processing unit designed using reversible logic have a significant contribution in low power computing.

Launder proved that a reversible circuit produces less power than the irreversible circuit. Moreover, by using power optimization algorithm, the power consumption of a reversible circuit can be optimized than their existing reversible counterparts. For example, for a 64-bit comparator, the proposed circuit achieves the improvement of 55.67% in terms of power over the existing reversible one.

Reversible circuit has some disadvantageous properties. It requires a large number of constant ancilla inputs. It also produces a large number of garbage outputs to maintain the reversibility which eventually increases the number of gates, area, power and delay. By using efficient design techniques and suitable algorithms

the number of garbage outputs should be minimized. For example, for a 64-bit comparator, the proposed circuit achieves the improvement of 33.1% in terms of garbage outputs over the existing reversible one.

As reversible computing is a new era in logic synthesis, the existing reversible central processing units have the lack of completeness. However, researchers proposed various basic components of reversible central processing unit. In this thesis, the basic elements of the reversible circuits are newly proposed or improved in terms of numbers of gates, garbage outputs, quantum cost and delay compared to the existing counterparts. The proposed reversible basic elements are further used to construct the complex circuits, such as memory circuits, arithmetic logic unit, control unit etc., in order to design a complete reversible central processing unit.

The main contribution of this thesis is to design a new reversible central processing unit which is efficient in terms of numbers of gates, garbage outputs, quantum cost, ancilla inputs and delay. The reversible central processing unit is designed using novel modularization approach by presenting architecture of a logically reversible processor based on the Von Neumann architecture that can operate with very low power consumption, protection of power analysis attack and long span of life due to less heat dissipation. The organization and architecture of the proposed processor is designed from scratch. Algorithms are proposed to produce the components of the reversible processor and to calculate area and power consumption. The capabilities of the new processor is determined, the datapath layout is designed and the necessary logic is constructed to control the datapath. The computational complexity is considered to estimate the execution time of the algorithm. Existing component designs are compared with the proposed components and theorems are presented to prove the superiority of the proposed architecture. The comparative results show that the proposed circuit requires less power than the existing circuits in terms of numbers of gates, garbage outputs, quantum cost, area and power. For example, for a 64-bit comparator, the proposed design achieves the improvement of 66.6% in terms of number of gates, 33.1% in terms of garbage outputs, 27.6% in terms of quantum cost, 52.39% in terms of area and 55.67% in terms of power over the existing reversible counterpart. The proposed components are simulated and the simulation results verify the correctness of the proposed design. The proposed reversible central processing unit can make a significant contribution in the field of low power reversible computing and quantum computing.

This thesis is dedicated to the memory of my father

Acknowledgements

Undertaking this PhD has been a truly life-changing experience for me and it would not have been possible to do without the support and guidance that I received from many people.

First and foremost I would like to thank my supervisor Professor Dr. Hafiz Md. Hasan Babu. It has been an honor to be his first PhD student. I would like to thank him for encouraging my research and for allowing me to grow as a research scientist. His advice on both research as well as on my career have been priceless. I truly appreciate all his contributions of time and ideas to make my PhD experience productive and stimulating. The joy and enthusiasm he has for his research was contagious and motivational for me, even during tough times in the PhD pursuit. I am also thankful for the excellent example he has provided as a successful researcher and professor.

I gratefully acknowledge the funding received towards my PhD from Prime Minister's Office. I received the PhD Fellowship of Prime Minister's Research and Higher Education Assistance Fund.

I would like to express my sincere gratitude to all the faculty members of Department of Computer Science and Engineering, University of Dhaka.

I owe a lot to family members, who encouraged and helped me at every stage of my personal and academic life, and longed to see this achievement come true. I love them so much, and I would not have made it this far without them. I specially acknowledge my mother for all her love and support. I dedicate this thesis to the memory of my father who would have been happy to see me to complete the degree.

Above all, I owe it all to Almighty Allah for granting me the wisdom, health and strength to undertake this research task and enabling me to its completion.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Since the beginning of time, technology has helped us out as a human race. From the invention of the wheel to the Internet, technology has greatly affected the way the civilization has grown. Our personal life is highly dependent on the technology that people have developed. Technology has advanced with years and it has influenced the way to live, communicate, travel. These continuous technological advancements have brought about many changes. As peoples demands and life style change, the demand for the type of technology to progress is very high. In the 20th and 21st centuries, the advancement in technology has been exceptionally fast. Over the last few decades, improvements in computer technologies have reached an impressive level. The central processing unit (CPU) has perhaps made more significant contributions to our daily lives than any other 20th-century invention. Now-a-days, the processor is not required only in desktop computers. Powerful CPUs are also required in portable and smart devices.

The central processing unit (CPU) is regarded as the brain of a computer [13]. It is a piece of hardware that carries out the instructions needed to run a computer program. It performs the basic arithmetical, logical, and input/output operations of a computer system. A CPU usually has a number of components. The first is the arithmetic logic unit (ALU), which performs simple arithmetic and logical operations. Second is the control unit (CU), which manages the various components of the computer. It reads and interprets instructions from memory and transforms

them into a series of signals which activate other parts of the computer. The control unit handles all processor control signals. It directs all input and output flow, fetches code for instructions from microprograms and directs other units and models by providing control and timing signals. Computer performance is primarily determined by the performance of the CPU.

Energy consumption is an important aspect of most computing systems today and this is especially true for embedded systems and battery-dependent computers. Reversible computing has the potential to reduce power consumption and heat dissipation [14], [15]. Reversible Central Processing Unit (RCPU) can play a significant role in designing portable and smart devices which requires extended battery life.

1.1 Problem Identification and Motivation towards Reversible Computing

Since the days of the mainframe and the desktop pc, processor technology has changed continually to address users' needs. During this era VLSI design efforts have focused primarily on optimizing speed to realize computationally intensive real-time functions. As a result, semiconductor ICs were invented to successfully integrated various complex modules to meet the computation demands. While these solutions have addressed the real-time problem, they have not addressed the increasing demand for portable operation. Now, with mobile computing becoming a dominant market, processor innovation has become a critical driving force for change. Power dissipation is the main constrain when it comes to portability. The portable device consumer demands more features and extended battery life at a lower cost. This requires high levels of integration in advanced processors, but advanced processors have inherently higher power dissipation. So, there is a need to reduce power dissipation to reduce power consumption.

Generally, the more expensive is the processor, the more it runs faster. This requires advanced processors which dissipates higher power. The basic elements of

quantum computers are the quantum gates. Reversible logic gates can be implemented using quantum gates. Preservation of information is a key motivation for research in reversible computing. Reversible logic gates are designed as a method to reduce the energy dissipation of logic circuits based on Launder's concept [15]. For this reason, the quantum computers dissipate very low power which eventually results low power consumption.

Nanotechnologies are the design, characterization, production and application of structures, devices and systems by controlling shape and size at nanometre scale [16]. Smaller, faster and lighter computers will be the potential result of the nanotechnology impact on computer technology. Conventional computers do not support nanotechnologies. So there is a need to introduce nanotechnologies in computer technologies. Nanotechnology may be used to develop many new kinds of batteries that are quicker-charging, more efficient, lighter weight, have a higher power density, and hold electrical charge longer. Smaller, faster and lighter nanotechnology based computers will have a significant impact on computer technology.

More powerful and smaller computers will encrypt data and provide round-the-clock security. Hardware cryptography utilizes the unique properties of quantum mechanics which provides unbreakable security for businesses, government, and military. Security of information is a vital issue. Software is not enough to protect a system. It also needs physical protection [17]. Despite the computation power, general purpose processors are not able to compete with dedicated crypto-processors for certain applications where information security must be entertained. Moreover, high-performance general purpose processors have enormous power consumption. They produce a tremendous amount of heat. Power dissipation and heat generation, make these processors almost unusable for embedded systems. The target of this thesis is to propose a new CPU with emerging technologies such as nanotechnologies, quantum technologies etc.

In this work, the design methodologies for the realization of reversible logic based central processing unit has been proposed. Reversible central processing unit is the foundation of a reversible computer. Implementation of reversible central processing unit can be a solution to reduce power consumption and to increase

the performance of the whole system which leads to the main motivation of this thesis work. Reversible processor design is indeed a challenging task and thinking the organization and architecture of the design in reversible way requires a plenty of research works. Besides, previous works regarding reversible central processing unit lack efficiency and completeness. Hence, there is a scope of research to develop new design and methodology for realization of reversible central processing unit in which cost parameters are optimized.

Quantum computer is considered one of the most promising computing paradigm in which computation can be performed at an atomic level which makes it feasible to work beyond the existing limits of the semiconductor industry. A quantum computer is composed of quantum logic gates; each gate performing an elementary unitary operation on one, two or more two-state quantum systems called qubits. Each qubit represents an elementary unit of information corresponding to the classical bit values 0 and 1. Any unitary operation is reversible and hence quantum networks must be built from reversible logic components [18]. Reversible computers can easily be transformed into quantum computers. Thus, the feasibility of reversible logic circuits could critically impact in the realization of quantum computing. So, the efficient design of reversible computers leads us to construct the computers with the emerging technologies such as quantum computers, DNA computers etc. Moreover, energy dissipation would not occur if a computation is carried out in a reversible way [19]. The basic elements of reversible gates are quantum gates. So, the quantum computers, which are made of quantum gates, can play a significant role in the paradigm of low power computer devices. In contrast, there are emerging nanotechnologies such as Quantum Dot Cellular Automata (QCA) Computing, Optical Computing, and DNA Computing etc. [20], [21], [22], [23], [24] where the energy dissipated due to information destruction will be a significant factor of the overall heat dissipation of the system. Thus, one of the primary motivation for adopting reversible logic lies in the fact that it can provide a logic design methodology for designing ultra-low power circuits beyond KTln2 limit for those emerging nanotechnologies in which the energy dissipated due to information destruction will be a significant factor of the overall heat dissipation.

The performance of any computing system is given by the number of useful operations per unit time. The improving system performance requires increasing the

average energy efficiency of useful operations which can be done by minimizing the energy dissipated during a job. Thus, in emerging nanotechnologies in which the signal energy is few orders of magnitude higher than the $KT\ln 2$ limit (where K is the Boltzmann constant and T is the operating temperature), further improvements can only be gained by going beyond the $KT\ln 2$ which is only possible by adopting the reversible logic. This shows that reversible logic can be beneficial towards raising computer performance in emerging nanotechnologies. Reversible logic could also help to potentially recover and retain a fraction of the signal energy that can be reused for subsequent operations by doing the computation using the forward path and then undoing the computation using the backward path. These concepts have been implemented in CMOS to save significant amount of energy dissipation even close to 90% using the concepts such as reversible energy recovery logic [25].

Reversible logic have also promising applications in online and offline testing of faults. For example, it has been proved by researchers that for reversible logic circuits, the test set that detects all single stuck-at faults can also detect multiple stuck-at faults [26].

In summary, the motivations for this dissertation are: (i) to explore the design and synthesis of reversible central processing unit considering metrics of numbers of reversible gates, garbage outputs, quantum cost, delay and ancilla inputs, and (ii) to explore the benefits of reversible logic in circuits based on emerging nanotechnologies.

To propose the new CPU, the existing processor technologies need to be studied. Section 1.2 discusses the brief history and architectures of different microprocessors.

1.2 Brief History of Microprocessor

The first microprocessor, Intel 4004, was introduced in 1971 [27]. This processor could work with 4-bit at a time and run at a clock speed of 108KHz. It had

2,300 transistors and was built on a 10-micron process. The maximum addressable memory was 640 bytes. It was originally designed for use in a calculator. However, it proved to be useful for many other functions because of its inherent programmability.

8008 processor was introduced in 1972 [28]. It could run at a clock speed of 200KHz and it had 3,500 transistors. The big change in the 8008 was that it could move data 8-bit at a time, twice as much as the previous chip. It could also address 16KB memory. This chip was mainly used in dumb terminals and general-purpose calculators.

In 1974, 8080 processor was introduced [28]. It ran at a clock rate of 2 MHz. Its performance was 10 times better than the performance of the 8008. The 8080 chip, which contained 6,000 transistors, was built on a 6-micron process. As it had an 8-bit data bus, it could transfer 8 bits of data at a time. The 8080 could address up to 64KB of memory. Since this was the first processor chip used in the first personal computer, the Altair 8800, it initiated the PC revolution

Z-80 processor was the first clone processor. It was a vastly improved version of the 8080. The Z-80 also incorporated a superset of 8080 instructions. It could run all 8080 programs. In addition, it also included new instructions and new internal registers. Initially, the Z-80 ran initially at 2.5MHz. It contained 8,500 transistors and could access 64KB of memory.

The 8085 was released by Intel in 1976 [29]. It was popular as an embedded controller, finding use in scales and other computerized equipment. The 8085 ran at 5MHz and contained 6,500 transistors. It was built on a 3-micron process and incorporated an 8-bit data bus.

In 1976, MOS Technologies were introduced in 6502 microprocessor. It was an 8-bit processor. This chip was used in the designs of Apple I and Apple II. The 6502 and its successors were also used in game consoles, including the original Nintendo Entertainment System (NES).

In 1978, Intel introduced the 8086 [28]. The 8086 chip brought with it the original x86 instruction set that is still present in current x86-compatible chips such as the Pentium 4 and AMD Athlon. The 8086 could work on 16-bit numbers and data

internally and also transfer 16 bits at a time in and out of the chip. The 8086, which initially ran at up to 5 MHz, contained 29,000 transistors. It used 20-bit addressing. Although the 8086 was a great chip, its disadvantage was that it was expensive at the time. Moreover, it required expensive 16-bit board designs and infrastructure to support it.

Intel released 8088 processor in 1979. It used the same internal core as the 8086, had the same 16-bit registers, and could address the same 1MB of memory. However, the external data bus was reduced to 8 bits. This enabled support chips from the older 8-bit 8085 to be used, and far less expensive boards and systems could be made. These are the reasons why IBM chose the 8088 instead of the 8086 for the first PC.

Intel introduced the IA-64 (Intel Architecture, 64-bit) in the form of the Itanium and Itanium 2 processors. It was first announced in 1994 as a CPU development project with Intel and HP, and the first technical details were made available in October 1997. This resulted in the IA-64 architecture and Itanium chip, which was officially released in 2001.

AMD developed 64-bit extensions to IA-32, which it calls AMD64 (originally known as x86-64). Intel eventually released its own set of 64-bit extensions, which it calls EM64T or IA-32e mode.

The latest development is the introduction of dual-core processors from both Intel and AMD. Dual-core processors have two full CPU cores operating off of one CPU package. It allows a single processor to perform the work of two processors like multiple single-core processors, dual-core processors, split up the workload caused by running multiple applications at the same time. Current dual-core processors also support AMD64 or EM64T 64-bit extensions, which allows the user to enjoy both dual-core and 64-bit computing's advantages.

Modern desktop computers support systems with multiple CPUs. Both Intel and AMD currently offer fast quad, hex and octa-core desktop CPUs, making multi-CPU systems obsolete for many purposes. The desktop market has been in a transition towards quad-core CPUs since Intel's Core 2 Quad was released and are now common, although dual-core CPUs are still more prevalent.

1.2.1 Architecture of Intel 4004 Microprocessor

The basic function of a CPU is to fetch, decode and execute instructions from memory. To accomplish this it fetches data from an external memory source and transfer it into the register. It can distinguish between instructions and operands. The CPU also performs additional tasks such as responding to external events such as resets and interrupts, provide memory management facilities to the operating system, etc. In brief, the CPU can do following operations: 1. Provide temporary storage for addresses and data 2. Perform arithmetic and logic operations 3. Control and schedule all operations

Figure 1.1 shows the basic architecture of Intel 4004 Microprocessor.

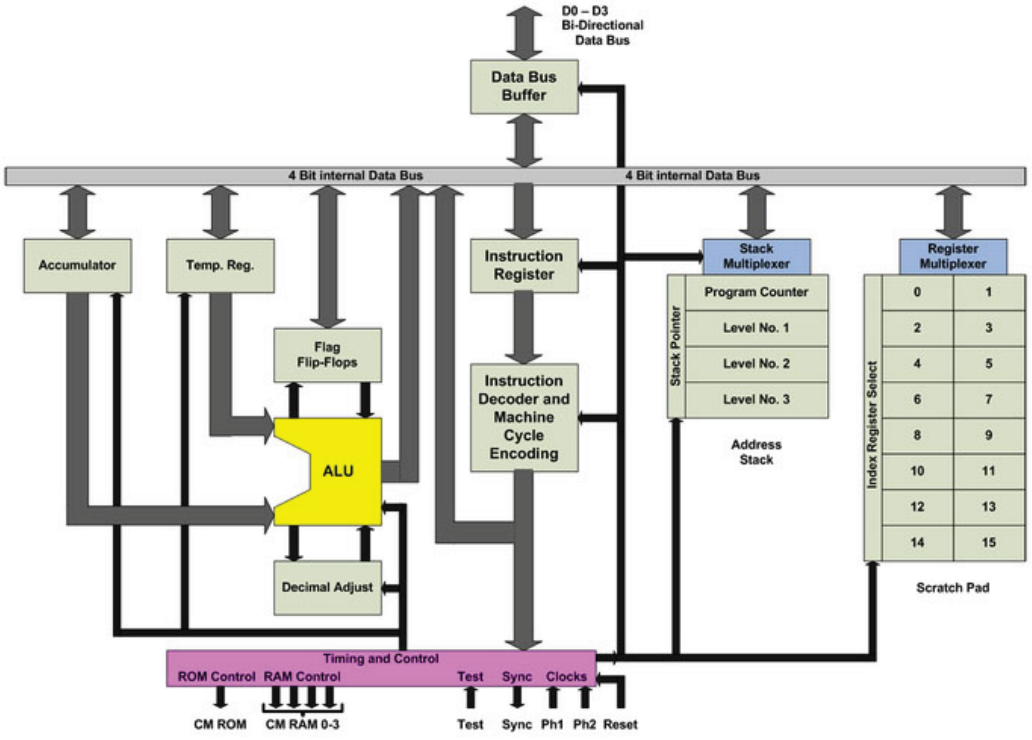

FIGURE 1.1: Intel 4004 Microprocessor Architecture [1]

The Intel 4004 microprocessor is consists of some basic elements which are instruction register, stack pointer, instruction decoder, program counter, accumulator register, flag register, arithmetic logic unit and control unit [1], [28].

Registers: Registers are used for a variety of purposes such as holding the address of instructions and data, storing the result of an operation, signaling the result of a logic operation, or indicating the status of the program or the CPU itself. They consist of several integrated transistors which are configured as a flip-flop circuits each of which can be switched into a 1 or 0 state. In order to provide backward compatibility, registers may be sub-divided. For example, the Pentium processor is a 32-bit CPU, and its registers are 32 bits wide. Some of these are sub-divided and named as 8-bit and 16-bit registers in order to run 8-bit and 16-bit applications designed for earlier x86 microprocessors.

Instruction Register: Instruction Register is a register which holds the instruction that is currently on execution. When the Bus Interface Unit receives an instruction it transfers it to the Instruction Register for temporary storage.

Stack Pointer: Stack registers are used when system calls are made by a process to operating system routines; or when hardware interrupts generated by input/output (I/O) transactions on peripheral devices; or when a process initiates an I/O transfer; or when a process rescheduling event occurs on foot of a hardware timer interrupt. The stack pointer is the register which holds the address of the most recent stack entry.

Instruction Decoder: Instruction Decoder decodes the instruction so that related microcode modules can be transferred from the CPU's microcode ROM to the execution unit. The Instruction Decoder also initiates the store of referenced operands in appropriate registers so data at the memory locations referenced can be fetched.

Program Counter: The Program Counter (PC) is the register that stores the address in primary memory of the next instruction to be executed. When the referenced instruction is fetched, the address in the PC is incremented to the address of the next instruction to be executed.

Accumulator Register: The accumulator may contain data to be used in a mathematical or logical operation, or it may contain the result of an operation. General purpose registers are used to support the accumulator by holding data to be loaded to/from the accumulator.

Flag Register: Each bit in the Flag register has a pre-assigned meaning and the contents are monitored by the control unit to help control CPU related actions.

Arithmetic and Logic Unit: The Arithmetic and Logic Unit (ALU) performs all arithmetic and logic operations in a microprocessor, e.g., addition, subtraction, logical AND, OR, EX-OR, etc. A typical ALU is connected to the accumulator and general purpose registers and other CPU components that help transfer the result of its operations to RAM via the Bus Interface Unit and the system bus.

Control Unit: The control unit coordinates and manages CPU activities, in particular the execution of instructions by the arithmetic and logic unit (ALU). In Pentium processors its role is complex, as microcode from decoded instructions are pipelined for execution by two ALUs.

1.2.2 Architecture of Pentium III Microprocessor

The Pentium III model, which was introduced in 1999, represents Intel's 32-bit x86 desktop and mobile microprocessors in accordance with the sixth-generation P6 micro-architecture. The Pentium III processor included SDRAM, which enabled incredibly fast data transfer between the memory and the microprocessor. Pentium III was also faster than its predecessor, the Pentium II, featuring clock speeds of up to 1.4 GHz. The Pentium III included 70 new computer instructions which allowed 3-D rendering, imaging, video streaming, speech recognition and audio applications to run faster. It has 512KB L2 Cache. It implements a Dynamic Execution microarchitecture a unique combination of multiple branch prediction, data flow analysis, and speculative execution. This enables these processors to deliver higher performance than the Intel Pentium processor, while maintaining binary compatibility with all previous Intel architecture processors. The processor also executes Streaming SIMD (single-instruction, multiple data) Extensions for enhanced floating point and 3-D application performance [2].

The processor utilizes multiple low-power states such as Sleep, and Deep Sleep to conserve power during idle times. The processor includes an integrated on-die 512KB 8-way set associative level-two (L2) cache. The L2 cache implements the Advanced Transfer Cache architecture with a 256-bit wide bus. In addition, the processor includes a 16 KB level one (L1) instruction cache and 16 KB L1 data cache. These cache arrays run at the full speed of the processor core. It has a dedicated L2 cache bus, thus maintaining the dual independent bus architecture to deliver high bus bandwidth and performance. Memory is cacheable for 64 GB of addressable memory space, allowing significant headroom for desktop systems. It supports a lower voltage differential and single-ended clocking for the system bus.

Figure 1.2 shows the block diagram of Intel Pentium III microprocessor.

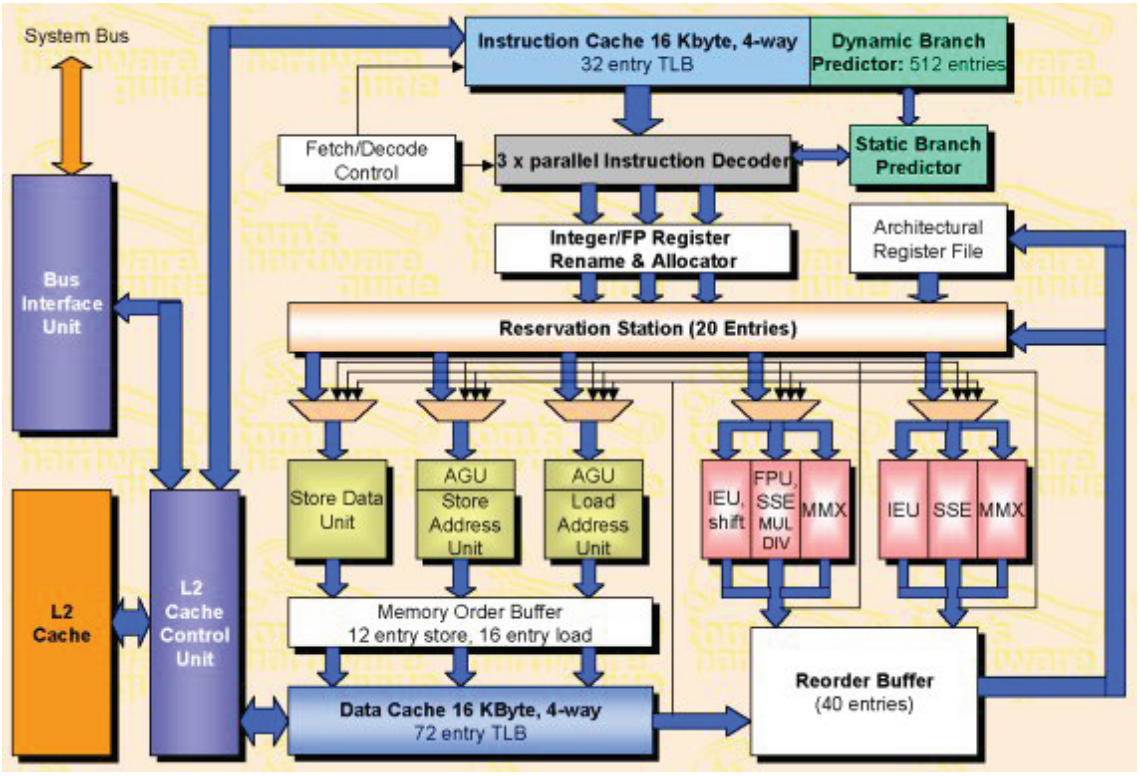

FIGURE 1.2: Block Diagram of Intel Pentium III Microprocessor Architecture [2]

1.2.3 Architecture of Core i3 Microprocessor

Core i3 microprocessor was introduced in 2010. The features of the i3 processor are highly improved as compared to previous version of the processor by Intel. Core i3 processors offer the perfect accuracy and high performance and response rate, as a result of which it provides the users with the high throughput rates, and also reduced time in executing the programs by the processor.

Core i3 processors offer the perfect accuracy and high performance and response rate which in result provide the users with the high throughput rates, and also reduced time in executing the programs by the processor. It is fully equipped by the latest HD graphics with powerful and advanced video engine that provides smooth high quality display along with the 3d graphics capabilities. On the whole i3 processors can be considered as the high graphical and multi media display processors for daily computing. Intel i3 processors also provide hyper threading technology to its users which enable the multitasking capability of both user and the system. The systems with i3 processors can perform execution and compilation of two tasks simultaneously without causing the executing delays and debuggers errors. i3 processors are smarter, faster and more adaptive in all kinds of networking scheme. They can be used with any of the hard disk configurations. Core i3 processors have 3.06 GHz and 2.93 GHz core speed which is very high as compared to the previous configurations of Intel processors. They have four processing threads that enables multithreading and multi tasking. 4 mega byte additional Cache memory is also provided inside the processor.

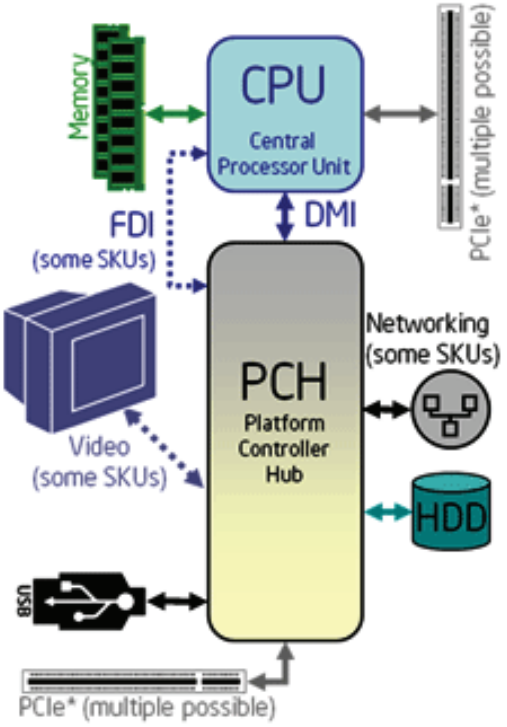

FIGURE 1.3: Block Diagram of Intel Core i3 Microprocessor Architecture [3]

Figure 1.3 shows the block diagram of Intel Core i3 microprocessor. It has several notable features, including:

- Dual core processing has the ability to run two independent programs with one hardware.
- i3 processors have improved Pentium base, they have totally new architecture with more integrations and high speed performance structure.
- Hyper threading technology also enables user to enjoy the high speed and better performance with more reliable outputs. It has 4 tasking threads that allows user to easily execute 3 to 4 programmes at a time.
- Smart memory and cache sequence allows user to enjoy the optimized and efficient data access both direct and sequentially. Effective shortcuts have reduced the access time of the file and system.
- HD graphical features also make these processors distinguished from the others because they are considered as best in their resolution.

1.2.4 Architecture of Core i5 Microprocessor

Core i5 processors are more advanced as compared to the i3 or all the previous versions of the processors. Core i5 processors are introduces to do the intelligent networking and enhance the performance of the working for the sake of different purposes such as for gaming, faster procession, reliable data transmission etc. One of the important feature of the i5 processors is that it automatically manages the power supply where needed and does not break the speed and the performance of the system. They also allow the user to enjoy the heavy applications with the higher rate such as HD video composing, composing a music and many more. Core i5 also provide the opportunity to the users to use the system with multi tasking. They are also able to increase the memory of the system and help users to work with the high bandwidth and great performance. A big feature of the i5 processors is that they have ability to run two multitasking processors together that are generally called as dual processors and can increase the working performance of the system efficiently. Turbo boost technology of i5 processors is the key beneficial feature of the i5 processors that allow the users to do their regular and important working with the help of heavy applications. Core i5 processor also consists of Hyper Threading technology that enables the users for multitasking and improves their business or working by working on the two different tasks at the same time.

The features of Core i5 Microprocessor are listed below:

- Core i5 processors have ability to work with integrated memory and can enhance the performance of the applications. The increase the memory up to 1333 MHz.
- Core i5 processors have high speed performing rate so they are able to perform at the maximum CPU rate of 3.6 GHz.
- Turbo technology is present in the device that boost up the working speed of the computational systems.
- It also provides the 64-bit architecture for the users for the reliable and much more faster working.

- Micro architecture for the i5 processors was presented by the Nehalem and these processors have a cache rate up to 8 MB.
- It provides high quality visualization for advanced applications.
- It introduces high performance with the help of dual processor technology.
- The HD graphics enhance the video graphing and applications related to the same architecture.

1.2.5 Architecture of Core i7 Microprocessor

The revolution of computer system has moved up enormously. From the age of heavy, bulky and large sized computers today the world has moved to thinnest notebooks. From the era of simple 4-bit Intel 4004 processors, we have moved up to Intel Core i7 [3].

Intel's Core i7 microprocessor is the latest and fastest microprocessor. Intel core i7 processor is one of the first core i-series processors built based on the new Intels micro architecture technology called Nehalem. Intels Nehalem micro architecture has key features that outrank from previous processor technologies.

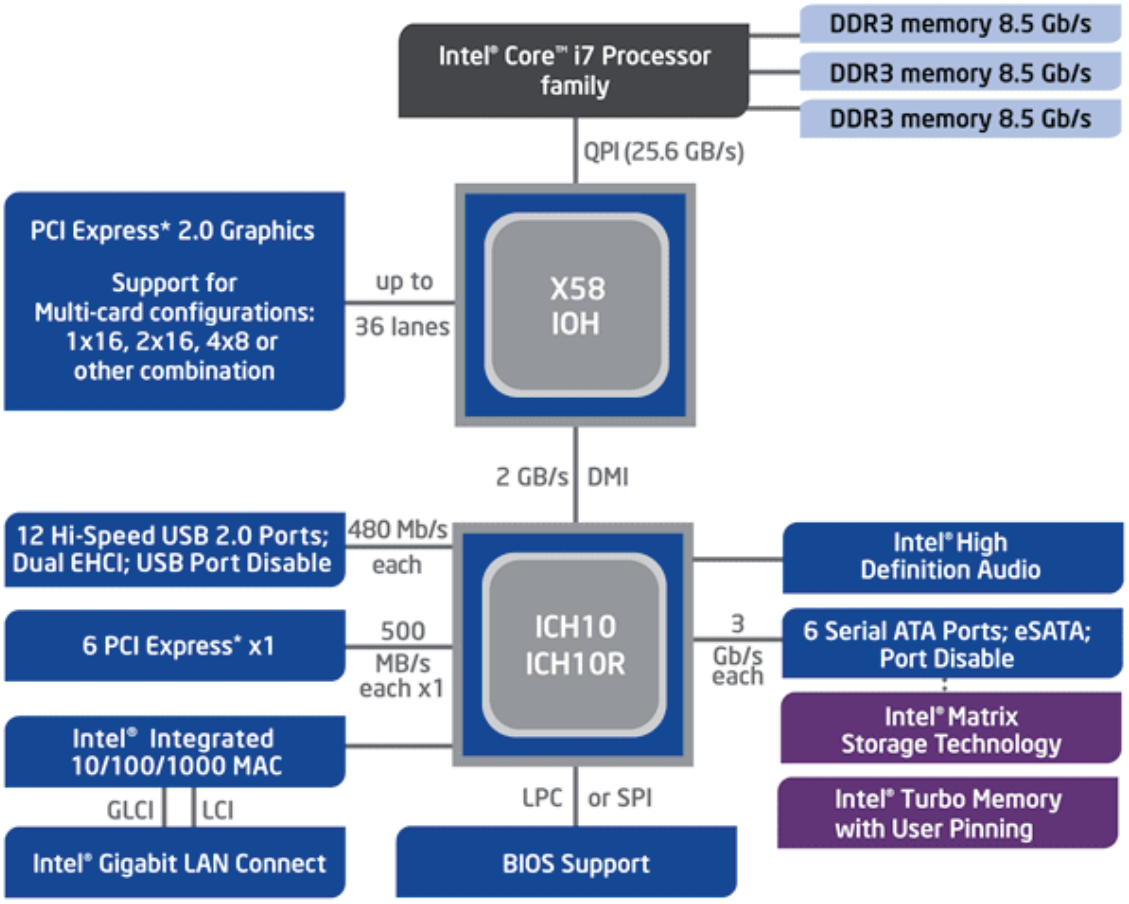

FIGURE 1.4: Block Diagram of Intel Core i7 Microprocessor Architecture [3]

Figure 1.4 shows the block diagram of Intel Core i7 microprocessor. It has several notable features, including:

- Four independent CPU cores
- 64-bit execution support
- Speed ranges from 2.66GHz to 3.33GHz
- Two-way multithreading per CPU core
- Front Side Bus Speed include 2GHz, 4.8GHz or 6.4GHz
- A built-in two-channel DDR3 DRAM controller
- Integrated L1, L2, and L3 caches

- Direct Media Interface (DMI) connection between the processor and the Platform Controller Hub (PCH)
- Enhanced Intel SpeedStep Technology
- Virtualization Technology
- Streaming SIMD Instructions (MMX)
- Over clocking capability
- Supports Intel Turbo Boost technology

The Intel Core i7 processor achieves its high performance through its multiple CPU cores and its Simultaneous Multithreading (SMT) feature. Between the four cores and the two-way multithreading per core, the Intel Core i7 processor appears to software as eight independent 64-bit CPUs [3].

The Core i7 design is based on current Core 2 processors but has been widely revised, from its front end to its memory and I/O interfaces and nearly everywhere in between. The Core i7 integrates four cores into a single chip, brings the memory controller onboard, and introduces a low-latency point-to-point interconnect called Quick Path to replace the front-side bus. Intel has modified the chip to take advantage of this new system infrastructure, tweaking it throughout to accommodate the increased flow of data and instructions through its four cores. The memory subsystem and cache hierarchy have been redesigned, and simultaneous multithreadingbetter known by its marketing name, Hyper-Threadingmakes its return, as well. The end result blurs the line between an evolutionary new product and a revolutionary one, with vastly more bandwidth and performance potential than other microprocessor in a single CPU socket.

Core i7 has almost 731 million transistors arranged into a 263 mm^2 area via the same $45nm$, high-k fabrication process used to produce "Penryn" Core 2 chips. Penryn has roughly 410 million transistors and a die area of 107 mm^2 and it takes two Penryn dies to make one quad-core product.

Core i7 has large L3 shared cache memory. This L3 cache is the last level of a fundamentally reworked cache hierarchy. Inside of each core is a 32 kB L1 instruction cache, a 32 kB L1 data cache (it's 8-way set associative), and a dedicated 256

kB L2 cache (also 8-way set associative). Outside of the cores is the L3, which is much larger at 8 MB and smarter (16-way associative) than the L2s. The Core i7's cache setup differs from the Phenom's in key respects, though, including the fact that it's inclusivethat is, it replicates the contents of the higher level cachesand runs at higher clock frequencies. As a result of these and other design differences, including a revamped TLB hierarchy, the Core i7's cache latencies are much lower than the Phenom's, even though its L3 cache is four times the size.

One mechanism Intel uses to make its caches more effective is prefetching [30], in which the hardware examines memory access patterns and attempts to fill the caches speculatively with data that's likely to be requested soon. Intel claims the Core i7's prefetching algorithm is both more efficient than Penryn'ssome server admins wound up disabling hardware prefetch in Xeons because it harmed performance with certain workloads, a measure Intel says should no longer be neededand more aggressive, as well.

The Core i7 can get to main memory very quickly which eliminates the chip-to-chip "hop" required when going over a front-side bus to an external north bridge. Officially, the maximum memory speed supported by the first Core i7 processors is 1066 MHz, which is a little conservative for DDR3, but frequencies of 1333, 1600, and 2000 MHz are possible with the most expensive Core i7, the 965 Extreme Edition.

With the memory controller onboard and the front-side bus, the Core i7 communicates with the rest of the system via the Quick Path interconnect (QPI). Quick Path is Intel's answer to Hyper Transport, a high-speed, narrow, packet-based, point-to-point interconnect between the processor and the I/O chip. The QPI link on the Core i7-965 Extreme operates at 6.4 GT/s. At 16 bits per transfer, that adds up to 12.8 GB/s, and since QPI links involve dedicated bidirectional pairs, the total bandwidth is 25.6 GB/s. Lower-end Core i7 processors have 4.8 GT/s QPI links with up to 19.2 GB/s of bandwidth.

This first, high-end desktop implementation of Nehalem is code-named Bloomfield, and it's essentially the same silicon that should go into two-socket servers eventually. As a result, Bloomfield chips come with two QPI links onboard, as the die shot above indicates. However, the second QPI link is unused. In 2P servers

based on this architecture, that second interconnect will link the two sockets, and over it, the CPUs will share cache coherency messages and data again.

In order to take advantage of this radically modified system architecture, the design team tweaked Nehalem's processor cores in a range of ways big and small. Although the Core 2's basic four-issue-wide design and execution resources remain more or less unchanged, almost everything around the execution units has been altered to keep them more fully occupied. The instruction decoder can fuse more types of x86 instructions together and, unlike Core 2, it can do so when running in 64-bit mode. The branch predictor's accuracy has been enhanced, too. Many of the changes involve the memory subsystemnot just the caches and memory controller but inside the core itself. The load and store buffers have been increased in size, for instance. A single Core i7 processor supports a total of eight threads.

The Core i7 processors series is recommended for multitasking, multithreading applications, extreme 3D gaming, creating professional movies and editing graphical tasks etc. [31]

1.3 Objectives

The objectives of this thesis are as follows:

- To design a low cost Reversible Arithmetic Logic Unit (RALU).
- To construct a cost efficient Reversible Control Unit (RCU).
- To find an efficient technique to combine the RALU, RCU and other components to construct a low cost RCPU.
- To optimize the number of reversible gates, garbage outputs, quantum cost, area, power and delay of the RCPU by optimizing all of the individual components of RCPU.

1.4 Methodologies of this Research

While working on this research, the following steps were followed:

- Studying the existing Reversible Central Processing Units (RCPU), Reversible Arithmetic Logic Units (RALU) and Reversible Control Units (RCU) and find the drawbacks of existing circuits.
- Proposing an overall block diagram of the RCPU which overcomes the problems of the existing RCPUs.
- Proposing the block diagram of the RALU.
- Designing the optimized individual circuits of the proposed RALU such as adder, subtractor, multiplier, divider, shifter, rotator, comparator, logical unit etc.
- Analyzing the proposed components in terms of cost and performance efficiency.
- Proposing the block diagram of the RCU.
- Proposing a new reversible gate to construct a cost effective Reversible Flip-Flop (RFF) which is used as the basic elements of the reversible memory circuits such as instruction registers and sequence counters.
- Proposing the cost effective designs of the individual elements of the proposed RCU such as decoders, counters, registers, control logic gates etc.
- Analyzing the proposed components in terms of cost and performance efficiency.
- Designing other components of the RCPU such as multiplexer, registers etc.
- Placing the proposed RALU, RCU and other components of the proposed RCPU efficiently and layout the optimized datapath which can handle all of the operations of the proposed RCPU.
- Realizing the overall architecture and organization of the proposed RCPU.

1.5 Outline of the Dissertation

The remainder of this dissertation is organized as follows:

- Chapter 2 **"Basic Concepts of Reversible Logic"** provides the basic definitions and ideas of reversible logic. With this introductory part the reader becomes familiar with the objective and notations of reversible logic theory.
- Chapter 3 **"Literature Review on Reversible Central Processing Unit"** discusses the existing works on reversible arithmetic logic unit, reversible control unit and reversible central processing unit. It also describes the advantages and disadvantages of the existing works in literature.
- Chapter 4 **"Proposed Components for Reversible CPU"** presents the proposed designs of various components of the proposed reversible central processing unit. It presents the comparative analysis of proposed basic components with existing approaches and also includes simulation results.
- Chapter 5 **"Implementation of Proposed Reversible CPU"** presents the proposed designs of reversible arithmetic logic unit, eversible control unit and the datapath construction and integration method of the proposed RALU and RCU.
- Chapter 6 **"Conclusions and Future Works"** highlights the accomplishments and limitation of this work and points to the directions of further research in reversible logic area.

Chapter 2

Basic Concepts of Reversible Logic

Preservation of information is a key motivation for research in reversible computing. Reversible logic gates are designed as a method to reduce the energy dissipation of logic circuits based on Launder's [15] concept. In 1961, he stated that all logical operations are associated with a physical operation and since physical irreversibility requires heat generation, then so does logical irreversibility. According to Launder's [15] research, each bit of information loss necessitates at least $KT\ln 2$ Joules of energy dissipation, where K is the Boltzmann constant and T is the operating temperature. In room temperature T, the amount of dissipating heat is small (i.e. 2.9×10^{-21} Joules), but not negligible. This amount may not seem to be significant, but it will become relevant in the future. According to Gordon Moore's [32] prediction, the heat dissipated by the device will be equal to a nuclear reactor by the end of 2050. The problem of power dissipation can be solved by preserving information which leads to the development of reversible logic. Moore's law predicts exponential growth of the heat generated due to the information loss, which will be a noticeable amount of heat loss in the next decade.

Arithmetic circuits play a crucial role in VLSI technology. Among various arithmetic circuits, multiplier and barrel shifter are two important circuits that can be realized with low power dissipation. Multiplier is very useful in many high performance systems such as FIR filters, microprocessors, digital signal processors

etc. A system's performance is mostly dependent on multiplier since it is regarded as the slowest element in the system. Shifter is another important circuit which is used in various applications such as floating point normalization, bit-indexing, variable-length coding etc. Usually digital signal processor and some high performance processors use barrel shifter which can shift multiple bits in a single cycle. This thesis presents logically reversible signed multiplier and bidirectional barrel shifter which do not erase a bit of information and as a result solve at least two problems: overheating and power saving.

This chapter presents the basic terminologies of reversible logic along with some popular reversible logic gates relevant to this research work. Illustrative figures and examples are included in respective sections.

2.1 Boolean Logic

Boolean Logic is a form of algebra in which all values are in two forms: TRUE or FALSE. Boolean logic is specially important for computer science because it matches with the binary number system, in which each bit has a value of either 1 or 0. There are two Boolean constants, 0 and 1. Boolean algebra was introduced by the mathematician George Boole. A Boolean function $f(x_1, x_2, ..., x_n)$ of n variables can be represented by a truth table with n input columns and one output column.

Example 2.1:
Let, $F(A, B, C) = B + AC$, where A, B and C are Boolean variables and F is the Boolean function. The output will be high if $B = 1$ or $AC = 1$ or both are 1. The truth table of this function is shown in Table 2.1. The number of rows in the truth table is 2^n where n is the number of input variables ($n = 3$ for the given equation). Hence there are $2^3 = 8$ possible input combination of inputs.

An n-input k-output multiple output Boolean function ($f_1(x_1, x_2, ..., x_n)$; $f_2(x_1, x_2, ..., x_n)$; ...; $f_k(x_1, x_2, ..., x_n)$) can be represented in a truth table with $(n+k)$ columns, where the last k columns represent the function output for the input pattern contained in the first n columns.

TABLE 2.1: Truth Table of a Boolean Function

A	B	C	$F(A,B,C)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Example 2.2:

Let, $F1(A,B,C) = B + AC$ and $F2(A,B,C) = ABC$ where A, B and C are Boolean variables and $F1$ and $F2$ are multiple output Boolean function. The truth table of this multiple output function is shown in Table 2.2. The number of rows in the truth table is 2^n where n is the number of input variables ($n = 3$ for the given equation). Hence there are $2^3 = 8$ possible input combination of inputs. The number of columns in the truth table is $(n + k)$ where, k is the number of output functions ($k = 2$ for the given equation).

TABLE 2.2: Truth Table of a Multiple Output Boolean Function

A	B	C	$F1(A,B,C)$	$F2(A,B,C)$
0	0	0	0	0
0	0	1	0	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

2.2 Reversible Logic

The multiple output Boolean function $F(x_1, x_2, ..., x_n)$ of n Boolean variables is called reversible if: 1. the number of outputs is equal to the number of inputs;

2. there is a one-to-one relationship between the input and output sets; In other words, reversible functions are those that perform permutations of the set of input vectors.

2.3 Reversible Logic Gate

According to Kerntopf [33], a Boolean function is reversible if it maps each input assignment into a unique output assignment. So, an n-input n-output gate (circuit) is reversible if it realizes an $n \times n$ reversible function [33]. If $I_v = (I_1, I_2, ..., I_{n-1}, I_n)$ is the input vector and $O_v = (O_1, O_2, ..., O_{n-1}, O_n)$ is the output vector, then an $n \times n$ reversible gate (circuit) can be represented as $I_v \leftrightarrow O_v$. Figure 2.1 shows the realization of an $n \times n$ reversible logic gate.

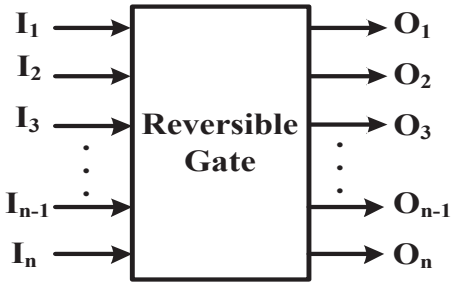

FIGURE 2.1: An $n \times n$ Reversible Gate

2.3.1 Garbage Output

Garbage output is the information that is not needed for the computation [34]. More formally, every output of a gate that is not used as a input to other gate or as a primary output is considered as garbage output. For example, Feynman gate [35] can be used to perform Exclusive-OR between two inputs. In this case, one extra output will be generated which is the garbage output in this regard. The garbage output of Feynman gate is marked as * in Figure 2.2.

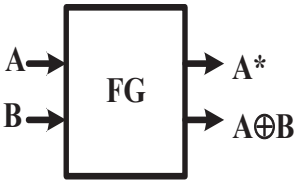

FIGURE 2.2: Garbage Output of Reversible Feynman Gate

2.3.2 Constant Input

In reversible circuits, some constant input bits are used in addition to the primary inputs to realize different logic functions which are referred to as ancilla inputs or constant inputs. The number of constant inputs should be minimum to design reversible logic circuits. Reversible Feynman gate [35] is used to copy an input when a constant 0 is fed to the second input as shown in Figure 2.3.

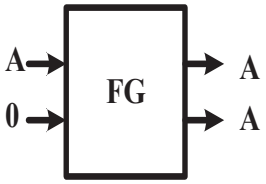

FIGURE 2.3: Constant Input of Reversible Feynman Gate

2.3.3 Delay

The delay of any reversible logic gate can be computed by calculating its logical depth [36]. The logical depth corresponds to the number of quantum gates in a path. Reversible 1×1 and 2×2 gates have unit delay. The delay is denoted by Δ. In this research work, logical depth is used as a measure of delay. For example, The Fredkin gate requires 5Δ delay as shown in Figure 2.4.

2.3.4 Area

The area of a reversible logic circuit is the summation of individual area of each reversible gate of the circuit. Suppose, a reversible circuit consists of n reversible gates and area of those n gates are $a_1, a_2, ..., a_n$. Then by using above definition,

area (A) of that circuit is

$$A = \sum_{i=1}^{n} a_i$$

Given the above definition the area of a circuit can be calculated easily by obtaining area of each individual gate using CMOS 45 nm Open Cell Library [37].

2.3.5 Power

The power of a reversible logic circuit is the summation of individual power of each reversible gate of the circuit. Suppose, a reversible circuit consists of n reversible gates and individual power of those gates are $p_1, p_2, ..., p_n$, respectively. Then, the power (P) of the circuit is

$$P = \sum_{i=1}^{n} p_i$$

The current rating of each gate can be obtained by using CMOS 45 nm Open Cell Library [37], Microwind DSCH 3.5 [38] and voltage is constant across each transistor, power requirement of each transistor can be calculated using the formula $P = V \times I$. Here, P refers to the power, V refers to the voltage and I refers to the current.

2.3.6 Quantum Cost

One of the key challenges in reversible logic design is to optimize quantum cost of a reversible circuit. The quantum cost of a reversible gate (circuit) can be calculated by counting the number of elementary quantum gates required to realize the reversible function [39]. The quantum gates are inherently reversible and manipulates qubits [40]. Just as a classical bit has a state either 0 or 1, a qubit also has a state either |0> or |1> [39]. The state of a qubit can be expressed like $|\psi> = \alpha\ |0> + \beta\ |1>$, where α and β are complex numbers and denotes the probability of having the state |0> or |1>. The most used elementary quantum gates are the NOT gate [35] (a single qubit is inverted), the Controlled-NOT (CNOT) gate [35] (the target qubit is inverted if the control qubit is 1), the Controlled-V gate [40] (also known as a square root of NOT), and the Controlled

V+ gate [40] (performs the inverse operation of Controlled-V gate and thus is also a square root of NOT). The Controlled-V and Controlled-V^+ are 1×1 gates having the following properties:

$$V \times V = NOT$$

$$V \times V^+ = V^+ \times V = I$$

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

Here, I represents an identity and NOT represents an inverse operation. According to the quantum circuit shown in Figure 2.4, the quantum cost of a 3×3 reversible Fredkin gate is 5 [41] as it requires two CNOT (marked as rectangular block), one Controlled-V and two NOT gates.

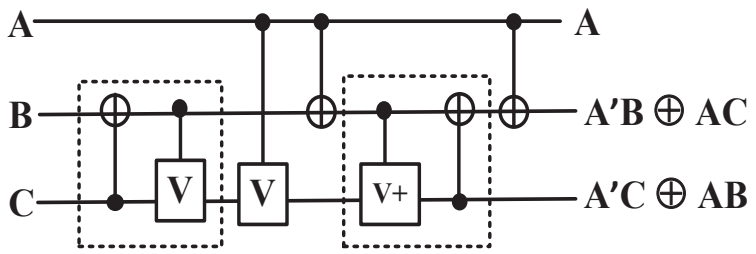

FIGURE 2.4: Quantum Realization of Reversible Fredkin gate

2.3.7 Popular Quantum Gates

A quantum gate or quantum logic gate is a rudimentary quantum circuit operating on a small number of qubits. They are the analogues for quantum computers to classical logic gates for conventional digital computers. Quantum logic gates are reversible, unlike many classical logic gates. In this section, some popular quantum gates are defined formally with their input-output vectors and quantum realizations.

Pauli-X Gate (NOT Gate)

The Pauli-X gate acts on a single qubit [42]. It is equivalent to NOT gate. It maps $|0>$ to $|1>$ and $|1>$ to $|0>$. It is represented by the Pauli Matrix is

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

Controlled-NOT Gate

A two-bit gate closely related to the NOT gate is the two-bit Controlled-NOT (or C-NOT) gate [42]. It performs a NOT on the second bit if the first bit is $< 1 >$, but otherwise has no effect. The CNOT is sometimes also called XOR, since it performs an exclusive OR operation on the two input bits and writes the output to the second bit. The graphical representation of this gate has two horizontal lines representing the two variable bits, and the conditionality of the operation is represented by the addition of a vertical line originating from the first bit and terminating with a NOT symbol on the second bit. The X symbol on the first bit indicates that this bit is being checked to see whether it is in $< 1 >$, before performing NOT on the second bit. It is represented by the matrix

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

Hadamard Gate

The Hadamard gate acts on a single qubit [42]. It maps $|0>$ to $\frac{|0>+|1>}{\sqrt{2}}$ and $|1>$ to $\frac{|0>-|1>}{\sqrt{2}}$, which means that a measurement will have equal probabilities to become 1 or 0. It is represented by the matrix

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

Swap Gate

The swap gate swaps two qubits [42]. With respect to the basis $|00>$—, $|01>$, $|10>$, $|11>$. it is represented by the matrix:

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

2.3.8 Popular Reversible Gates

In this section, some popular reversible logic gates needed in this work are defined formally with their input-output vectors and quantum realizations.

Feynman Gate

The Feynman gate (FG) [35], also known as CNOT gate, is a 2×2 reversible gate with the mapping of (A, B) to $(P = A, Q = A \oplus B)$, where A, B are the inputs and P, Q are the outputs. Figure 2.5 (a) and 2.5 (b), respectively show the block diagram and quantum circuit of FG gate. The quantum cost and delay of FG gate are 1 and 1Δ, respectively.

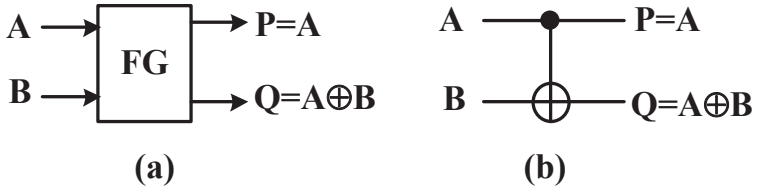

FIGURE 2.5: Reversible Feynman Gate: (a) Block Diagram and (b) Quantum Realization

Feynman Double Gate

The Feynman Double gate (F2G) [43] is a 3×3 reversible gate with the mapping of (A, B, C) to $(P = A, Q = A \oplus B, R = A \oplus C)$, where A, B, C are the inputs

and P, Q, R are the outputs. Figures 2.6 (a) and 2.6 (b) show the block diagram and quantum realization of F2G gate. The quantum cost and delay of F2G gate are 2 and 2Δ, respectively.

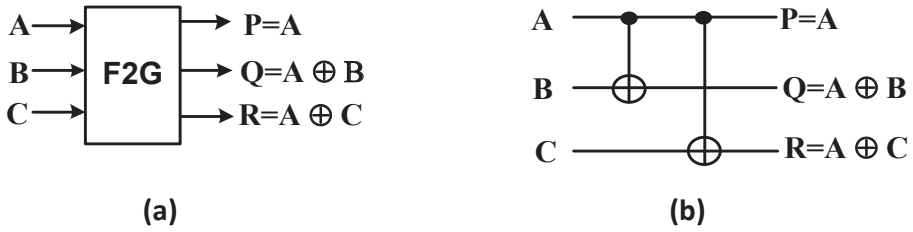

FIGURE 2.6: Reversible Feynman Double Gate: (a) Block Diagram and (b) Quantum Realization

Fredkin Gate

Fredkin gate (FRG) [14] is a 3×3 reversible gate that maps three inputs (A, B, C) to three outputs ($P = A$, $Q = A'B \oplus AC$, $R = A'C \oplus AB$) as shown in Figure 2.7 (a). The quantum cost and delay of FRG gate are 5 and 5Δ, respectively as shown in Figure 2.7 (b).

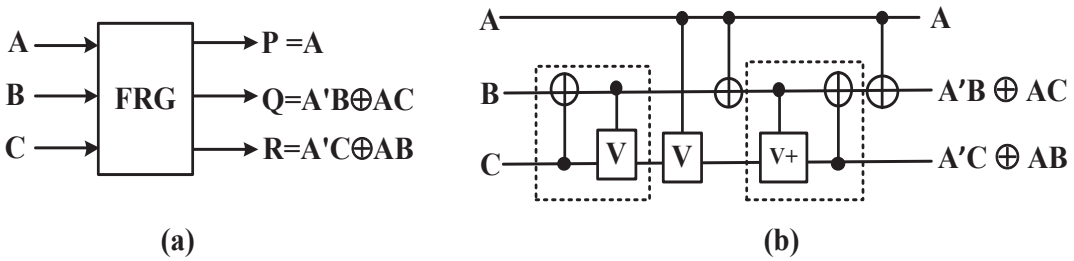

FIGURE 2.7: Reversible Fredkin Gate: (a) Block Diagram and (b) Quantum Realization

Modified Fredkin Gate

Modified Fredkin gate (MFG) [44] is a 3×3 reversible gate that maps three inputs (A, B, C) to three outputs ($P = A$, $Q = A'B \oplus AC'$, $R = A'C \oplus AB$). The block diagram and quantum circuit of reversible MFG gate is shown in Figures 2.8 (a) and 2.8 (b). The quantum cost and delay of MFG gate are 4 and 4Δ, respectively.

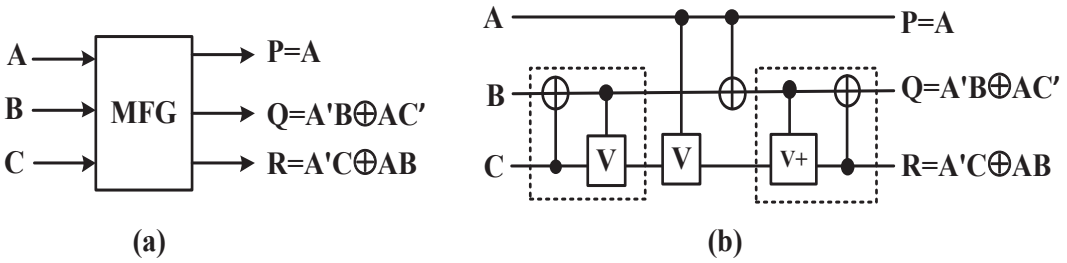

FIGURE 2.8: Reversible Modified Fredkin Gate: (a) Block Diagram and (b) Quantum Realization

Toffoli Gate

Toffoli gate (TG) [14] is a 3×3 reversible gate with the mapping of (A, B, C) to ($P = A$, $Q = B$, $R = AB \oplus C$), where A, B, C are the inputs and P, Q, R are the outputs. The quantum cost and delay of TG gate are 5 and 5Δ, respectively. Figure 2.9 (a) and 2.9 (b) show the block diagram and quantum realization of reversible TG gate.

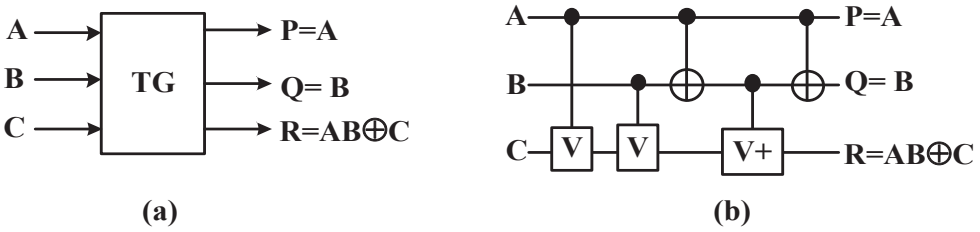

FIGURE 2.9: Reversible Toffoli Gate: (a) Block Diagram and (b) Quantum Realization

Peres Gate

Peres (PG) gate [45] is a 3 × 3 reversible gate that maps three inputs (A, B, C) to three outputs ($P = A$, $Q = A \oplus B$, $R = AB \oplus C$). The quantum cost of Peres gate is 4 and the delay is 4Δ. Figure 2.10 (a) and 2.10 (b) show the block diagram and quantum circuit of reversible PG gate.

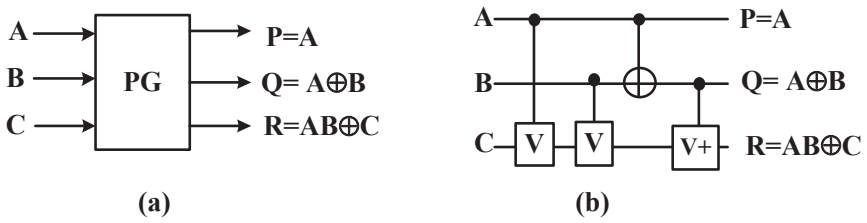

FIGURE 2.10: Reversible Peres Gate: (a) Block Diagram and (b) Quantum Realization

Double Peres Gate

Double Peres (DPG) gate [46] is a 4 × 4 reversible gate that maps four inputs (A, B, C, D) to four outputs ($P = A$, $Q = A \oplus B$, $R = A \oplus B \oplus D$, $(A \oplus B)D \oplus AB \oplus C$). The quantum cost of Double Peres gate is 6 and the delay is 6Δ. Figure 2.11 (a) and 2.11 (b) show the block diagram and quantum circuit of reversible DPG gate.

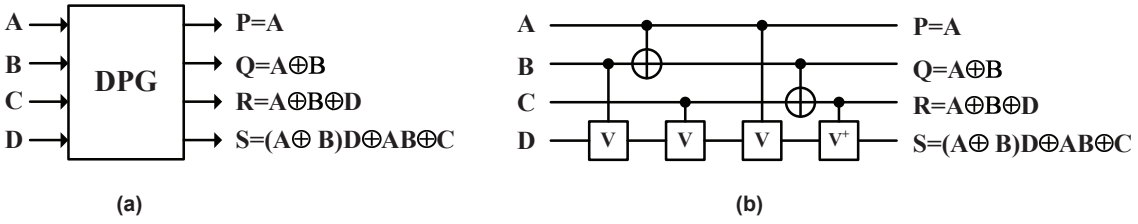

FIGURE 2.11: Reversible Double Peres Gate: (a) Block Diagram and (b) Quantum Realization

HNFG Gate

The HNFG gate [47] is a 4 × 4 reversible gate with the mapping of (A, B, C, D) to ($P = A$, $Q = A \oplus C$, $R = B$, $S = B \oplus D$), where A, B, C, D are the inputs and P, Q, R, S are the outputs. Figures 2.12 (a) and 2.12 (b) show the block diagram and quantum circuit of HNFG gate. The quantum cost and delay of HNFG gate are 2 and 2Δ, respectively.

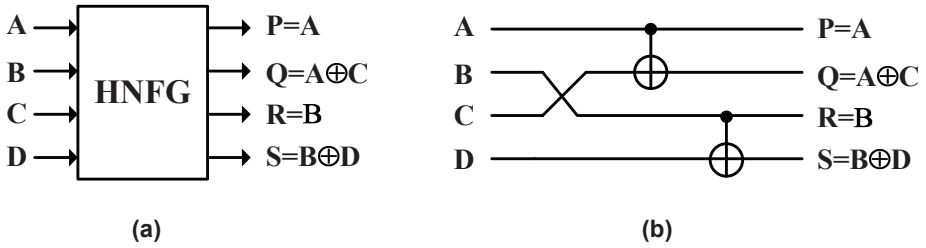

FIGURE 2.12: Reversible HNFG Gate: (a) Block Diagram and (b) Quantum Realization

BJN Gate

BJN gate [48] is a 3 × 3 reversible gate that maps three inputs (A, B, C) to three outputs ($P = A$, $Q = B$, $R = (A + B) \oplus C$) as shown in Figure 2.13 (a). The quantum cost and delay of BJN gate are 5 and 5Δ, respectively as shown in Figure 2.13.

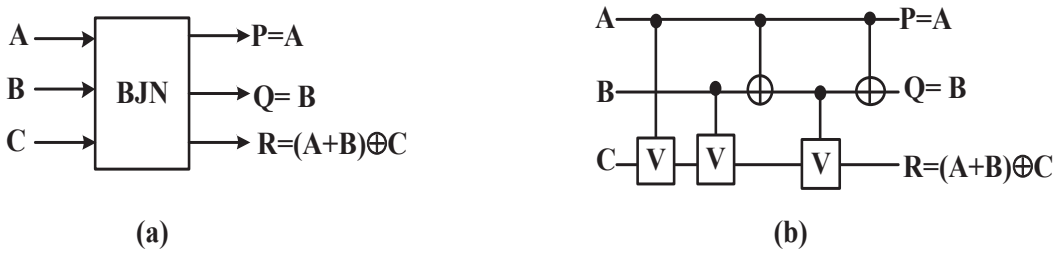

FIGURE 2.13: Reversible BJN Gate: (a) Block Diagram and (b) Quantum Realization

HNG Gate

HNG gate [12] is a 4 × 4 reversible gate that maps four inputs (A, B, C, D) to four outputs ($P = A$, $Q = B$, $R = A \oplus B \oplus C$, $S = (A \oplus B)C \oplus AB \oplus D$). The quantum cost and delay of HNG gate are 6 and 4Δ, respectively. Figure 2.14 (a) and 2.14 (b) show the block diagram and quantum realization of reversible HNG gate.

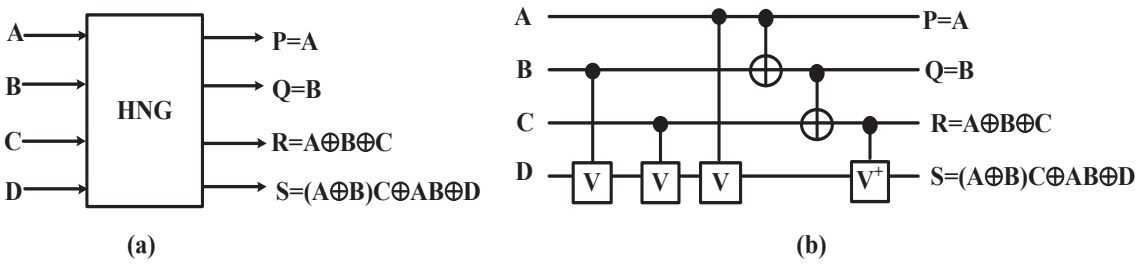

FIGURE 2.14: Reversible HNG Gate: (a) Block Diagram and (b) Quantum Realization

MIG Gate

MIG gate [49] is a 4 × 4 reversible gate that maps four inputs (A, B, C, D) to four outputs ($P = A$, $Q = A \oplus B$, $R = AB \oplus C$, $S = AB' \oplus D$). The quantum cost and delay of MIG gate is 7. Figure 2.15 (a) and 2.15 (b) show the block diagram and quantum realization of reversible MIG gate.

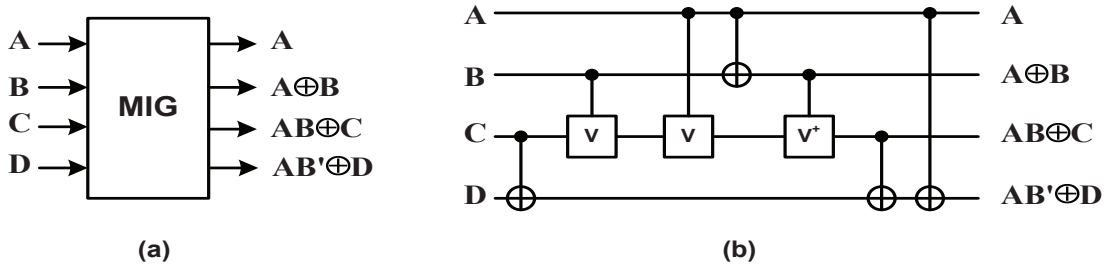

FIGURE 2.15: Reversible MIG Gate: (a) Block Diagram and (b) Quantum Realization

TR Gate

TR gate [50] is a 3 × 3 reversible gate that maps three inputs (A, B, C) to three outputs ($P = A$, $Q = A \oplus B$, $R = AB' \oplus C$). The quantum cost and delay of TR gate are 4 and 4Δ, respectively. Figure 2.16 (a) and 2.16 (b) show the block diagram and quantum realization of reversible TR gate.

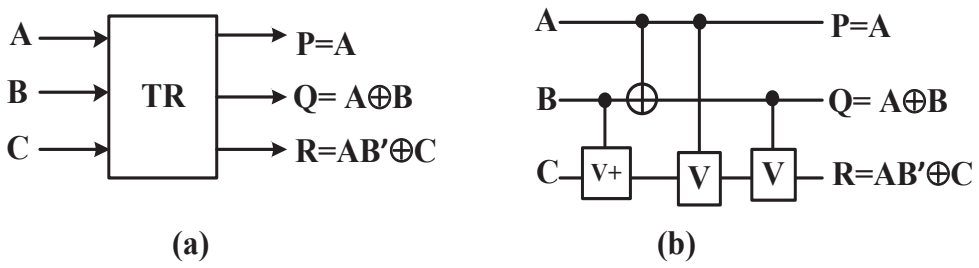

FIGURE 2.16: Reversible TR Gate: (a) Block Diagram and (b) Quantum Realization

RS Gate

RS gate (RSG) [51] is a 5×5 reversible gate that maps five inputs (A, B, C, D, E) to five outputs (A, B, C, $AB \oplus D$, $AC \oplus E$). The quantum cost and delay of RSG gate is 10 and 7Δ, respectively. Figure 2.17 (a) and 2.17 (b) show the block diagram and quantum realization of reversible RSG gate.

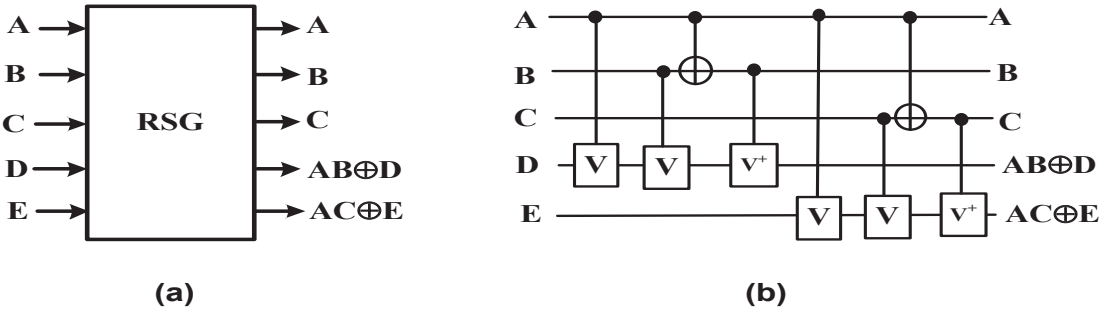

FIGURE 2.17: Reversible RSG Gate: (a) Block Diagram and (b) Quantum Realization

BVF Gate

The BVF gate [52] is a 4×4 reversible gate with the mapping of (A, B, C, D) to ($P = A$, $Q = A \oplus B$, $R = C$, $S = C \oplus D$), where A, B, C, D are the inputs and P, Q, R, S are the outputs. Figures 2.18 (a) and 2.18 (b) show the block diagram and quantum circuit of BVF gate. The quantum cost and delay of BVF gate are 2 and 2Δ, respectively.

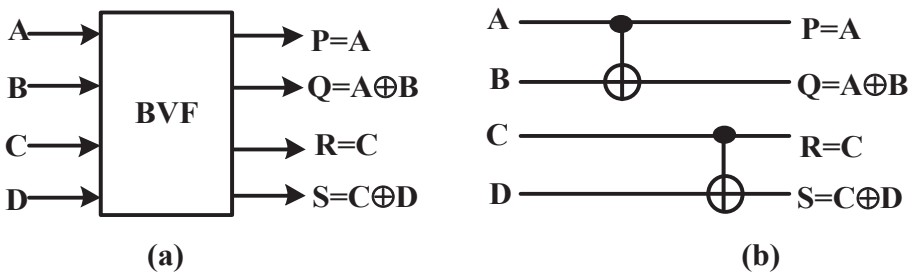

FIGURE 2.18: Reversible BVF Gate: (a) Block Diagram and (b) Quantum Realization

2.4 Summary

In this chapter, basic theory regarding reversible logic, quantum cost, garbage output, constant input, delay, area, power, reversible gates and quantum gates have been discussed. Some of the reversible gates, used to design our proposed reversible central processing unit, are discussed extensively.

Chapter 3

Literature Review on Reversible Central Processing Unit

A reversible central processing unit (RCPU) has a number of components. The first is the reversible arithmetic logic unit (RALU), which performs simple arithmetic and logical operations. Second is the reversible control unit (RCU), which manages the various components of the computer. It reads and interprets instructions from memory and transforms them into a series of signals to activate other parts of the computer. The control unit calls upon the arithmetic logic unit to perform the necessary calculations.

This chapter discusses the design methodologies of existing reversible central processing units. It also discusses different existing reversible arithmetic logic units and reversible control units. The advantages and disadvantages of corresponding existing circuits are also discussed.

3.1 Existing Reversible Arithmetic Logic Units

In [4], a new design of a reversible arithmetic logic unit for quantum arithmetic has been proposed. The authors proposed a 1-bit Reversible Arithmetic Logic Unit (RALU) which can perform some basic arithmetic and logic operations. The authors design the proposed RALU using quantum gates. The design is primarily

focused on the selected operations resulting from the combinations of the control lines for the reversible ALU. The proposed design can perform basic arithmetic and logic operations such as addition, subtraction, negation, AND, OR, NOT, EX-OR etc. The authors used the existing full adder (using MTSG gate) as the adder circuit of the proposed RALU. For subtract operation, was done using the same full adder. To achieve this operation, the second input bit is complemented to get the 1's complement. For negative subtract operation, the first input is complemented. The reaming procedure is same as the regular subtract operation. The carry input is asserted to 1 to get the 2's complemented form of the input. In this paper, The authors did not show any mechanism to interface the proposed RALU with Reversible Control Unit. Moreover, there was no comparative discussion of the proposed RALU with existing RALUs. Figure 3.1 shows the Proposed RALU by Zhou et al. [4].

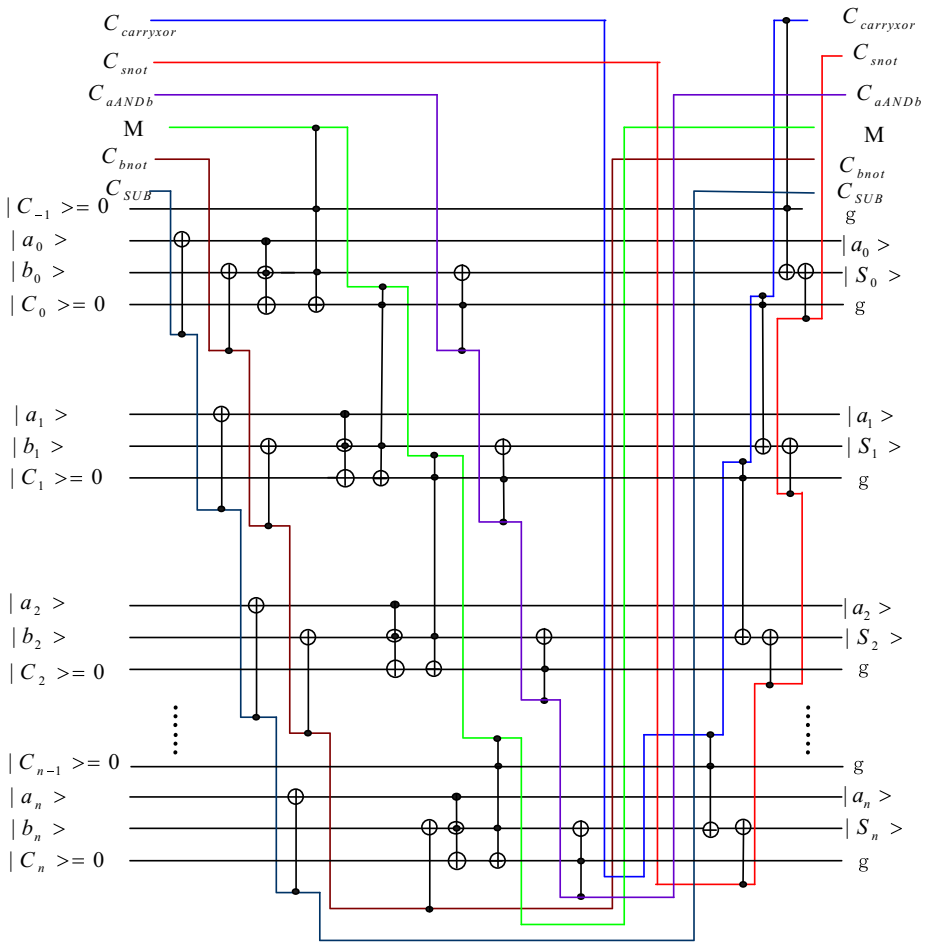

FIGURE 3.1: RALU proposed by Zhou et al. [4]

In [5], Morrison and Ranganathan proposed two design of 1-bit RALU each can perform six basic operations. The first design approach can perform ADD, SUB, EQUAL, XOR, OR and NOR operations. The second design approach can perform ADD, SUB, AND, NAND, OR and NOR operations. On the way to design the proposed arithmetic logic unit, the authors proposed two new reversible logic gates such as MRG gate and PAOG gate. The proposed MRG gate consists of three XOR gates, two Controlled-V and one Controlled-V+ gate. The quantum cost of the MRG gate is six. The MRG gate is used to calculate four logical calculations: OR, NOR, XOR and XNOR. The proposed PAOG gate is based on the Peres Gate. The cost and delay of the PAOG gate are identical to the MRG gate. MRG gate is used to calculate four logical calculations: OR, NOR, AND and NAND. Figure 3.2 shows the first design approach of the proposed RALU by Morrison et al. [5]. The cost of the first design approach of 1-bit ALU is 24, and the delay is 16. Figure 3.3 shows the second design approach of the proposed RALU by Morrison et al. [5]. The cost and delay of the second design approach of 1-bit ALU are same as the first design approach. Both of the designs are simple 1-bit design. They have only two arithmetic operations ADD and SUB.

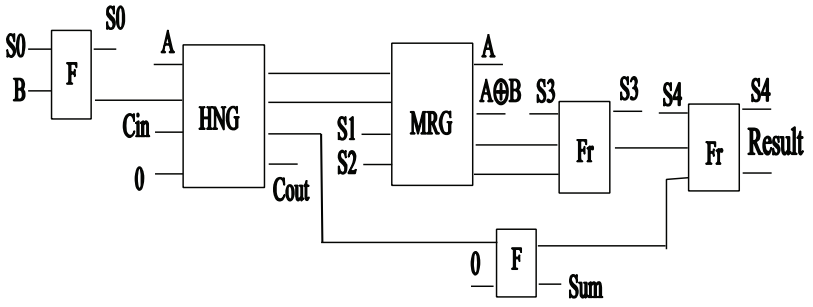

FIGURE 3.2: RALU proposed by Morrison et al. [5] (Approach 1)

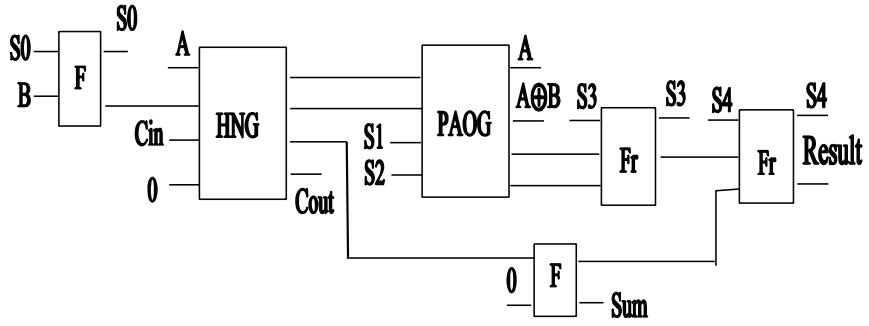

FIGURE 3.3: RALU proposed by Morrison et al. [5] (Approach 2)

In [6], Haghparast and Bolhassani proposed a 1-bit Quantum Reversible Arithmetic Logic Unit which can perform some arithmetic and logic operations. They presented three different approaches to design the Quantum RALU. In the first approach, the proposed reversible ALU has 8 inputs and 8 outputs. The elementary quantum gates are used to implementation the proposed ALU. The circuit can perform two arithmetic and four logical operations: ADD, SUB, AND, NAND, OR and NOR. The first design approach for one-bit reversible ALU is shown in Figure 3.4 which requires five dotted rectangles, four V, two V+ and eight CNOT gates. The quantum cost of this circuit is 19. In the second approach, the proposed reversible ALU has 9 inputs and 9 outputs. The second design appproach for one-bit reversible ALU is shown in Figure 3.5 which requires two dotted rectangles, eight V, four V+ and six CNOT gates. The quantum cost of this circuit is 24. There are 2 garbage values and 1 constant input. It has 9 circuit lines and 5 selection lines. It can perform two arithmetic and six logical operations ADD, SUB, AND, NAND, OR, NOR, EX-OR, EX-NOR. In the third design, the proposed reversible ALU has 7 inputs and 7 outputs. The third design approach of 1-bit reversible ALU is shown in Figure 3.6. Here, four V, two V+ and six CNOT gates are used. The quantum cost of this circuit is 12. The circuit has five select inputs. Total quantum cost of the circuit is tweleve. There is no garbage values and constant inputs. It can perform some basic operations such as, ADD, SUB, NSUB, XOR and NOP. All of the proposed designs are isolated Quantum RALUs, which lack the completeness.

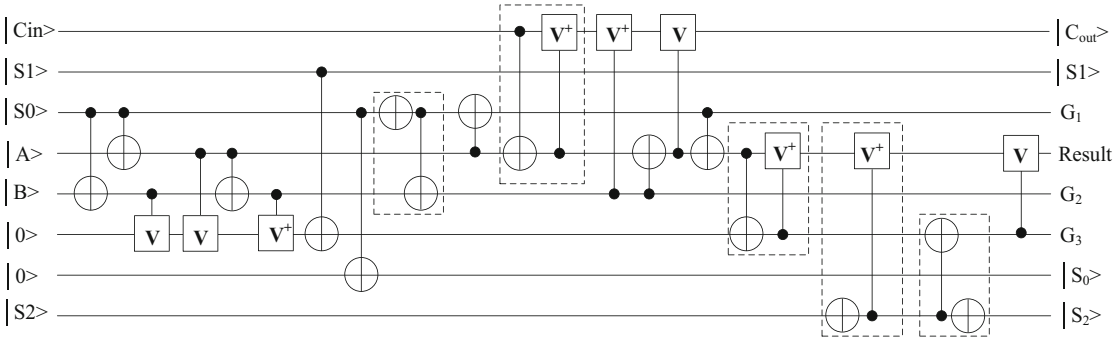

FIGURE 3.4: RALU proposed by Haghparast et al. [6] (Approach 1)

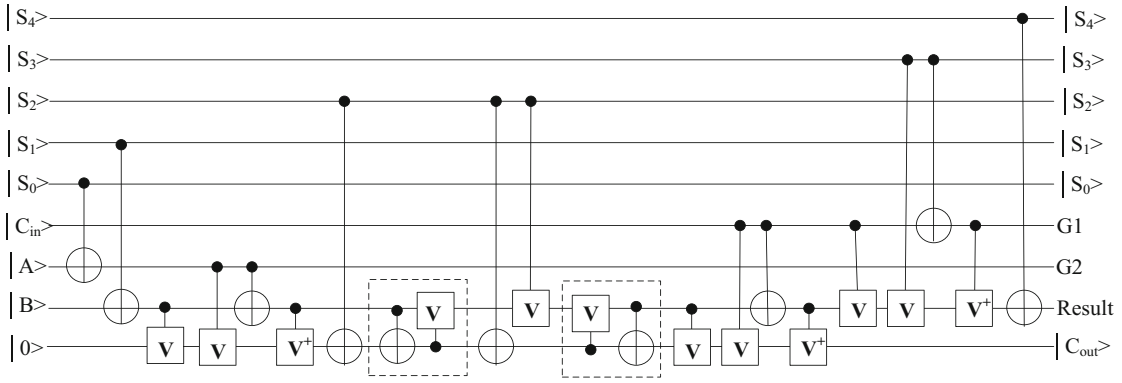

FIGURE 3.5: RALU proposed by Haghparast et al. [6] (Approach 2)

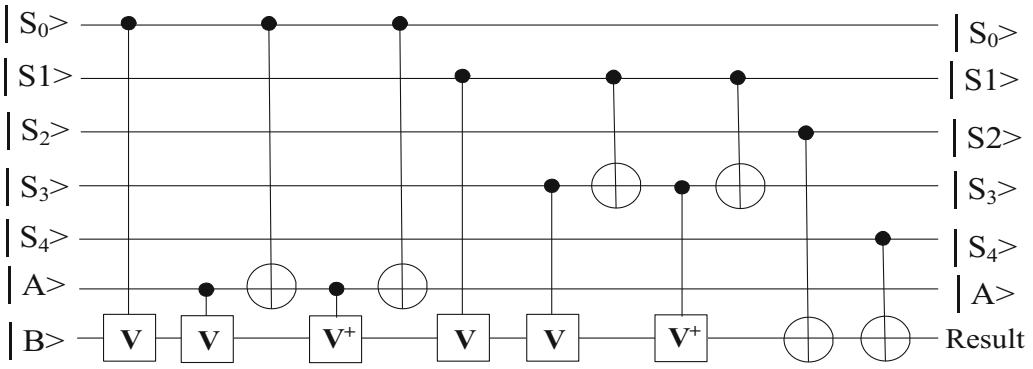

FIGURE 3.6: RALU proposed by Haghparast et al. [6] (Approach 3)

3.2 Existing Reversible Control Units

In [7], Aradhya, Kumar and Muralidhara proposed a Reversible Control Unit (RCU) for Reversible Arithmetic Unit. The authors proposed a reversible control logic which is basically designed to control a parallel adder and an arithmetic unit with some basic operations only. The designed circuit has two control signals with a provision for realizing some basic arithmetic operations such as TRANSFER, INCREMENT, DECREMENT and ADD. Two control variables and a carry in signal is used to control the operations. The RCU can not control any logic operations. The authors did not show any comparative analysis of the proposed

circuit with existing circuits. The design of the reversible control unit is shown in Figure 3.7.

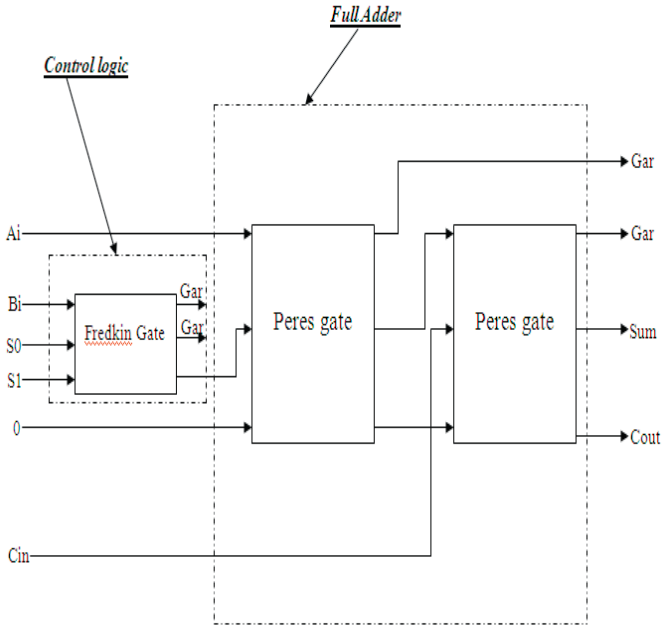

FIGURE 3.7: RCU proposed by Aradhya et al. [7]

3.3 Existing Reversible Central Processing Unit

In [53], Hall discussed a reversible processor architecture and algorithms based on the PDP-10 instruction set. The decision to use a CISC instruction set allows shorter code, but for this thesis, a more straightforward RISC style makes the data path and controller design simpler. Hall does not suggest even a block diagram level design for a processor. While he claims that intermediate results produced during some operations, such as effective address calculation, can be reversibly undone more easily in a CISC machine, he does not consider that a suitably restricted instruction set could effectively eliminate these intermediate results while using a simpler data path. This work is mainly an software approach. It lacks the circuit level design.

In [8], Thomsen et al. proposed an overall block diagram and datapath of a Reversible Central Processing Unit (RCPU). Their proposed RCPU can perform addition, subtraction, negation, exclusive-or and exclusive-or immediate (XORI)

operation. The left shifting operation was implemented by multiplication and the right shifting operation was implemented by division by 2. They did not show the design of any individual components of the RCPU. They mainly discuss about the software part of the instruction decoding to convert a high level instruction to machine code. Figure 3.8 shows the Proposed RCPU by Thomsen et al. [8]. The component level design of Figure 3.8 is absent in this work.

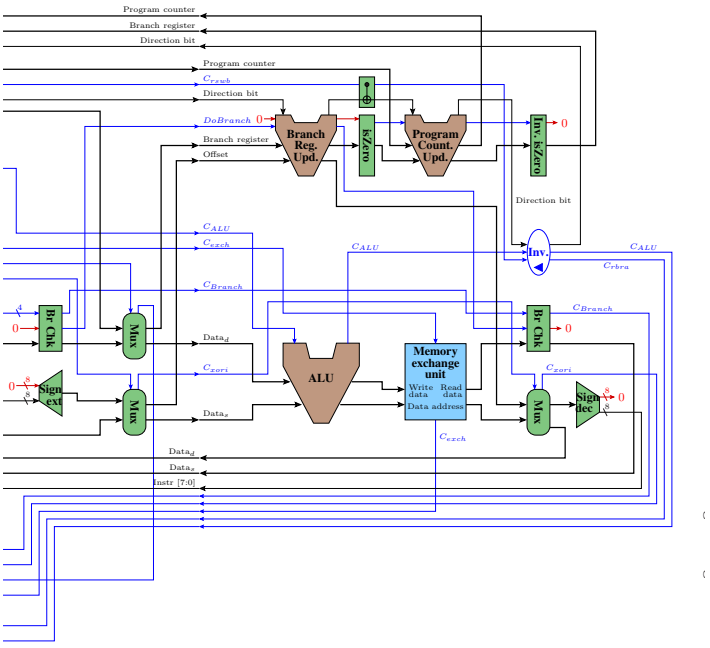

FIGURE 3.8: RCPU proposed by Thomsen et al. [8]

In [9], the authors proposed a block diagram of the RALU which can perform only add, sub, increment and compare operation. The authors show a comparative performance between a Reversible ALU and an Irreversible ALU with same operations without mentioning the implementation procedure of the RALU. Figure 3.9 shows the Proposed RCPU by Ilammathi et al. [9].

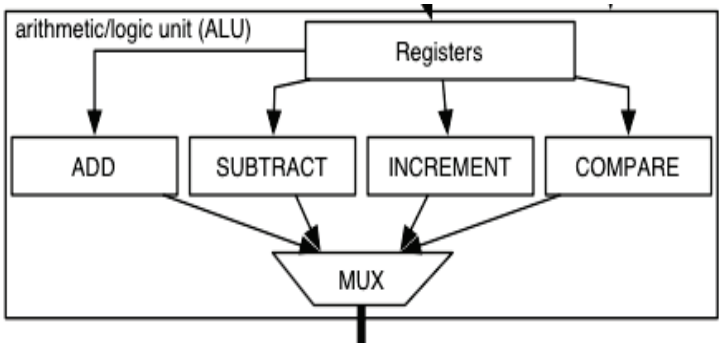

FIGURE 3.9: RCPU proposed by Ilammathi et al. [9]

In [10], the authors proposed a block diagram of the RCPU which can perform some arithmetic and logical operations such as add, sub, shift, rotate, jump, load, AND, OR, NOT, XOR etc. The respective assembler programs are transformed into sequences of binary instruction words, which are processed by the proposed RCPU. The CPU has been designed as a Harvard architecture, where the bit-width of both, the program memory and the data memory, is 16 bit. The size of the program memory is 4 kB, while the size of the data memory is 128 kB. It has 8 registers. The proposed RCPU architecture is divided into two basic types of components: combinational components and sequential components. The combinational components include the control unit, the ALU, the program counter, and the register file etc. The sequential components include the flip-flops and registers. The authors show the detail design of adder and program counter. The circuit design for the other operations of the RCPU were not mentioned. Figure 3.10 shows the Proposed RCPU by Wille et al. [10].

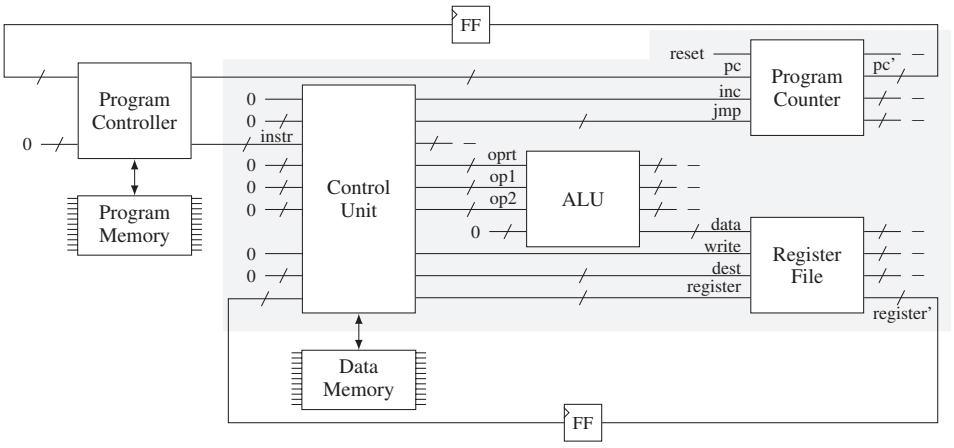

FIGURE 3.10: RCPU proposed by Wille et al. [10]

In [11], the authors proposed a block diagram of the RCPU. They also show the block diagram of the RALU. The authors main goal was to minimize the number of garbage outputs only which r eventually increases the other cost parameters. The detail design of the components were absent in the work. Figure 3.11 shows the Proposed RCPU by Thomsen et al. [11].

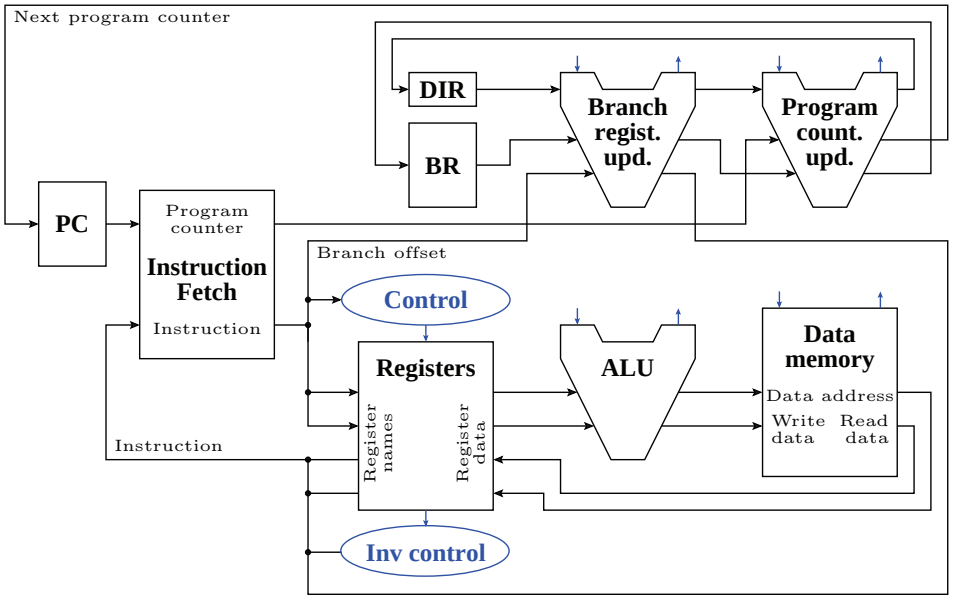

FIGURE 3.11: RCPU proposed by Thomsen et al. [11]

3.4 Summary

This chapter discusses the design procedure of existing reversible arithmetic logic unit, reversible control unit and reversible processor. The advantages and limitations of each design are also discussed in this chapter.

Chapter 4

Proposed Components for Reversible CPU

In this chapter, the design methodologies of different memory components and arithmetic components of the proposed reversible central processing unit has been presented. Each component has been compared with their existing counterparts. The complexities of the circuits have been proved and the simulation of different components have been provided.

4.1 Proposed Reversible Memory Components

A Reversible Central Processing Unit (RCPU) requires various memory components such as instruction register, stack pointer, program counter, accumulator register, flag register, register files and cache memory. Reversible flip-flop is the basic component of a reversible memory circuit. A flip-flop is a circuit that has two stable states and can be used to store state information [54]. If the flip-flop is designed in an optimized way, the whole memory circuit will be optimized.

4.1.1 Proposed Reversible J-K Flip-Flop

The J-K flip-flop (FF) is the most versatile of the basic flip-flops. It has the input-following character of the clocked D flip-flop but has two inputs, traditionally labeled J and K. If J and K are different then the output Q takes the value of J at the next clock edge. If J and K are both low then no change occurs. If J and K are both high at the clock edge then the output will toggle from one state to the other. It can perform the functions of the set/reset flip-flop and has the advantage that there are no ambiguous states [54]. It can also act as a T flip-flop to accomplish toggling action if J and K are tied together. This toggle application finds extensive use in binary counters. The characteristic equation of a J-K flip-flop is

$$Q_{next} = JQ' + K'Q \tag{4.1}$$

To design an efficient reversible J-K FF, a new reversible gate, namely BJ Gate, has been proposed. BJ gate is a 4 × 4 reversible gate with the mapping of (A, B, C, D) to ($P = A$, $Q = AB \oplus C$, $R = A'B \oplus AC'$, $S = A'B \oplus AC' \oplus D$), where A, B, C, D are the inputs and P, Q, R, S are the outputs. The block diagram of the proposed gate is shown in Figure 4.1. It can be verified from the corresponding truth table (shown in Table 4.1) that 16 input and 16 output combinations of the proposed 4×4 reversible BJ gate have one-to-one mapping between them. The quantum cost of the BJ gate is twelve. The proposed BJ gate can be used as universal gate. The NAND implementation of the BJ gate is shown in Figure 4.2.

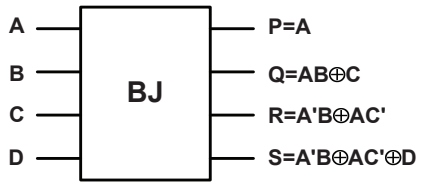

FIGURE 4.1: Block Diagram of Proposed Reversible BJ Gate

The proposed BJ gate is used to construct a J-K FF. From the characteristic equation of J-K FF (Eq. 4.1), it can be found that $Q_{next} = JQ' + K'Q$. If J is

TABLE 4.1: Truth Table of 4×4 Reversible BJ gate

INPUT				OUTPUT			
A	B	C	D	$P = A$	$Q = AB \oplus C$	$R = A'B \oplus AC'$	$S = A'B \oplus AC' \oplus D$
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	1	0	0
0	0	1	1	0	1	0	1
0	1	0	0	0	0	1	1
0	1	0	1	0	0	1	0
0	1	1	0	0	1	1	1
0	1	1	1	0	1	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	0	1	0
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	1	0	0	1	1	1	1
1	1	0	1	1	1	1	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	1

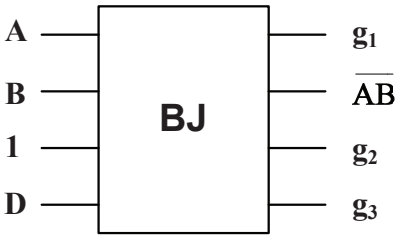

FIGURE 4.2: NAND Implementation of BJ Gate

asserted in the second input, K in the third input and CLK in the fourth input of the BJ Gate and the third output is feeded to the first input of the BJ gate, it will produce Q in the second output and Q' in the fourth output. Figure 4.3 shows the design of the proposed reversible J-K FF. The comparative results of the proposed and the existing reversible J-K FFs [55], [56], [57] are shown in Table 4.2.

4.1.2 Proposed Reversible Instruction Register

Instruction register is a high-speed circuit that holds an instruction for decoding and execution. A reversible 16-bit instruction register is designed using sixteen

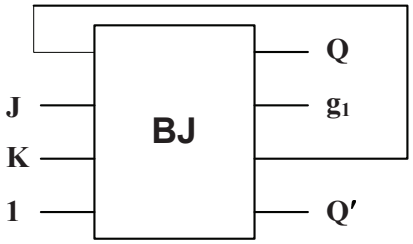

FIGURE 4.3: Design of the Proposed Reversible JK FF

TABLE 4.2: Comparison of Different J-K FFs

J-K FF with Q and Q'	No. of Gates	Garbage Outputs	Delay	Quantum Cost
Proposed Method	1	1	1	12
Existing Design [55]	3	3	3	14
Existing Design [56]	4	3	4	18
Existing Design [57]	10	12	10	30
Improvement(%) w.r.t [55]	66.66	66.66	66.66	14.28
Improvement(%) w.r.t [56]	75	75	75	33.33
Improvement(%) w.r.t [57]	90	91.66	90	60

Here, Q = Output of the J-K FF, Q' = Inverted Output of the J-K FF

HNFG gates and sixteen J-K FFs that was designed in the previous section. The J-K FFs take the inputs through HNFG gates and with the change of the clock pulse they produce the normal and complemented outputs. Figure 4.4 shows the design of the proposed reversible instruction register. The proposed circuit requires 32 reversible gates, it produces 17 garbage bits and the quantum cost of the circuit is 224.

The proposed instruction register can be expanded for any number of bits. An n-bit instruction requires n J-K FFs and n HNFG gates (total $2n$ gates), it produces $n + 1$ garbage outputs and it requires $12n + 2n = 14n$ quantum cost.

The performance comparison of instruction registers using proposed and existing techniques has been shown in the Table 4.3. Table 4.3 clearly depicts that the proposed design outperforms the existing ones in terms of numbers of gates (improvements are 50%, 60% and 83.33% with respect to [14], [15] and [16], respectively), garbage outputs (improvements are 65.3%, 65.3% and 91.83% with respect to [14], [15] and [16], respectively), delay (improvements are 65.3%, 73.85% and 89.44%

with respect to [14], [15] and [16], respectively) and quantum cost (improvements are 12.5%, 30% and 56.25% with respect to [14], [15] and [16], respectively).

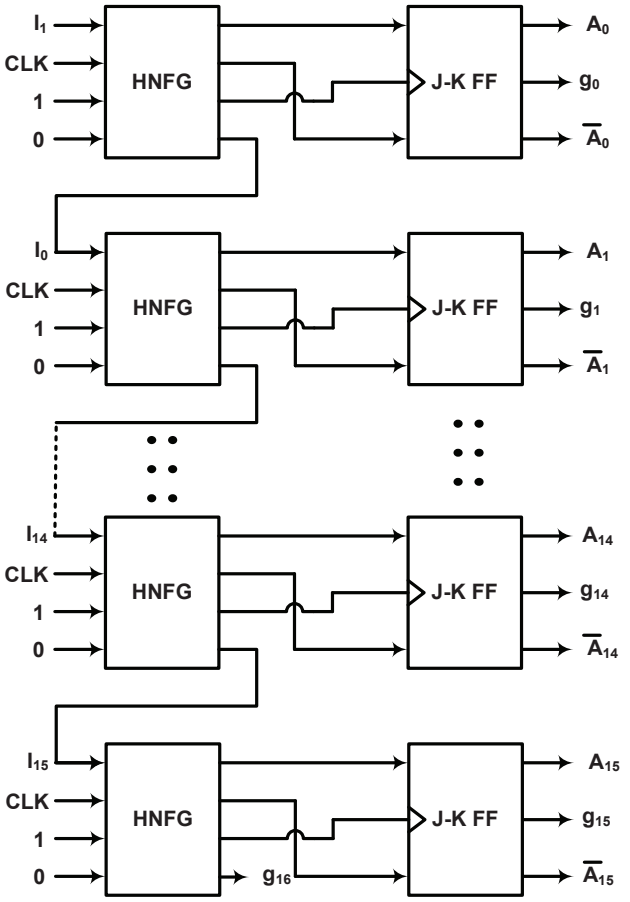

FIGURE 4.4: Proposed Reversible Instruction Register

TABLE 4.3: Comparison of Different 16-bit Reversible Instruction Registers

Instruction Register	No. of Gates	Garbage Outputs	Delay	Quantum Cost
Proposed Method	32	17	17	224
Existing Design [55]	64	49	49	256
Existing Design [56]	80	49	65	320
Existing Design [57]	192	208	161	512
Improvement(%) w.r.t [55]	50	65.3	65.3	12.5
Improvement(%) w.r.t [56]	60	65.3	73.85	30
Improvement(%) w.r.t [57]	83.33	91.82	90	6.25

4.1.3 Proposed Reversible D Flip-Flop

Unlike the J-K flip-flops, D flip-flop has only one synchronous control input, D, which stands for *data* [54]. This flip-flop stores the value that is on the data line. It can be thought of as a basic memory cell. A D flip-flop can be made from a set/reset flip-flop by tying the set to the reset through an inverter. If E is the enable input, the characteristic equation of D FF is written as

$$Q_{next} = E'Q \oplus DE \tag{4.2}$$

To design an efficient reversible D FF, a new reversible gate, namely NP Gate, has been proposed. NP gate is a 4 × 4 reversible gate with the mapping of (A, B, C, D) to ($P = A$, $Q = A'B \oplus AC'$, $R = A'C \oplus AB$, $S = A'C \oplus AB \oplus D$), where A, B, C, D are the inputs and P, Q, R, S are the outputs. The block diagram and quantum circuit of NP gate are shown in Figures 4.5(a) and 4.5(b) respectively. It can verified from the corresponding truth table shown in Table 4.4 that 16 input and 16 output combinations of the proposed NP gate have one-to-one mapping between them. So, NP gate satisfies the condition of reversibility. The quantum cost of the NP gate is five.

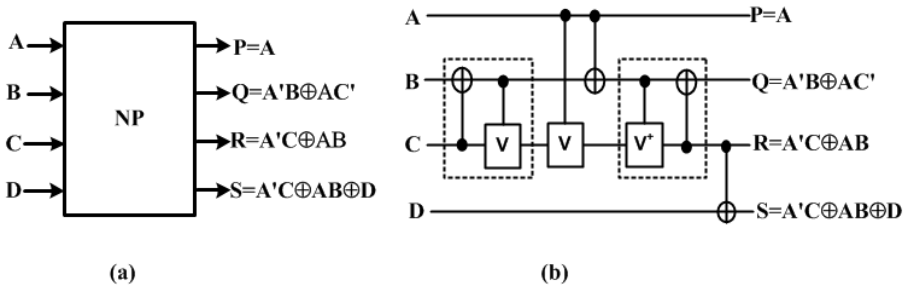

FIGURE 4.5: Proposed NP Gate: (a) Block Diagram and (b) Quantum Realization

The proposed NP gate is used to construct a D FF. From the characteristic equation of D FF (Eq. 4.2), it is found that $Q_{next} = E'Q \oplus DE$. Figure 4.6(a) shows the construction of D FF with output Q. The complement output Q' can be produced by adding a Feynman gate as shown in Figure 4.6(b). These two different designs are compared with existing designs in Table 4.5 and 4.6.

TABLE 4.4: Truth Table of 4×4 Reversible NP gate

INPUT				OUTPUT			
A	B	C	D	$P = A$	$Q = A'B \oplus AC'$	$R = A'C \oplus AB$	$S = A'C \oplus AB \oplus D$
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	1
0	1	1	0	0	1	1	1
0	1	1	1	0	1	1	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	1
1	1	0	0	1	1	1	1
1	1	0	1	1	1	1	0
1	1	1	0	1	0	1	1
1	1	1	1	1	0	1	0

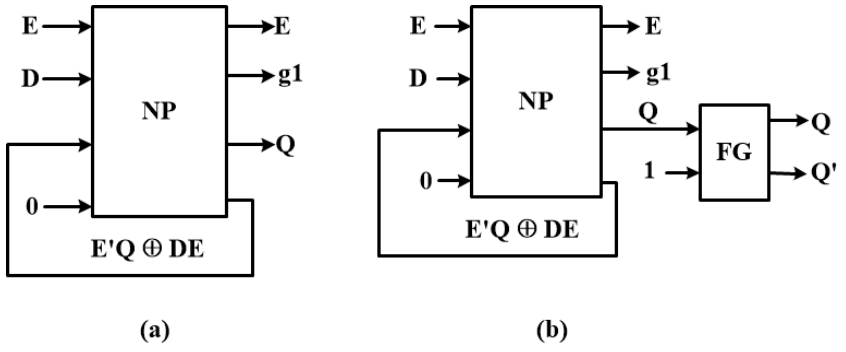

FIGURE 4.6: Proposed Reversible D FF (a) with Output Q and (b) with Output Q and Q'

4.1.4 Proposed Reversible Data Register

Register is the fundamental unit of memory. In this section, a 2-bit reversible data register and a generalized m-bit reversible reversible register have been designed using proposed reversible D FF and FRG [14] gate. The FRG gate is used as output buffer. There are two data inputs (D_0, D_1) and two data outputs (Q_0, Q_1) in a 2-bit register. The proposed design of 2-bit reversible register is shown in Figure 4.7. A 2-bit reversible data register requires two D FF and two FRG gates. Figure 4.6(a) shows that a D FF requires only one gate. So, total number of gates

TABLE 4.5: Comparison of Reversible D FFs with Q

D FF with Q	No. of Gates	Garbage Outputs	Quantum Cost	Delay
Proposed Circuit	1	2	5	5Δ
Existing Circuit [55]	1	2	6	6Δ
Existing Circuit [56]	2	2	6	6Δ
Existing Circuit [58]	2	2	6	6Δ
Existing Circuit [59]	2	2	6	6Δ
Improvement(%)w.r.t [55]	0	0	16.67	16.67
Improvement(%)w.r.t [56]	50	0	16.67	16.67
Improvement(%)w.r.t [58]	50	0	16.67	16.67
Improvement(%)w.r.t [59]	50	0	16.67	16.67

Here, Q = Output of the D FF

TABLE 4.6: Comparison Reversible D FFs with Q & Q'

D FF with Q & Q'	No. of Gates	Garbage Outputs	Quantum Cost	Delay
Proposed Circuit	2	1	6	6Δ
Existing Circuit [60]	2	2	7	7Δ
Existing Circuit [55]	2	2	7	7Δ
Existing Circuit [56]	3	2	7	7Δ
Existing Circuit [61]	2	2	9	9Δ
Improvement(%)w.r.t [60]	0	50	14.28	14.28
Improvement(%)w.r.t [55]	0	50	14.28	14.28
Improvement(%)w.r.t [56]	0	0	14.28	14.28
Improvement(%)w.r.t [61]	0	50	33.33	14.28

Here, Q = Output of the D FF, Q' = Inverted Output of the D FF

of 2-bit reversible data register is (2+2) = 4. It produces five garbage outputs. The quantum cost of the proposed reversible 2-bit Data register is 20.

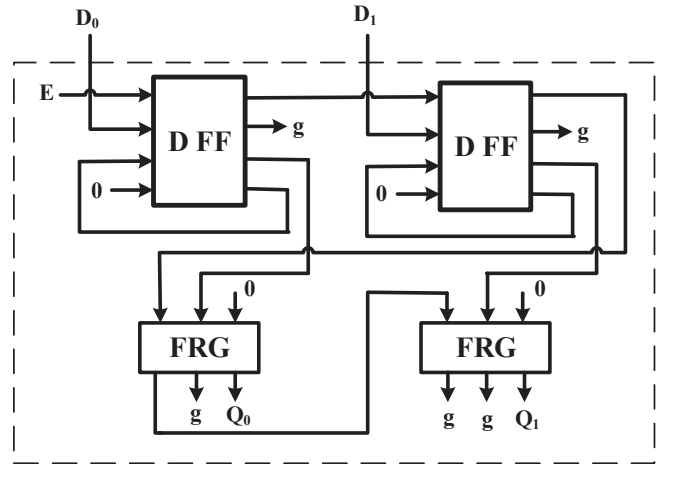

FIGURE 4.7: Proposed Reversible 2-bit Register

An m-bit reversible register requires m number of reversible D FF and m number of FRG gates. It has m number of data input lines $(D_0, D_1, ..., D_{m-1})$ and m data output lines $(Q_0, Q_1, ..., Q_{m-1})$. The proposed design of m-bit reversible register is shown in Figure 4.8. The proposed m-bit reversible register requires total $2m$ reversible gates. The quantum cost of this register is $10m$ and it produces total $2m + 1$ garbage outputs.

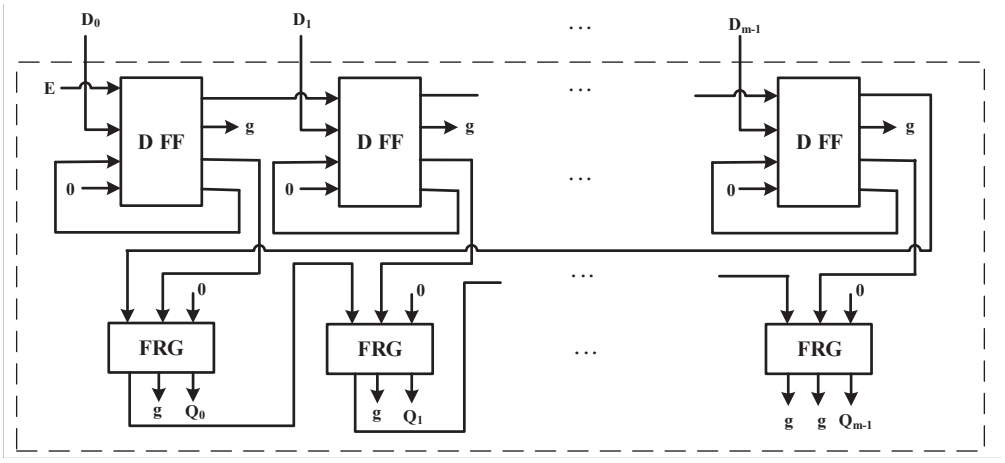

FIGURE 4.8: Proposed Reversible m-bit Register

4.2 Proposed Designs of the Components of Reversible Arithmetic Unit

The proposed Reversible Arithmetic Logic Unit has two main units: arithmetic unit and logic unit. The arithmetic unit performs addition, subtraction, increment, decrement, multiplication, division, comparison and shift operation. The basic components of the arithmetic units have been proposed in this section. The proposed circuits have been simulated and compared with existing designs. The cost complexities of the proposed circuits have been proved.

4.2.1 Proposed Reversible Carry Look-Ahead Adder

Carry-Lookahead adder is a fast parallel adder which reduces the propagation delay of the classical adder [62]. If input vector for classical full adder is, $I_v = (a, b, c_0)$

then, Sum (s_i) and Carry (c_{i+1}) can be computed from the Eq. 4.3 and Eq. 4.4, where c_0 is the initial carry whose value is zero.

$$s_i = a_i \oplus b_i \oplus c_i \tag{4.3}$$

$$c_{i+1} = a_i b_i \oplus b_i c_i \oplus c_i a_i \tag{4.4}$$

For the addition of two n-bit numbers ripple carry adder or carry look-ahead adder is a straight forward choice. In ripple carry adder (or serial carry and parallel output full adder) the previous carry (c_{i-1}) is calculated in order to compute the next one (c_i). This delay disadvantage can be prevented if this dependencies among carry bits [63] are forefended. Carry look-ahead adder actually skips this dependency among carry bits by two modules, namely, carry generation (G_i) and carry propagation (P_i) where,

$$G_i = a_i b_i \tag{4.5}$$

$$P_i = a_i \oplus b_i \tag{4.6}$$

Therefore, the Eq. 4.3 and Eq. 4.4 can be rewritten as,

$$s_i = P_i \oplus c_i \tag{4.7}$$

$$c_{i+1} = G_i \oplus P_i c_i \tag{4.8}$$

Then the series of iterative operations to expand Eq. 4.7 and Eq. 4.8 for creating carry look-ahead adder can be carried out as follows:

$$c_1 = G_0 \oplus P_0 c_0$$
$$c_2 = G_1 \oplus P_1 c_1$$
$$c_3 = G_2 \oplus P_2 c_2 = G_2 \oplus P_2(G_1 \oplus P_1 c_1) = G_2 \oplus P_2(G_1 \oplus P_1(G_0 \oplus P_0 c_0)$$

Therefore,

$$c_n = G_{n-1} \oplus P_{n-1}(G_{n-2} \oplus ... \oplus P_{n-2}P_{n-3}...P_0 c_0)$$
$$s_0 = P_0 \oplus c_0$$
$$s_1 = P_1 \oplus c_1 = P_1 \oplus G_0 \oplus P_0 c_0$$
$$s_2 = P_2 \oplus c_2 = P_2 \oplus G_1 \oplus P_1 c_1 = P_2 \oplus G_1 \oplus P_1(G_0 \oplus P_0 c_0) =$$

$$s_3 = P_3 \oplus c_3 = P_3 \oplus G_2 \oplus P_2 c_2 = P_3 \oplus G_2 \oplus P_2(G_1 \oplus P_1 c_1) = P_3 \oplus G_2 \oplus P_2(G_1 \oplus P_1 G_0 \oplus P_1 P_0 c_0)$$

Therefore,

$$S_n = P_n \oplus G_{n-1} \oplus P_{n-1}(G_{n-2} \oplus ... \oplus P_{n-2} P_{n-3} ... P_0 c_0)$$

To generate $P_i = a_i \oplus b_i$ and $G_i = a_i b_i$ reversible Peres gate is used which is shown in Figure 4.9. From Figure 4.9 it is found that, A Peres gate with third input set to zero can be used for skipping the carry, thus the reversible full adder will not wait for the carry from the lower digit. From Figure 4.9, it is clear that there is one garbage output and its quantum cost is 4. Architecture of proposed 4-bit reversible carry look-ahead adder is shown in Figure 4.10. Algorithm for proposed reversible n-bit carry look-ahead adder is given in Algorithm 1. Reversible PG (Shown in Figure 4.9) gate is used to generate P_i and G_i. Here Feynman gate is used as a copying gate to generate sufficient number of P_i and G_i. Finally, s_i is calculated with all this P_i, G_i and initial carry.

Algorithm 1 n-bit reversible carry look-ahead adder

Input: Initial carry $c_0 = 0$, two n-bit numbers $(a_{n-1}, ..., a_2, a_1, a_0)$ and $(b_{n-1}, ..., b_2, b_1, b_0)$
Output: An n-bit sum $(s_{n-1}, ..., s_2, s_1, s_0)$
for $i = 0 \rightarrow n - 1$
Generate P_i and G_i with the help of PG
With the help of FG generate P_i and G_i to calculate s_i
From generated P_i, G_i and with the supplied initial carry c_0 calculate s_i using reversible TG and FG
end for

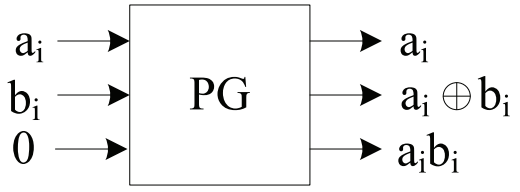

FIGURE 4.9: Reversible Peres gate as carry generator and propagator

Theorem 4.1. *Let GC be the required number of the gates for n-bit reversible carry look-ahead adder. Then,* $GC = n^2 + n$.

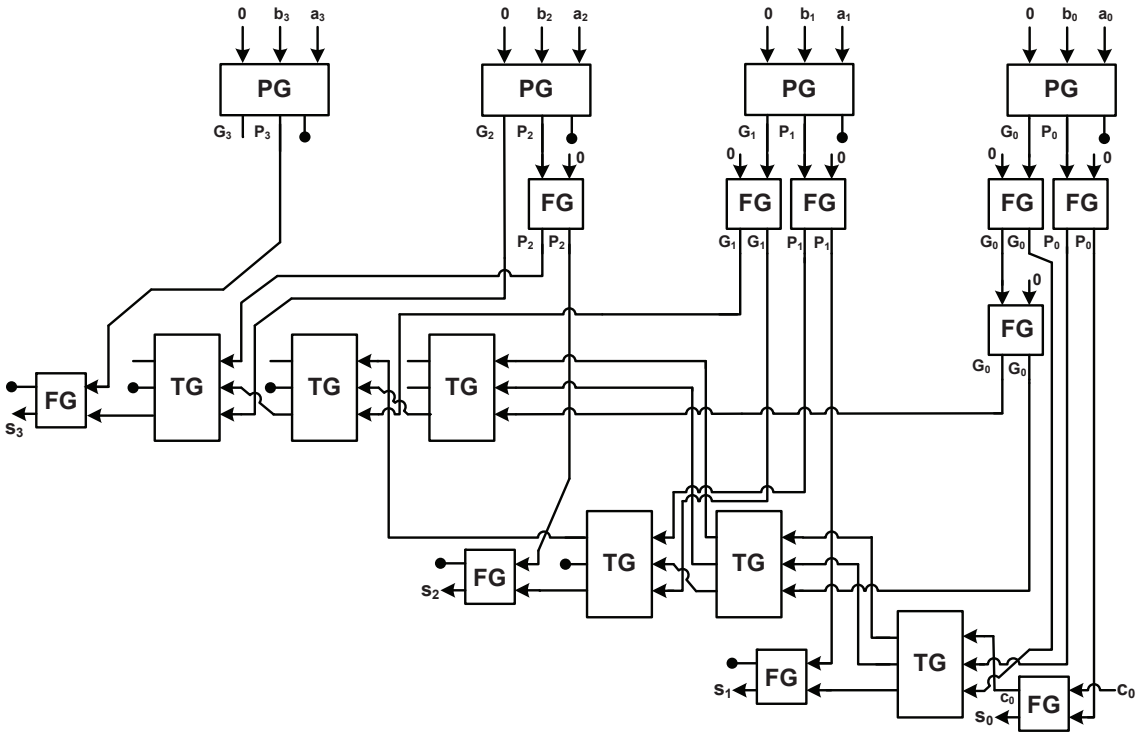

FIGURE 4.10: Proposed Reversible Carry Look Ahead Adder

Proof : For an n-bit carry look-ahead adder there are total of n carry propagator (P_i) and n carry generator (G_i). According to the design procedure, n Peres gate (PG) is required to generate n numbers of P_i and n numbers of G_i. To generate required number of P_i and G_i, $(n-i-1)$ Feynman gate (FG) is required for each i, where $i = 0$ to $n-1$. Finally, i numbers of TG and one FG gate is required to calculate each s_i except the first one. For the first s_i (i.e. s_0) only one FG gate is required. Therefore, total number of gates for n-bit reversible carry look-ahead adder is,

$$GC = nPG + \tfrac{n(n+1)}{2}FG + \tfrac{n(n-1)}{2}TG = n + \tfrac{n(n+1)}{2} + \tfrac{n(n-1)}{2} = n^2 + n.$$

Theorem 4.2. *Let, QC be the quantum cost for n-bit reversible carry look-ahead adder. Then, $Q = 3n^2 + 2n$.*

Proof: In Theorem 4.1, it is proved that total of $nPG + \frac{n(n+1)}{2}FG + \frac{n(n-1)}{2}TG$ gates are required for n-bit reversible carry look-ahead adder. The quantum cost of each PG, FG and TG are 4, 1 and 5 respectively. Therefore, total quantum cost for n-bit reversible carry look-ahead adder is,

$$4n + \frac{n(n+1)}{2} + \frac{5n(n-1)}{2} = 3n^2 + 2n.$$

Theorem 4.3. *Let GA be the number of the garbage outputs produced by a n-bit reversible carry look-ahead adder. Then,* $GA = n^2 + n - 6$.

Proof : From Theorem 4.1 it is found that the proposed n-bit reversible carry look-ahead adder requires n numbers of PG gates. Each PG gate produces one garbage output. The circuit requires $\frac{n(n+1)}{2}$ FG gates. Only the last $n-1$ numbers of FG produces one garbage output each. The other FG gates do not produce any garbage output. So, garbage outputs produces by FG gates are $n-1$. The circuit has $\frac{n(n-1)}{2}$ numbers of TG gates. The TG gate produces total $n(n-1)-5$ garbage outputs. Therefore, total garbage outputs produced by n-bit reversible carry look-ahead adder is,

$$n + n - 1 + n(n-1) - 5 = n^2 + n - 6.$$

4.2.1.1 Simulation of the Proposed Reversible Carry Look-Ahead Adder

The proposed design of reversible carry look-ahead adder is simulated using Microwind DSCH 3.0 [38] software on a computer, which has Intel(R) Core(TM) i7-4790 CPU with 3.60 GHz Clock Speed and 4.00 GB RAM. The timing diagram, shown in Figure 4.11, verifies the correctness of the proposed reversible 4-bit carry look-ahead adder. Here, two 4-bit inputs are a and b and the output is sum. At time 800 ns, input $a = 5$ and $b = 7$. At that time, the output $sum = 12$. At time 1400 ns, input $a = 9$ and $b = 9$. At that time, the output $sum = 18$. From the timing diagram, it is found that the proposed circuit gives the correct result for each different set of inputs. So, the simulation result verifies the correctness of the circuit.

4.2.1.2 Performance Analysis of the Proposed Reversible Carry Look-Ahead Adder

Table 4.7 shows the comparative analysis of the proposed and existing 4-bit reversible carry look-ahead adders. From Table 4.7, it is found that the proposed

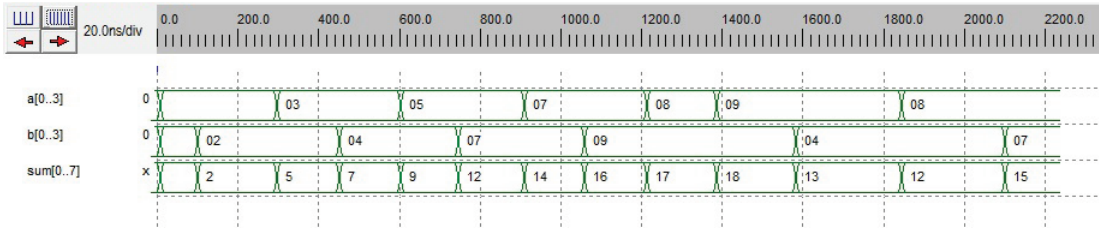

FIGURE 4.11: Simulation Result of Proposed Reversible 4-bit Carry Look-Ahead Adder

design based on reversible Peres, Toffoli and Feynman gates is much better than the existing ones [64], [65] in terms of number of gates, garbage outputs and quantum cost.

TABLE 4.7: Comparison of Different 4-bit Reversible Carry Look-Ahead Adders

4-bit Carry Look-Ahead Adder	No. of Gates	Garbage Outputs	Quantum Cost
Proposed Method	20	10	56
Existing Design [64]	22	19	99
Existing Design [65]	25	11	88
Improvement(%) w.r.t [64]	9.09	47.37	43.43
Improvement(%) w.r.t [65]	20	9.09	36.36

4.2.2 Proposed Reversible Divider

The divider circuit performs division operation between two operands. Let, A, D, Q and R be the dividend, divisor, quotient and remainder respectively. Then $A = Q \times D + R$. In every step i, the divisor D is shifted i bits to the right. The right-shifted divisor is subtracted from or added to the dividend and produces partial remainder. The sign of the partial remainder determines the quotient bit. In non-restoring division, this sign determines whether to add or subtract the shifted divisor in the next step.

Here, two approaches have been used to construct the divider. In the first approach, conventional division array has been used and in the second approach high speed division array has been used [66], [67], [68], [69].

4.2.2.1 Proposed Reversible Divider using Conventional Division Array

Let, Dividend, $A = A_0.A_1A_2...A_n$; Divisor, $D = D_0.D_1D_2...D_n$; Remainder, $R = R_0.R_1R_2...R_n$; Quotient, $Q = Q_0.Q_1Q_2...Q_n$.

The operands are assumed to be positive, normalized fractions. So, $A_0 = D_0 = 0$ and $A_1 = D_1 = 1$. The quotient is positive and the partial remainder R is a signed fraction, and R_0 is the sign bit with R being represented in 2's complement form. To perform the n-bit parallel non-restoring division, an $(n+1)$ by $(n+1)$ non-restoring array divider is required [66], [67], [68], [69].

The non-restoring array divider consists of rows of carry-propagate adders with each logic cell containing a Double Peres gate and two Feynman gates. The divisor shifts right bit by bit. The second Feynman gate controls the divisor input to the full adder. The control signal P determines whether an addition or subtraction is to be performed. Subtraction is performed in 2's complement form by forming the 1's complement of the divisor and forcing a carry into the rightmost cell. Figure 4.12 shows the reversible cell of conventional division array. This cell and Feynman gates have been used to construct the entire division array. Figure 4.13 shows the proposed reversible conventional division array for 8-bit dividend and 4-bit divisor.

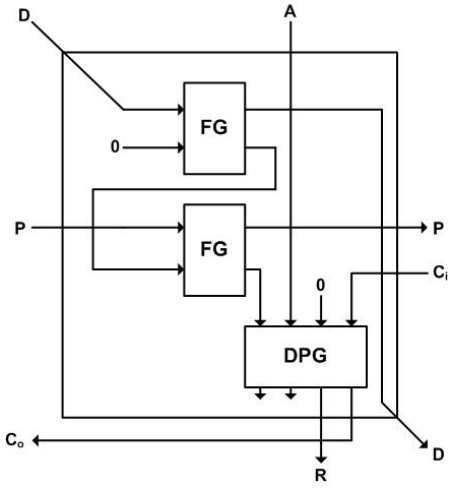

FIGURE 4.12: Reversible Cell of Proposed Conventional Division Array

Theorem 4.4. *The n-bit reversible divider using conventional division array can be realized by at least* $\frac{3(n+2)^2+2n}{4}$ *gates, where n is the number of bits of the dividend.*

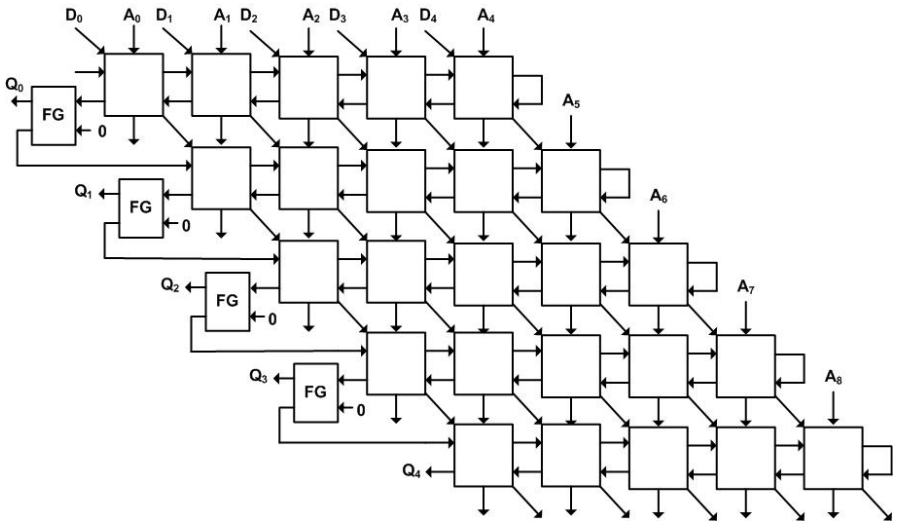

FIGURE 4.13: Proposed Reversible Conventional Division Array: 8-bit Dividend and 4-bit Divisor

Proof: As an n-bit reversible divider requires $(\frac{n}{2}+1)(\frac{n}{2}+1)$ conventional cells and each cell is constructed with two FGs and one DPG, the total number of gates in all cells is $3(\frac{n}{2}+1)(\frac{n}{2}+1)$. Beside this, $\frac{n}{2}$ extra FGs are required to shift the carry out. So, total number of gates required for the divider is

$$3(\tfrac{n}{2}+1)(\tfrac{n}{2}+1)+\tfrac{n}{2}=\tfrac{3(n+2)^2+2n}{4}.$$

Theorem 4.5. *The n-bit reversible divider using conventional division array produces at least $\frac{(n+2)^2}{2}$ garbage outputs.*

Proof: The n-bit reversible divider requires $(\frac{n}{2}+1)(\frac{n}{2}+1)$ conventional cells, where each cell produces two garbage outputs. The extra Feynman gates do not produce any garbage outputs. So, total number of garbage outputs produced by the divider is

$$2(\tfrac{n}{2}+1)(\tfrac{n}{2}+1)=\tfrac{(n+2)^2}{2}.$$

Theorem 4.6. *The n-bit reversible divider using conventional division array requires at least $\frac{4(n+2)^2+n}{2}$ quantum costs.*

Proof: The n-bit reversible divider requires $(\frac{n}{2}+1)(\frac{n}{2}+1)$ conventional cells. Each cell is constructed with two FGs and one DPG. The quantum costs of FG [35] and

DPG [46] are one and six, respectively. Moreover, the division array requires $\frac{n}{2}$ extra FGs to shift the carry out. So, the total quantum cost of the divider is

$$8(\tfrac{n}{2}+1)(\tfrac{n}{2}+1)+\tfrac{n}{2} = \tfrac{4(n+2)^2+n}{2}.$$

4.2.2.2 Proposed Reversible Divider using High Speed Division Array

The reversible divider using high speed division array is designed by some major modifications of the design described in the previous section. The carry ripple time, which is proportional to n, has been omitted in this design. The partial remainder R is not developed in each row of the array, but is represented by two binary vectors S and C which, if added, would produce the correct partial remainder at that row level. A single carry-lookahead circuit is used to determine from the S and C vectors what the carry sign would be, facilitating the determination of the sign of the resulting partial remainder, the quotient bit for that row level, and the control (add or subtract divisor) to the next row. Subtraction of the divisor is implemented using 2's complement addition as in the conventional array [66], [69].

Three types of cell, namely, A cell, S cell and CLA cell have been used in the high speed array design. The A_js (dividend) are input from the top, and the complemented D_js (divisor) are input through the diagonal lines. Addition or subtraction is to be performed in the A cells according to the same concept of the previous section. S^i_j is the sum bit generated in the jth bit position of ith row, and C^i_j and C^i_{j-1} are the carry-in and carry-out of that position, respectively. The two outputs resulting from the upper row are input to the row below. Representing them in vectors:

$$\begin{array}{lllllll}
S^{i-1} = S^{i-1}_0. & S^{i-1}_1 & S^{i-1}_2 & \ldots & S^{i-1}_{n-1} & A_{n+i} \\
C^{i-1} = C^{i-1}_0. & C^{i-1}_1 & C^{i-1}_2 & \ldots & 0 & Q_{i-1} \\
\pm D = D_0. & D_1 & D_2 & \ldots & D_{n-1} & D_n \\
\hline
S^i = S^i_0. & S^i_1 & S^i_2 & \ldots & S^i_{n-1} & S^i_n \\
C^i = C^i_0. & C^i_1 & C^i_2 & \ldots & C^i_{n-1} & 0
\end{array}$$

The A cell implements the following operations:

$$\begin{aligned} S_j^i &= S_j^{i-1} \oplus C_j^{i-1} \oplus (D_j \oplus K^i) \\ C_{j-1}^i &= (D_j \oplus K_i)(S_j^{i-1} + C_j^{i-1}) + S_j^{i-1} C_j^{i-1} \\ G_j^i &= C_j^i S_j^i \\ P_j^i &= C_j^i \oplus S_j^i \end{aligned}$$

The S cell implements the following operation:

$$S_0^i = S_0^{i-1} \oplus C_0^{i-1} \oplus \bar{K}^i \oplus C_0^i$$

The CLA cell implements the following operation:

$$C^i = G_1^i + P_1^i G_2^i + P_1^i P_2^i G_3^i + ... + P_1^i P_2^i ... P_{n-2}^i G_{n-1}^i$$

$\bar{K}^i$ (where $\bar{K}^i$ is Q_{j-1}) is a control signal propagating from left to right in the ith row without any delay. Since the carry is saved, not waiting time is required to propagate from the rightmost cell to the leftmost cell. So, the operation carried here is independent of the word length n.

Figure 4.14(a), Figure 4.14(b) and Figure 4.14(c) show the diagrams of A cell, S cell and CLA cell, respectively. Figure 4.15 shows the proposed reversible high speed division array for 8-bit dividend and 4-bit divisor.

Theorem 4.7. *The n-bit reversible divider using high speed division array can be realized by at least* $\frac{(n+2)(3n+11)}{2}$ *gates, where n is the number of bits of the dividend.*

Proof: An n-bit reversible division array requires $\frac{n}{2}(\frac{n}{2}+1)$ A cells, $\frac{n}{2}+1$ S cells and $\frac{n}{2}+1$ CLA cells.

Figures 4.14(a), 4.14(b) and 4.14(c) show that an A cell, S cell and CLA cell consist of six, four and five reversible gates, respectively. Moreover, the divider array requires additional $n+2$ numbers of FGs.

So, total number of reversible gates required in the high speed divider is

TABLE 4.8: Comparison Between Proposed and Existing 2-bit, 4-bit, 8-bit and 16-bit Dividers

Circuit	2-bit divider				4-bit divider				8-bit divider				16-bit divider				Functional Feature
	GC	GO	QC	DL	GC	GO	QC	DL	GC	GO	QC	DL	GC	GO	QC	DL	
Proposed Conventional	13	8	33	24	33	18	74	54	79	50	204	150	251	162	656	486	Positive fractions
Proposed High Speed	34	28	82	12	69	51	165	27	175	115	415	85	531	315	1251	297	Positive fractions
Existing [70]	53	40	172	-	89	62	294	-	161	106	538	-	305	194	1026	-	Positive integers
Existing [71]	-	42	210	-	-	66	360	-	-	114	660	-	-	210	1260	-	Positive integers
Existing [72]	-	61	210	-	-	101	422	-	-	181	846	-	-	341	1694	-	Signed integers

Here, GC = Gate count, GO = Garbage output, QC = Quantum cost, DL = Delay and '-' means not available

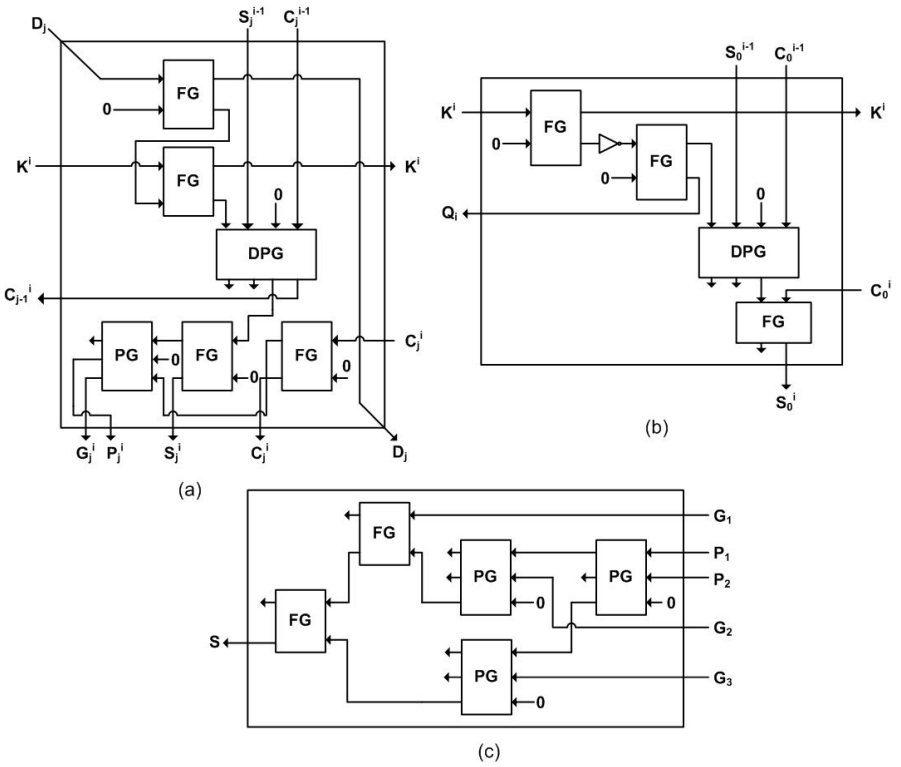

FIGURE 4.14: (a) A Cell; (b) S Cell; and (c) CLA Cell of Proposed High Speed Division Array

$$6\tfrac{n}{2}(\tfrac{n}{2}+1)+4(\tfrac{n}{2}+1)+5(\tfrac{n}{2}+1)+n+2=\tfrac{(n+2)(3n+11)}{2}.$$

Theorem 4.8. *The n-bit reversible divider using high speed division array generates at least* $\frac{(n+2)(3n+22)}{4}$ *garbage outputs.*

Proof: Figures 4.14(a), 4.14(b) and 4.14(c) show that each A cell, S cell and CLA cell of the division array produce three, three and seven garbage outputs, respectively. The circuit has additional $n+2$ numbers of FGs. The last $\frac{n+2}{2}$ numbers of FGs do not produce any garbage outputs. The other $\frac{n+2}{2}$ numbers of FGs produce one garbage output each.

$\frac{n}{2}(\frac{n}{2}+1)$ A cells, $\frac{n}{2}+1$ S cells and $\frac{n}{2}+1$ CLA cells are required to construct the n-bit reversible division array.

So, total number of garbage outputs produced by the high speed divider is

$$3\tfrac{n}{2}(\tfrac{n}{2}+1)+3(\tfrac{n}{2}+1)+7(\tfrac{n}{2}+1)+\tfrac{n+2}{2}=\tfrac{(n+2)(3n+22)}{4}.$$

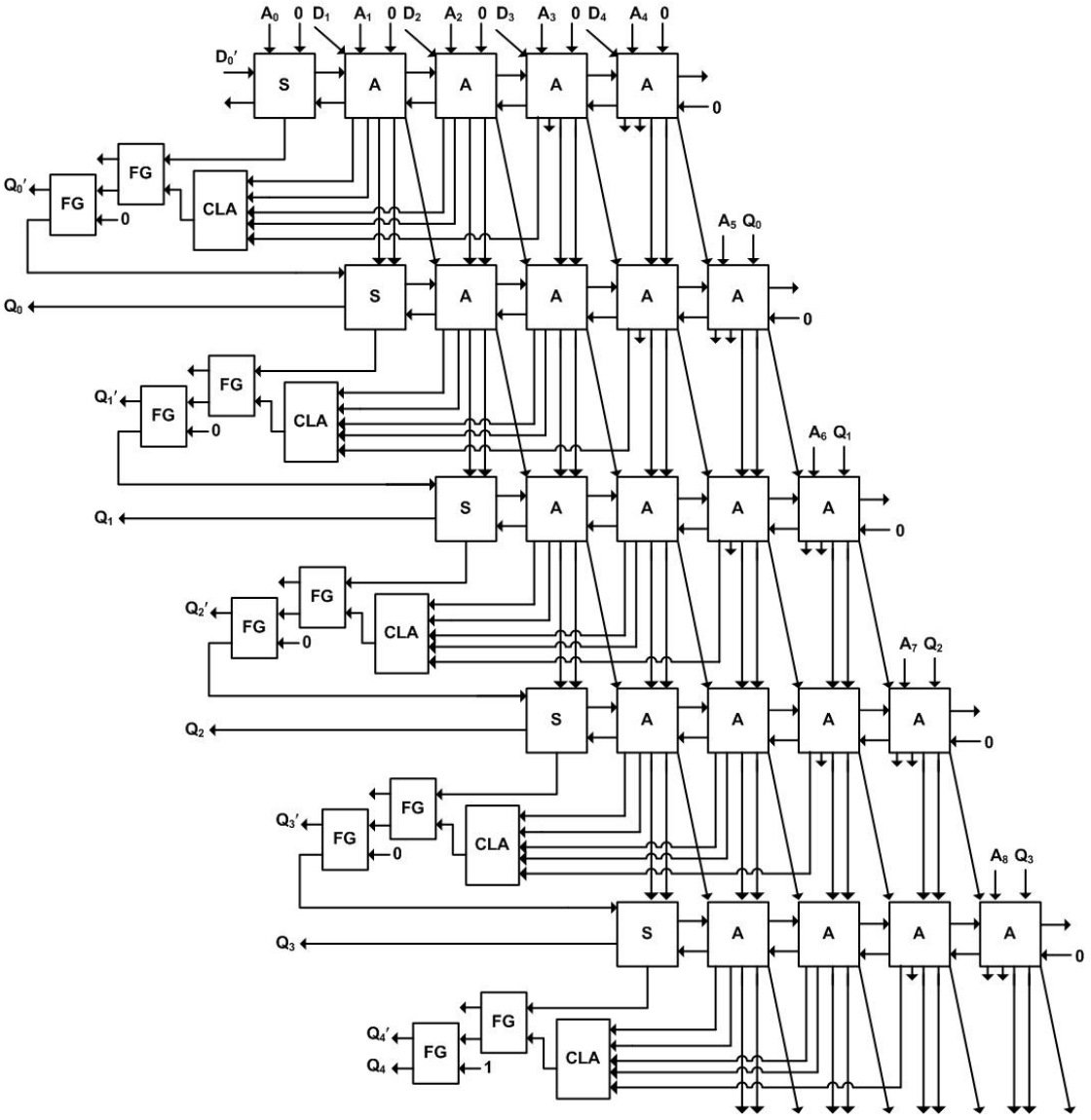

FIGURE 4.15: Proposed Reversible High Speed Division Array

Theorem 4.9. *The n-bit reversible divider using high speed division array requires at least* $\frac{(n+2)(7n+27)}{2}$ *quantum costs.*

Proof: The quantum costs of each FG, PG and DPG are one, four and six, respectively [35], [45], [46]. An A cell requires four FGs, one PG and one DPG; S cell requires three FGs and one DPG; CLA cell requires two FGs and three PGs. Additional $n + 2$ numbers of FGs are required to construct the circuit.

So, the total quantum costs produced by the high speed divider is

$$(4 + 1 \times 4 + 1 \times 6)\tfrac{n}{2}(\tfrac{n}{2} + 1) + (3 \times 1 + 6)(\tfrac{n}{2} + 1) + (2 \times 1 + 3 \times 4)(\tfrac{n}{2} + 1) + n + 2$$

$$= \frac{(n+2)(7n+27)}{2}.$$

4.2.2.3 Performance Analysis of the Proposed Reversible Divider

In this section, the performance of the proposed dividers with the existing designs [70], [71], [72] has been analyzed.

Table 4.8 shows the comparative result between the proposed and the existing 8-bit dividers.

From the table it is found that the delay of high speed array divider is much less than the conventional one as the carry is saved and no waiting time is required to propagate bits from the rightmost cell to leftmost cell.

4.2.3 Proposed Reversible Comparator

In this section, a compact and improved version of reversible n-bit comparator has been proposed. A binary comparator compares the magnitude of two binary numbers to determine whether they are equal or one is greater/less than the other [54]. To construct the optimized n-bit comparator, two new reversible gates, namely BJSS and HLNS gate, have been proposed. In order to improve the design, an MSB (Most Significant Bit) comparator circuit has been proposed for comparing the nth bit (MSB) of two n-bit numbers. Then, a single-bit GE (greater or equal) comparator cell has been designed to generate greater and equal signal for the remaining $(n-1)$ bits of two numbers with the previous comparison result of MSB. Moreover, a single-bit LT (less than) comparator cell is designed to determine the less than signal. These three circuits are then extensively used to design 2-bit and n-bit reversible binary comparators.

To construct the optimized reversible n-bit comparator, a new reversible gate, BJSS gate, has been proposed. The proposed BJSS gate is used to design the 1-bit comparator and the MSB comparator. The proposed BJSS gate is shown in Figure 4.16. The corresponding truth table of the gate is shown in Table 4.9. It can be verified from the truth table that the input pattern corresponding to a particular output pattern can be uniquely determined and vice versa. Quantum

realization of the proposed BJSS gate is shown in Figure 4.17. The quantum cost of the proposed BJSS gate is six.

The proposed BJSS gate can implement all Boolean functions. When $C = 0$ and $D = 1$, the BJSS gate simultaneously implements $Q = A + B$, $R = AB$ and $S = A \oplus B$. When $B = 1$ and $C = 1$, the output is $Q = A'$.

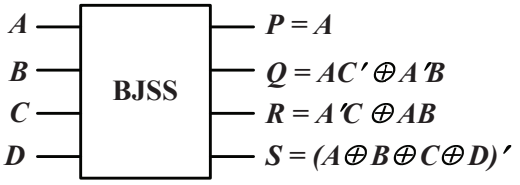

FIGURE 4.16: 4×4 Reversible BJSS Gate

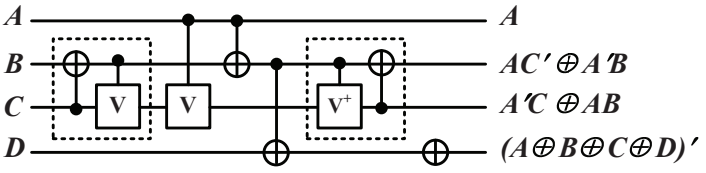

FIGURE 4.17: Quantum Realization of BJSS Gate

TABLE 4.9: Truth Table of 4×4 BJSS gate

INPUT				OUTPUT			
A	B	C	D	P	Q	R	S
0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0
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	1	1
0	1	1	0	0	1	1	1
0	1	1	1	0	1	1	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	0	0	1
1	0	1	1	1	0	0	0
1	1	0	0	1	1	1	1
1	1	0	1	1	1	1	0
1	1	1	0	1	0	1	0
1	1	1	1	1	0	1	1

TABLE 4.10: Truth Table of 3×3 HLNS gate

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

A 3 ×3 reversible gate is also proposed which is named as HLNS gate to construct the proposed comparator. The proposed gate is shown in Figure 4.18. The corresponding truth table of the gate is shown in Table 4.10. The truth table shows that the input-output combinations preserve the one-to-one mapping between them. Quantum realization of the proposed HLNS gate is shown in Figure 4.19. The quantum cost of the BJSS gate is five.

The HLNS gate can implement all Boolean functions. When $C = 1$, it implements EX-OR ($Q = A \oplus B$) and when $C = 0$, it implements EX-NOR ($Q = (A \oplus B)'$ and ($Q = AB$) functions. When $B = 1$, it implements OR function ($R = A + B$). When $B = 0$ and $C = 0$, it depicts NOT function ($Q = A'$). The proposed HLNS gate is used to design the single-bit greater or equal comparator cell.

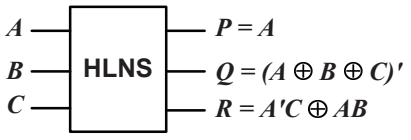

FIGURE 4.18: 3×3 Reversible HLNS Gate

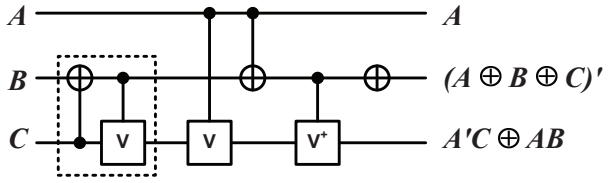

FIGURE 4.19: Quantum Realization of HLNS Gate

4.2.3.1 Proposed Design of 1-bit Reversible Comparator Circuit

The 1-bit comparator compares two 1-bit binary numbers and determines the result among $ALB(A < B)$, $AEB(A = B)$ and $AGB(A > B)$ [54]. The truth table of 1-bit comparator is shown in 4.11. From the truth table, it is obvious that $ALB = A'B$, $AEB = (A \oplus B)'$ and $AGB = AB'$. The proposed BJSS gate can be used as a 1-bit comparator. The gate takes two 1-bit binary numbers as inputs A and B and two constant inputs. The outputs of the gate produce one garbage output and three consecutive outputs AGB, ALB and AEB (shown in Figure 4.20). The quantum cost of the proposed 1-bit reversible comparator is six as it requires only one BJSS gate. It produces only one garbage output.

TABLE 4.11: Truth Table for 1-bit Binary Comparator

INPUT		OUTPUT		
A	B	ALB	AEB	AGB
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

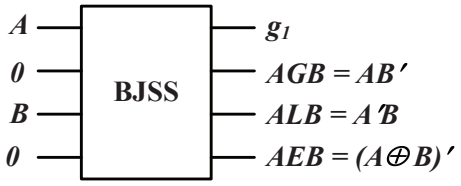

FIGURE 4.20: 1-bit Reversible Comparator

4.2.3.2 Proposed Design of Reversible MSB Comparator Circuit

The MSB comparator circuit takes MSB of two binary numbers A_n, B_n and sets two constant inputs as 0. It produces three outputs based on the bits present at the input level. $ALB(R_n)$, $AGB(Q_n)$ and $AEB(P_n)$ are the three outputs produced from this circuit. Later on these outputs are fed into the next level, where $(n-1)$th bits have also been compared. This proposed MSB comparator circuit shown in Figure 4.21 consists of only one BJSS gate. So, quantum cost of the circuit is six. It produces only one garbage output.

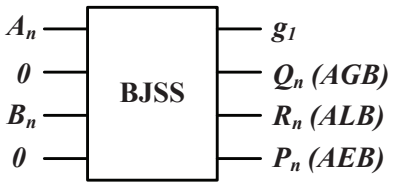

FIGURE 4.21: MSB Comparator Circuit

4.2.3.3 Proposed Design of Reversible Single-bit Greater or Equal Comparator Cell

The proposed reversible single-bit comparator cell consists of one HLNS gate and two Peres Gates. It takes $(n-1)$th bits of two binary numbers A and B and two more inputs P_n and Q_n from previous comparison result. Together they work to produce two outputs AEB (P_{n-1}) and AGB (Q_{n-1}) which indicates whether the given numbers A and B are equal to each other or greater than the other. Figs. 4.22 and 4.23 show the proposed circuit and the block diagram of the reversible single-bit greater or equal comparator cell, respectively. The proposed single-bit GE comparator cell requires two PG and one HLNS gates. The quantum cost of PG and HLNS are four and five, respectively. So, total quantum cost of this circuit is thirteen. It produces four garbage outputs.

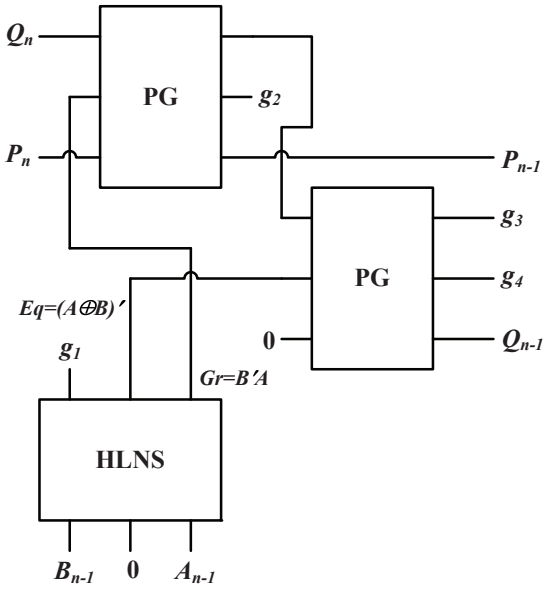

FIGURE 4.22: Proposed Design of Single-bit GE Comparator Cell

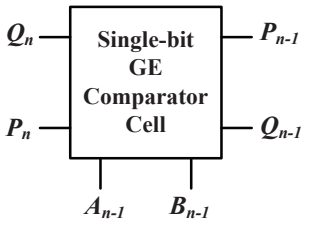

FIGURE 4.23: Block Diagram of Single-bit GE Comparator Cell

4.2.3.4 Proposed Design of Reversible Single-bit Less Than Comparator Cell

The 1-bit comparator circuit and MSB comparator circuit described above produce two outputs P_{n-1} (AEB) and Q_{n-1} (AGB). If $A_{n-1} < B_{n-1}$, It can be calculated using the equation, $ALB = (AEB \oplus AGB)'$. The design of the proposed reversible single-bit less than (LT) comparator circuit and its block diagram are shown in Figs. 4.24 and 4.25, respectively. The circuit requires only two FGs. So, the quantum cost of the circuit is two.

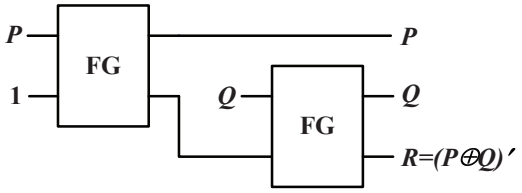

FIGURE 4.24: Proposed Design of Single-bit LT Comparator Cell

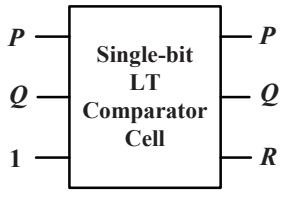

FIGURE 4.25: Block Diagram of Single-bit LT Comparator Cell

4.2.3.5 Proposed Design of Reversible 2-bit Comparator

A reversible 2-bit binary comparator consists of a proposed reversible MSB comparator, a single-bit greater or equal (GE) comparator and a single-bit LT comparator circuit. The reversible design of 2-bit binary comparator is shown in Figure

4.26. The quantum cost of MSB comparator, single-bit GE comparator and single-bit LT comparator are six, thirteen and two, respectively. So, the quantum cost of the reversible 2-bit comparator circuit is twenty one.

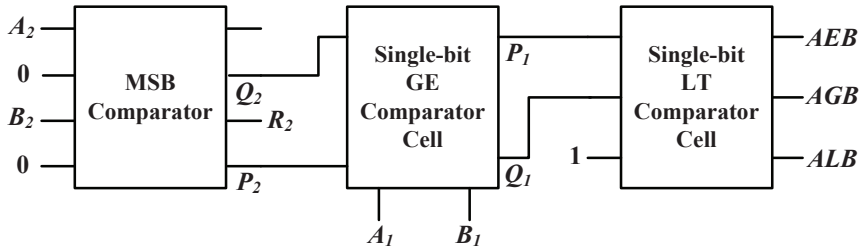

FIGURE 4.26: Proposed Design of Reversible 2-bit Comparator

4.2.3.6 Proposed Design of Reversible n-bit Comparator

The proposed reversible n-bit binary comparator is shown in Figure 4.27 which consists of one MSB comparator, $(n-1)$ single-bit GE comparators and a single-bit LT comparator circuit. The algorithm for constructing reversible n-bit binary comparator is given in Algorithm 2

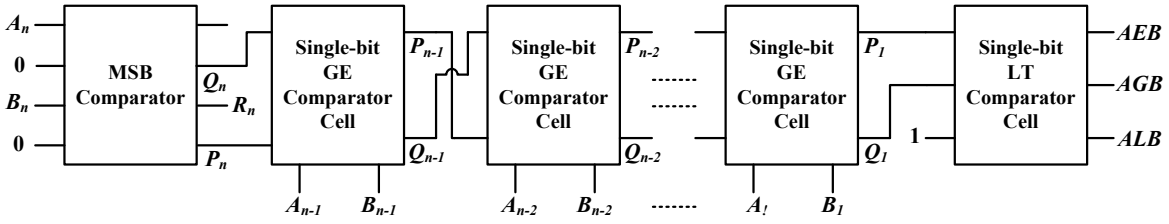

FIGURE 4.27: Proposed Design of Reversible n-bit Comparator

Theorem 4.10. *A reversible n-bit binary comparator $(n \geq 2)$ can be realized with $3n$ gates; where n is the number of data bits.*

Proof: The above statement is proved by mathematical induction.

An MSB comparator circuit requires one BJSS gate. A single-bit GE comparator cell is constructed with one HLNS gate and two Pers Gates. A single-bit LT comparator cell requires two Feynman Gates. A 2-bit comparator circuit is constructed with one MSB comparator circuit, one single-bit GE comparator cell and one single-bit LT comparator cell (shown in Figure 4.26). Hence, the total number

Algorithm 2 Reversible n-bit Binary Comparator

Take one MSB Comparator Circuit I_n. Outputs of this block are considered to be of Level L_n
$In[I_1] = A_n$
$In[I_2] = 0$
$In[I_3] = B_n$
$In[I_4] = 0$
if $(In[I_1] < In[I_2])$ **then**
  $In[O_3] = R_{Ln} = 1$
else if $(In[I_1] > In[I_2])$ **then**
  $In[O_2] = Q_{Ln} = 1$
else$In[O_4] = P_{Ln} = 1$
end if
For each single-bit comparator circuit, level of inputs and outputs are considered to be of L_n and L_{n-1}, respectively.
for $j = n - 1 \rightarrow 1$
  Take one single-bit GE comparator cell C_j
  if $(j = n - 1)$ **then**
    $C_j[I_1] = In[O_2] = Q_{Ln}$
    $C_j[I_2] = In[O_4] = P_{Ln}$
    $C_j[I_3] = A_{n-1}$
    $C_j[I_4] = B_{n-1}$
  else
    $C_j[I_1] = C_{j-1}[0_2] = Q_{Ln-1}$
    $C_j[I_2] = C_{j-1}[0_1] = P_{Ln-1}$
    $C_j[I_3] = A_j, C_j[I_4] = B_j$
  end if
end for
Take one single-bit LT comparator cell, L
$L[I_1] = C_1[O_1]$
$L[I_2] = C_1[O_2]$
$L[I_3] = 1$
$L[O_1] = C_1[O_1]$
$L[O_2] = C_1[O_2]$
$L[O_3] = C_1[O_1] \oplus C_1[O_2] \oplus 1$

of gates required to construct 2-bit binary comparator is (NOG_{2-bit}) is

$NOG_{2-bit} = NOG_{MSB} + +NOG_{SB-GE} + NOG_{SB-LT}$

$= 1 + (1 + 2) + 2$

$= 6$

$= 3 \times 2$

Hence, the statement holds for the base case $n = 2$.

Assume that, the statement holds for $n = k$. Hence, a reversible k-bit binary comparator can be realized with $3k$ reversible gates. A $(k+1)$-bit binary comparator requires one MSB comparator circuit, k single-bit GE comparator cell and one single-bit LT comparator cell. Hence, the total number of gates required to construct $(k+1)$-bit binary comparator ($NOG_{(k+1)-bit}$) is

$NOG_{(k+1)-bit} = NOG_{MSB} + k \times NOG_{SB-GE} + NOG_{SB-LT}$

$= 1 + k(1+2) + 2$

$= 3k + 3$

$= 3(k+1)$

Thus, the statement holds for $n = k + 1$.

Therefore a reversible n-bit binary comparator can be realized with $3n$ reversible gates.

Theorem 4.11. *A reversible n-bit binary comparator ($n \geq 2$) produces at least $4n - 3$ garbage outputs, where n is the number of data bits.*

Proof: A 2-bit comparator circuit consists of one MSB comparator, one single-bit GE comparator cell and one single-bit LT comparator cell. The MSB comparator circuit produces one garbage output and the single-bit GE comparator cell produces four garbage outputs. The single-bit LT comparator cell does not produce any garbage output. Hence, the total number of garbage outputs produced by a 2-bit comparator is at least

$G_{2-bit} = G_{MSB} + G_{SB-GE} + G_{SB-LT}$

$= 1 + 4 + 0$

$= 5$

$= 4 \times 2 - 3$

Hence, the statement holds for the base case $n = 2$.

Assume that, the statement holds for $n = k$. Hence, a reversible k-bit binary comparator produces at least $4k - 3$ garbage outputs.

A $(k+1)$-bit binary comparator requires one MSB comparator circuit, k single-bit GE comparator cell and one single-bit LT comparator cell. So, the total number

of garbage outputs produced by the $(k+1)$-bit binary comparator ($G_{(k+1)-bit}$) is
$G_{(k+1)-bit} = G_{MSB} + k \times G_{SB-GE} + G_{SB-LT}$
$= 1 + k \times 4 + 0$
$= 4k + 1$
$= 4(k+1) - 3$

Thus, the statement holds for $n = k + 1$.

Therefore a reversible n-bit binary comparator produces at least $4n - 3$ garbage outputs.

Theorem 4.12. *A reversible n-bit binary comparator* $(n \geq 2)$ *can be realized with* $13n - 5$ *quantum cost; where n is the number of data bits.*

Proof: An MSB comparator circuit requires one BJSS gate. A single-bit GE comparator cell is constructed with one HLNS gate and two Peres Gates. A single-bit LT comparator cell requires two Feynman Gates. A 2-bit comparator circuit is constructed with one MSB comparator circuit, one single-bit GE comparator cell and one single-bit LT comparator cell. The quantum cost of BJSS gate, HLNS gate, PG and FG are six, five, four and one, respectively. Hence, the total quantum cost of the 2-bit comparator circuit is
$QC_{2-bit} = QC_{MSB} + QC_{SB-GE} + QC_{SB-LT}$
$= 6 + (5 + 4 \times 2) + 2$
$= 21$
$= 13 \times 2 - 5$

Hence, the statement holds for the base case $n = 2$.

Assume that, the statement holds for $n = k$. Hence, a reversible k-bit binary comparator can be realized with $13k - 5$ quantum cost. A $(k+1)$-bit binary comparator requires one MSB comparator circuit, k single-bit GE comparator cell and one single-bit LT comparator cell. Hence, the quantum cost of the $(k+1)$-bit binary comparator is ($QC_{(k+1)-bit}$) is
$QC_{(k+1)-bit} = QC_{MSB} + k \times QC_{SB-GE} + QC_{SB-LT}$
$= 6 + k(5 + 4 \times 2) + 2$

$= 13k + 8$
$= 13(k + 1) - 5$

Thus, the statement holds for $n = k + 1$.

Therefore a reversible n-bit binary comparator can be realized with 13n - 5 quantum cost.

Theorem 4.13. *A reversible n-bit binary comparator* $(n \geq 2)$ *requires at least three ancilla inputs, where n is the number of data bits.*

Proof: An n-bit comparator circuit $(n \geq 2)$ consists of one MSB comparator, $(n-1)$ single-bit GE comparator cells and one single-bit LT comparator cell. The MSB comparator circuit requires two ancilla inputs and the single-bit LT comparator cell requires one ancilla input. The $(n-1)$ single-bit GE comparator cells do not require any ancilla input. Hence, the total number of ancilla inputs required for an n-bit comparator is at least
$AI = AI_{MSB} + (n-1) \times AI_{SB-GE} + AI_{SB-LT}$
$= 2 + (n-1) \times 0 + 1$
$= 3$

Theorem 4.14. *A reversible n-bit comparator* $(n \geq 2)$ *requires* $(0.15n - 0.03)ns$ *timing delay, where n is the number of data bits.*

Proof: The proposed n-bit comparator design has a serial architecture that has latency of $O(n)$. An n-bit comparator requires one MSB comparator circuit, $(n-1)$ GE comparator cells and one single-bit LT comparator cell. Hence, the delay of the comparator circuit is

$$T = T_{MSB} + (n-1) \times T_{SB-GE} + T_{SB-LT} \tag{4.9}$$

The critical path for single-bit GE comparator cell contains one HLNS gate and two Peres Gates. The delays of HLNS gate and PG are $0.07ns$ and $0.04ns$, respectively, (obtained by DSCH 3.5 [38]. Hence, the delay of single-bit GE comparator cell can be calculated as follows:

$$T_{SB-GE} = T_{HLNS} + 2T_{PG} = (0.07 + 2 \times 0.04)ns = 0.15ns$$

Using DSCH 3.5 [38], CMOS 45 nm Open Cell Library [37] and interconnect delay [73], the delays of the MSB and LT modules are obtained which are 0.07 and $0.05ns$, respectively. Using Eq. 4.9, the total delay of an n-bit comparator is
$T = 0.07 + (n-1) \times 0.15 + 0.05ns$
$= (0.15n - 0.15 + 0.12)ns$
$= (0.15n - 0.03)ns$

Theorem 4.15. *A reversible n-bit comparator ($n \geq 2$) requires $(42.5n - 14.5)\mu m^2$ area, where n is the number of data bits.*

Proof: An n-bit comparator is constructed with one MSB comparator circuit, $(n-1)$ single-bit GE comparator cells and one single-bit LT comparator cell. An MSB comparator circuit requires one BJSS gate. A single-bit GE comparator cell is constructed with one HLNS gate and two Peres Gates. A single-bit LT comparator cell requires two Feynman Gates. Hence, the total number of gates is

$$NOG = NOG_{MSB} + (n-1) \times NOG_{SB-GE} + NOG_{SB-LT} \tag{4.10}$$

$$= 1 \times BJSS + (n-1)(HLNS + 2 \times PG) + 2 \times FG$$

The areas of BJSS gate, HLNS gate, PG and FG are 20, 17.5, 12.5 and $7\mu m^2$, respectively, (obtained using CMOS $45nm$ Open Cell Library [37] and Algorithm 3.

Hence, the total area of an n-bit comparator os $A = A_{MSB} + (n-1) \times A_{SB-GE} + A_{SB-LT}$
$= 20 + (n-1)(17.5 + 2 \times 12.5) + 7\mu m^2$
$= (42.5n - 42.5 + 27)\mu m^2$
$= (42.5n - 14.5)\mu m^2$

Algorithm 3 Area Calculation

Take one MSB Comparator Circuit I_n
$Area = area(I_n)$
for $j = n-1 \rightarrow 1$
  Take one single-bit GE comparator cell C_j
  $Area = Area + area(C_j)$
end for
Take one single-bit LT comparator cell, L
$Area = Area + area(L)$

Theorem 4.16. *A reversible n-bit comparator* $(n \geq 2)$ *requires* $(117.76n-32.94)\mu W$ *power, where n is the number of data bits.*

Proof: One MSB comparator circuit, $(n-1)$ single-bit GE comparator cells and one single-bit LT comparator cell are required to realize an n-bit comparator. Hence, the total required power can be measured by using following equation:

$$P = P_{MSB} + (n-1) \times P_{SB-GE} + P_{SB-LT} \tag{4.11}$$

The single-bit GE comparator cell contains one HLNS gate and two Peres Gates. The power requirement of HLNS gate and PG are 58.88 and $29.44\mu W$, respectively, (obtained by DSCH 3.5 [38] and CMOS 45 nm Open Cell Library [37]). Hence, the power requirement of a single-bit GE comparator cell, P_{SB-GE}, can be calculated as follows:

$$P_{SB-GE} = (58.88 + 2 \times 29.44)\mu W = 117.76\mu W$$

Using DSCH 3.5 [38], CMOS 45 nm Open Cell Library [37] and Algorithm 4, the power requirement of MSB and LT modules have been obtained as 37.72 and $47.1\mu W$, respectively. Hence, the total power of an n-bit comparator can be modelled as below according to Eq. 4.11:

$P = 37.72 + (n-1) \times 117.76 + 47.1\mu W$
$= (117.76n - 117.76 + 84.82)\mu W$
$= (117.76n - 32.94)\mu W$

Algorithm 4 Power Calculation

Take one MSB Comparator Circuit I_n
$Power = power(I_n)$
for $j = n - 1 \rightarrow 1$
  Take one single-bit GE comparator cell C_j
  $Power = Power + power(C_j)$
end for
Take one single-bit LT comparator cell, L
$Power = Power + power(L)$

4.2.3.7 Performance Analysis of the Proposed Reversible Comparator

The simulation of the proposed circuit is done using Microwind DSCH 3.5 [38] and CMOS 45 nm Open Cell Library [37] software on a computer, which has the Intel(R) Core(TM) i7-4790 CPU with 3.60 GHz Clock Speed and 4.00 GB RAM. The simulation results show that the comparators give correct outputs for all possible combinations of inputs. The simulation results for 1-bit, 2-bit and 4-bit reversible binary comparators are shown in Figures 4.28, 4.29 and 4.30, respectively, which ensure the correctness of the functioning of the proposed comparator. For example, from Figure 4.28 it is found that at time 10 ns, $A = 0$, $B = 0$. At that time the output $AEB = 1$, $ALB = 0$ and $AGB = 0$. Similarly, at time 100 ns, $A = 0$, $B = 1$. At that time the output $AEB = 0$, $ALB = 1$ and $AGB = 0$. From Figure 4.29, it is found that at time 100 ns, $A_1A_0 = 00$, $B_1B_0 = 10$. At that time the output $AEB = 0$, $ALB = 1$ and $AGB = 0$. From Figure 4.30, it is found that at time 800 ns, $A_3A_2A_1A_0 = 1010$, $B_3B_2B_1B_0 = 0100$. At that time the output $AEB = 0$, $ALB = 0$ and $AGB = 1$.

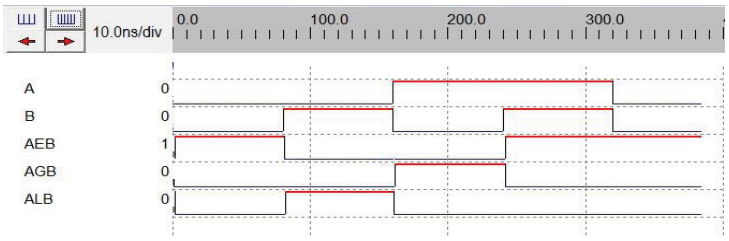

FIGURE 4.28: Simulation Result of Proposed Reversible 1-bit Comparator

Tables 4.12 and 4.13 show the comparative study of the proposed design with existing designs for a reversible n-bit comparator. Tables 4.14 and 4.15 show the

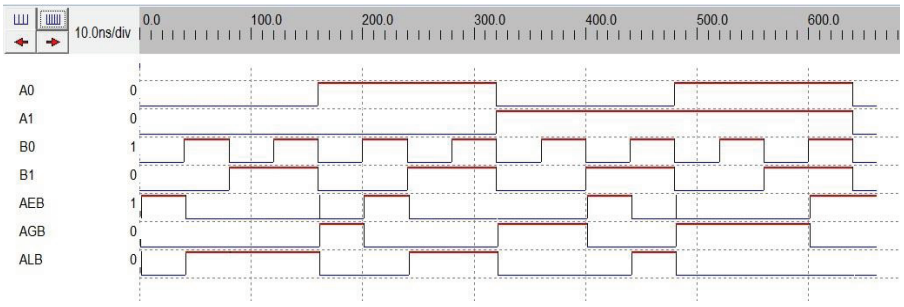

FIGURE 4.29: Simulation Result of Proposed Reversible 2-bit Comparator

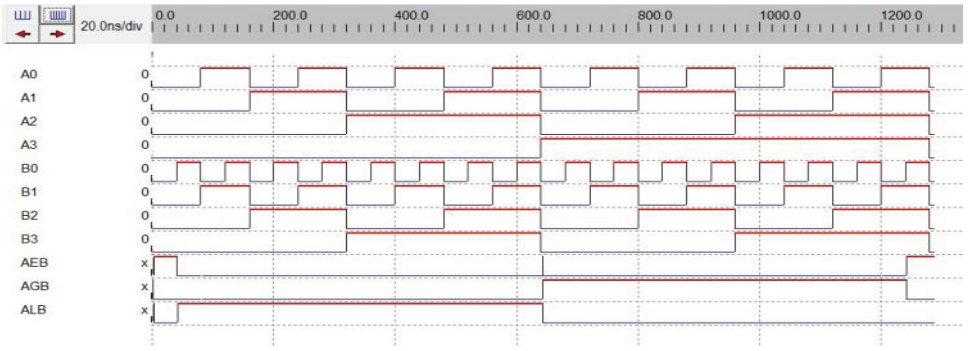

FIGURE 4.30: Simulation Result of Proposed Reversible 4-bit Comparator

comparative study of our proposed design with the existing designs for reversible 8-bit and 64-bit comparators, respectively. From Tables 4.14 and 4.15, it is clear that the proposed design outperforms the existing ones in terms of number of gates, garbage outputs, quantum costs, area, power and delay for 8-bit and 64-bit comparator circuits, respectively. For all of the cases, the proposed circuit requires less numbers of gates, garbage outputs and quantum costs than the existing designs [74], [75], [76], [77]. Although the proposed design has less delay than [74], it produces much delay than [75], [76]. As [75], [76] both are tree-based design, they require $O(log_2(n))$ delay, whereas the proposed design requires $O(n)$ delay. The proposed design outperforms the existing ones in terms of area and power also.

TABLE 4.12: Comparison of Different Reversible n-bit Comparators

Circuit	No. of Gates	Garbage Outputs	Quantum Cost
Proposed Method	$3n$	$4n-3$	$13n-5$
Existing Design [74]	$7n-4$	$5n-4$	$16n-10$
Existing Design [75]	$9n$	$6n-6$	$18n-10$
Existing Design [76]	$4n-2$	$5n-4$	$14n$
Existing Design [77]	–	$5n-1$	$17n-12$

TABLE 4.13: Area, Power and Delay Comparison of Different Reversible n-bit Comparators

Circuit	Area, μm^2	Power, μW	Delay, ns
Proposed Method	$42.5n - 12.5$	$117.76n - 32.94$	$0.15n - 0.03$
Existing Design [74]	$57.5n - 35$	$182.53n + 76.55$	$0.2n - 0.16$
Existing Design [75]	$90n - 77.5$	$268.23n - 239.2$	$0.23 * log_2 n + 0.1$
Existing Design [76]	$57.5n - 35$	$122.36n - 60.36$	$0.09 * log_2 n + 0.2$

TABLE 4.14: Comparison of Different Reversible 8-bit Comparators

Circuit	NOG	GO	QC	Area	Power	Delay
Proposed Method	24	29	99	325.5	909.14	1.17
Existing Design [74]	52	36	118	425	1536.79	1.44
Existing Design [75]	72	42	135	642.5	1906.64	0.79
Existing Design [76]	30	36	112	425	918.52	0.47
Improvement (%) with respect to [74]	53.8	19.4	16.1	23.41	40.84	18.75
Improvement (%) with respect to [75]	66.6	30.9	26.7	49.33	52.32	NI
Improvement (%) with respect to [76]	20	19.4	11.6	23.41	1.02	NI

Here, NOG = Number of Gates, GO = Garbage Output, QC = Quantum Cost

TABLE 4.15: Comparison of Different Reversible 64-bit Comparators

Circuit	NOG	GO	QC	Area	Power	Delay
Proposed Method	192	253	827	2705.5	7503.7	9.57
Existing Design [74]	444	316	1014	3645	11758.47	12.64
Existing Design [75]	576	378	1143	5682.5	16927.52	1.48
Existing Design [76]	254	316	896	3645	7770.68	0.74
Improvement (%) with respect to [74]	56.8	19.9	18.4	25.77	36.19	24.29
Improvement (%) with respect to [75]	66.6	33.1	27.6	52.39	55.67	NI
Improvement (%) with respect to [76]	24.4	19.9	7.7	25.77	3.43	NI

Here, NOG = Number of Gates, GO = Garbage Output, QC = Quantum Cost

4.2.4 Summary

This chapter discusses the detail design procedure of the reversible memory modules and reversible arithmetic circuits. The proposed components have been compared with existing counterparts. The simulation results proved the accuracy of the proposed circuits.

Chapter 5

Implementation of Proposed Reversible CPU

In this chapter, the design methodologies of the reversible arithmetic logic unit, reversible control unit and the overall reversible central processing unit have been discussed.

5.1 Proposed Design of the Reversible Arithmetic Logic Unit

Reversible arithmetic logic unit (RALU) is the fundamental building block of the reversible central processing unit (RCPU) of a computer. It performs both logical and mathematical operations on binary numbers. This is the unit where actual calculations take place. Reversible adder, subtracter, multiplier, divider performs the required arithmetic tasks. Different kinds of computers have different ALUs. But all of the ALUs contain two units: arithmetic unit and logic unit, which are the basic structures. The proposed RALU also has mainly two units: Reversible Arithmetic Unit and Reversible Logic Unit.

The proposed RALU performs total sixteen operations. It takes two inputs A and B. The RALU can perform various operations based on the input decision applied to the four selection control input M, Sel_2, Sel_1 and Sel_0. Table 5.1 shows the

operation lists of the proposed RALU. Figure 5.1 shows the basic block diagram of the proposed RALU.

TABLE 5.1: List of operations of the Proposed Reversible ALU

M	Sel_2	Sel_1	Sel_0	ALU Operations
0	0	0	0	$A + B$
0	0	0	1	$A - B$
0	0	1	0	$A + 1$
0	0	1	1	$A - 1$
0	1	0	0	$A \times B$
0	1	0	1	A/B
0	1	1	0	Shift
0	1	1	1	Compare
1	0	0	0	$A \oplus B$
1	0	0	1	$A\|B$ (OR)
1	0	1	0	AB
1	0	1	1	A'
1	1	0	0	$(A \oplus B)'$
1	1	0	1	$(A\|B)'$
1	1	1	0	$(AB)'$
1	1	1	1	A

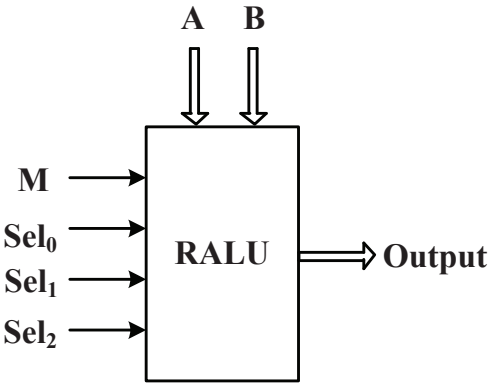

FIGURE 5.1: Block Diagram of the Proposed Reversible Arithmetic Logic Unit

The RALU operations are divided into two main categories: Arithmetic operations and Logic operations. The Control input M selects the arithmetic or logic operation. When $M = 0$ arithmetic operation is performed and when $M = 1$ logical operation is performed. Arithmetic Unit performs the arithmetic operations and the Logic Unit performs logic operations. The outputs of the arithmetic unit and logic unit is connected to the input lines of the multiplexer. The select

inputs of the multiplexer selects the desired output and passes it to the final output of the RALU. The following section presents the proposed design of reversible multiplexer which is an important component of the proposed RALU.

5.1.1 Proposed Reversible Multiplexer

A Multiplexer (Mux) is a device that sends multiple signals on a carrier channel at the same time in the form of a single complex signal to another device that recovers the separate signals at the receiving end. In other words, multiplexer selects one of many data sources and outputs that source into a single channel [54]. Thus, mux is also known as a data selector. The routing of the desired data input to the output is controlled by select inputs. For example, if a two-input multiplexer has data inputs I_0 and I_1 and select input S, then the logic level applied to the S input determines the output Z. If $S = 0$, $Z = I_0$; if $S = 1$, $Z = I_1$.

This section discusses the proposed reversible $4n$-to-n multiplexer which is used to select the outputs of the proposed reversible arithmetic logic unit. Considering the simplest case, $n = 1$, a 4-to-1 reversible mux can be generated. Figure 5.2 shows the architecture of the proposed 4-to-1 reversible mux and the corresponding timing diagram of the proposed 4-to-1 reversible mux is shown in Figure 5.3.

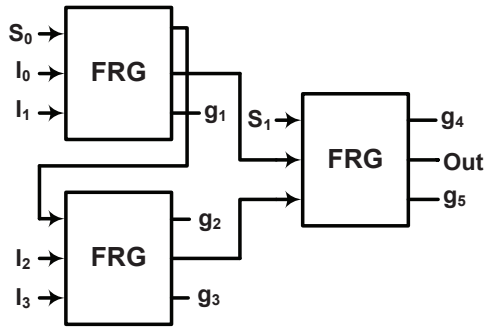

FIGURE 5.2: Proposed 4-to-1 Reversible Multiplexer

From Figure 5.2, it is found that the 4-to-1 reversible mux produces five garbage outputs, whereas no constant input is required. These are the optimum number of garbage outputs and constant input for 4-to-1 reversible Mux.

The proposed reversible multiplexer is simulated using Microwind DSCH 3.0 [38] software on a computer, which has Intel(R) Core(TM) i7-4790 CPU with 3.60

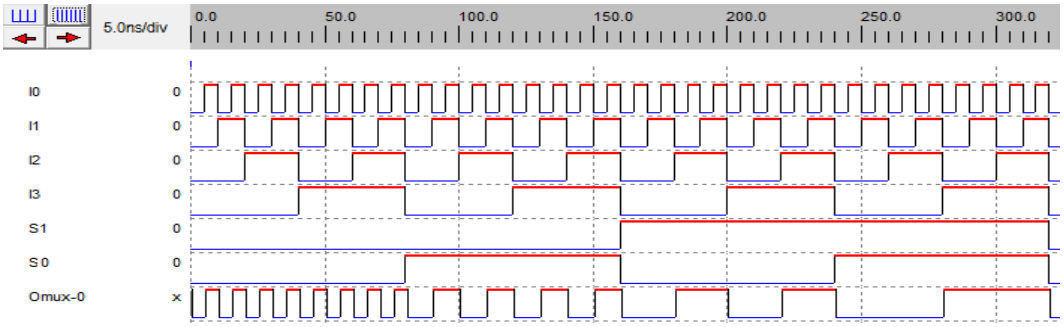

FIGURE 5.3: Timing Diagram of the Proposed 4-to-1 Reversible Multiplexer

GHz Clock Speed and 4.00 GB RAM. Figure 5.3 shows the simulation result of the proposed reversible multiplexer. From Figure 5.3, it is found that when both control signals are low, the final output O_{mux0} is I_0 $(0 - 80$ ns). However, when control signal $S_0 = 1$ and $S_1 = 0$, the final output O_{mux0} is I_1 $(80 - 160$ ns), when control signal $S_0 = 0$ and $S_1 = 1$, the final output O_{mux0} is I_3 (160240 ns) and so on.

Table 5.2 shows a comparative study of proposed 4-to-1 multiplexer with the existing designs [78, 79], from where it can be determined that the improvements in proposed design are 50% or more. Algorithm 5 represents the proposed $4n$-to-n reversible mux design procedure. The primary inputs of the algorithm is $4n$ data bits and $(n + 1)$ control bits. Initially, the algorithm builds a 4-to-1 reversible mux circuit, then recursively builds a $4n$-to-1 reversible mux using the procedure of 4-to-1 reversible mux.

Algorithm 5 Reversible $4n$-to-n Multiplexer $RMux(I, S, F2G, FRG)$

Input: Data input $I(I_0, I_1, ..., I_{4n-1})$, Control Input $S(S_0, S_1, ..., S_{n-1})$
Output: Data output
$i = input$
$o = output$
$j = 0$
begin procedure (4-to-1 RMux)
$S_j \rightarrow first.i.FRG.1\&2, I_j \rightarrow second.i.FRG.1$
$I_{j+2} \rightarrow second.i.FRG2, I_{j+1} \rightarrow third.i.FRG.1$
$I_{j+3} \rightarrow third.i.FRG.2, j++, S_j \rightarrow first.i.FRG.3$
$second.o.FRG.1 \rightarrow second.i.FRG.3$
$second.o.FRG.2 \rightarrow third.i.FRG.3$
return $second.o.FRG.3 \rightarrow$ desired output
end procedure

TABLE 5.2: Comparison of Different 4-to-1 Reversible Mux

Circuit	Number of gates	Garbage Outputs	Quantum Cost	Delay
Proposed Circuit	3	5	15	3
Existing Circuit [78]	9	11	57	6
Existing Circuit [79]	7	11	57	7

Theorem 5.1. *A 4n-to-n reversible mux requires at least (4n+1) garbage outputs and no constant inputs, where n is number of data bits.*

Proof: Let n be the number of data bits. Then, for a $4n$-to-n reversible mux, there are $4n$ data inputs and $(n+1)$ select inputs. So, to preserve the one-to-one mapping of reversibility there should be at least $(5n+1)$ outputs. Among these $(5n+1)$ outputs, there are n primary outputs. Hence, there should be at least $(4n+1)$ garbage outputs for $4n$-to-n reversible Mux, which causes no constant input.

Theorem 5.2. *Let n be number of data bits. Then 4n-to-n reversible Mux can be realized with 3n Fredkin gates.*

Proof: An $4n$-to-n Reversible Mux recursively calls n times the design procedure of the 4-to-1 Reversible Mux. According to our design procedure, 4-to-1 reversible Mux requires three reversible Fredkin gates. So, $4n$-to-n reversible Mux can be realized with $3n$ reversible Fredkin gates.

Theorem 5.3. *Let n be number of data bits. Then the quantum cost of 4n-to-n reversible Mux is 15n.*

Proof: A $4n$-to-n Reversible Mux recursively calls n times the design procedure of the 4-to-1 Reversible Mux. From Theorem 5.3, it is proved that a $4n$-to-n reversible Mux can be realized with $3n$ reversible Fredkin gates. The quantum cost of each FRG gate is 5 [14]. So, the total quantum cost of the $4n$-to-n reversible Mux is $5 \times 3n = 15n$.

5.1.2 Proposed Reversible Arithmetic Unit

The proposed reversible arithmetic unit and logic unit perform the following operations:

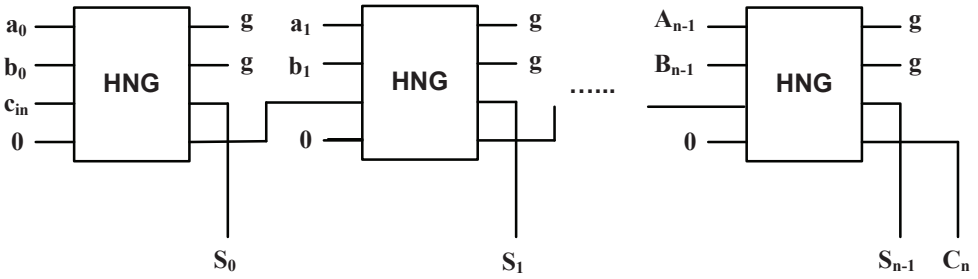

FIGURE 5.4: Full adder using HNG gates [12]

1. Arithmetic operations such as addition, subtraction, increment, decrement, multiplication and division.

2. Bitwise logic operations such as AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

3. Data transfer operation.

4. Shifting operation such as left shift, right shift, left rotate, right rotate.

5. Compare operation.

When $M = 0$, $Sel_2 = 0$, $Sel_1 = 0$, $Sel_0 = 0$ the proposed RALU performs the addition of two operands A and B. The proposed reversible carry look ahead-adder that was discussed in Section 4.2.1 performs the addition.

When $M = 0$, $Sel_2 = 0$, $Sel_1 = 0$, $Sel_0 = 1$ the proposed RALU performs the subtraction operation $A - B$. For subtraction operation, the existing full adder [12] (shown in Figure 5.4) which consists of n HNG gates, where n is the number of bits, has been used with some modifications. To get the 1's complement of the operand B it is inverted using NOT gate. Finally, the C_in bit is asserted to one to get the 2's complement of B.

When $M = 0$, $Sel_2 = 0$, $Sel_1 = 1$, $Sel_0 = 0$ the proposed RALU performs the increment operation of the operand A. The full adder circuit of 5.4 has been used to perform the increment operation where the input B is asserted to zero.

When $M = 0$, $Sel_2 = 0$, $Sel_1 = 1$, $Sel_0 = 1$ the proposed RALU performs the decrement operation of the operand A. The full subtractor circuit has been used to perform the decrement operation where the input B is asserted to zero.

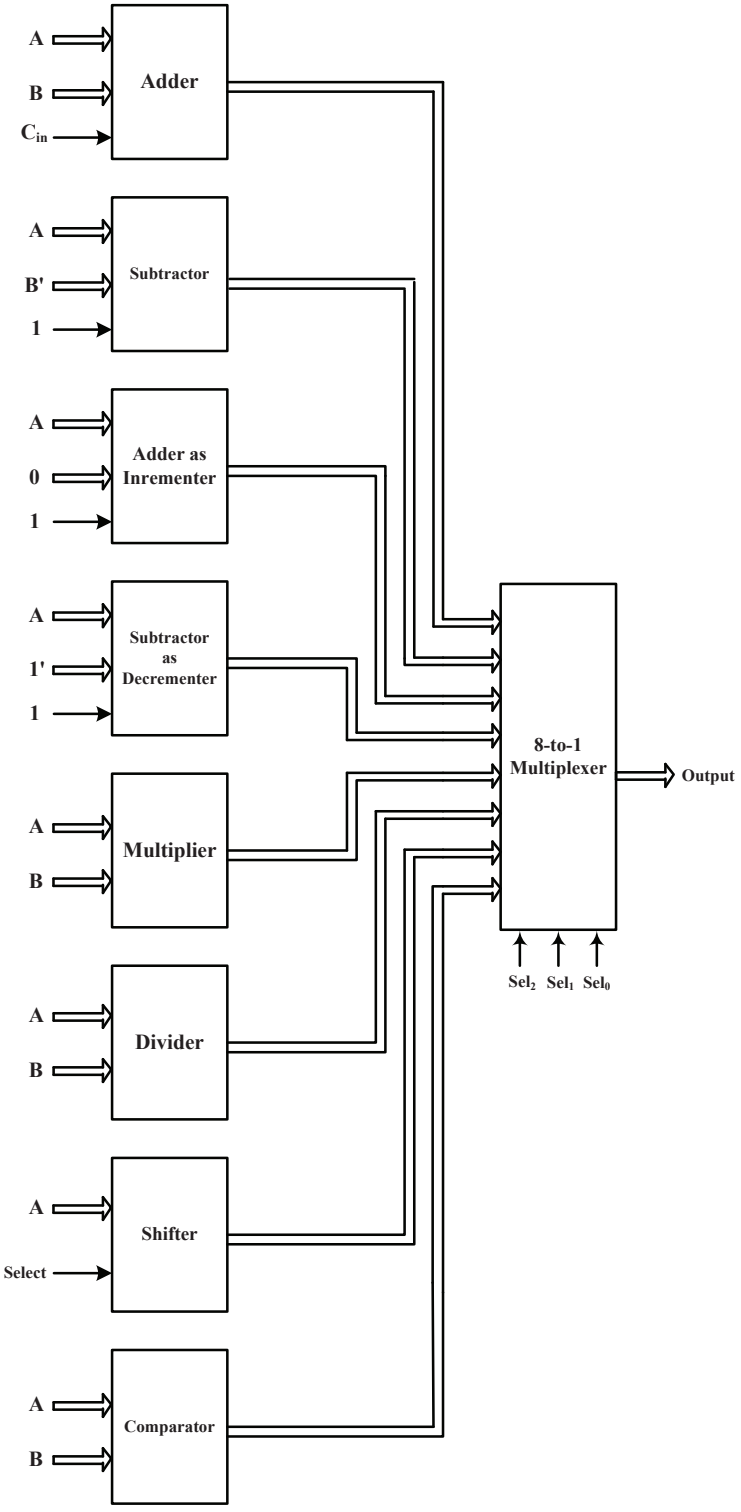

FIGURE 5.5: Proposed Reversible Arithmetic Unit

When $M = 0$, $Sel_2 = 1$, $Sel_1 = 0$, $Sel_0 = 0$ the proposed RALU performs the multiplication operation of two operands A and B, where A is the multiplicand and B is the multiplier. The partial product generation circuit and multi-operand addition circuit of reversible multiplier [80] (shown in Figures 5.6 and 5.7) has been used to perform the multiplication operation of the proposed RALU.

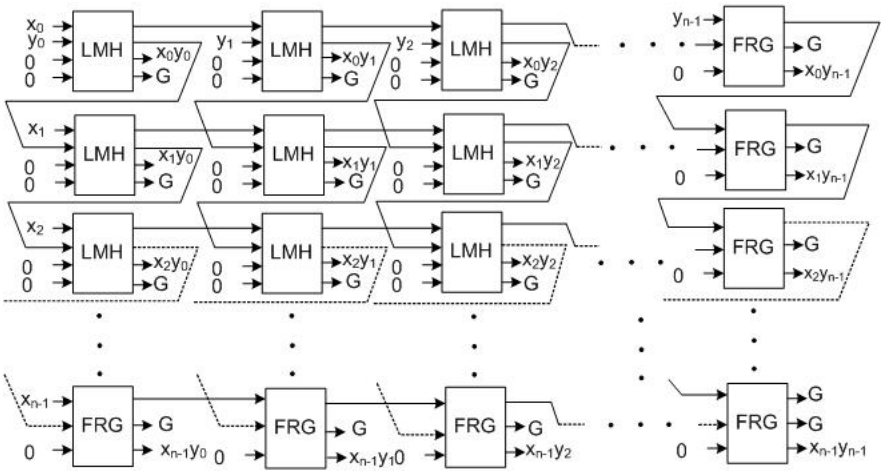

FIGURE 5.6: Architecture of Partial Product Generation Circuit

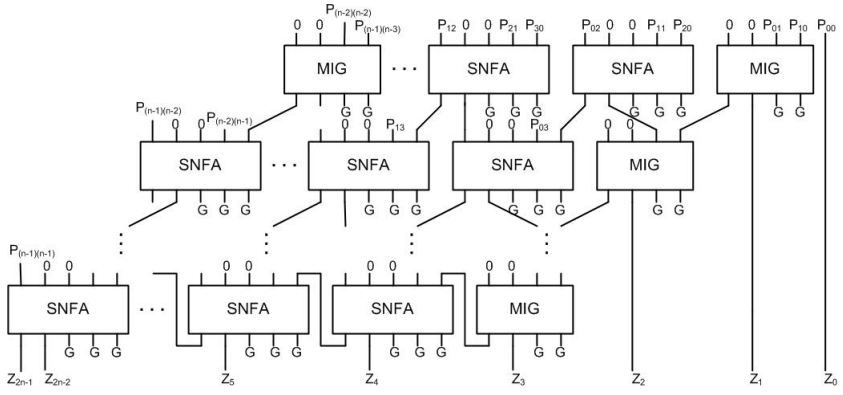

FIGURE 5.7: Generalized Architecture of Multi-Operand Addition Circuit

When $M = 0$, $Sel_2 = 1$, $Sel_1 = 0$, $Sel_0 = 1$ the proposed RALU performs the division operation of two operands A and B, where A is the dividend and B is the divisor. The proposed reversible division circuit that was discussed in Section 4.2.2 is used to perform the division operation of the proposed RALU.

When $M = 0$, $Sel_2 = 1$, $Sel_1 = 1$, $Sel_0 = 0$ the proposed RALU performs the shift operation of operand A. The reversible bidirectional barrel shifter [81] (shown in 5.8)has been used to perform the shift operation of the proposed RALU.

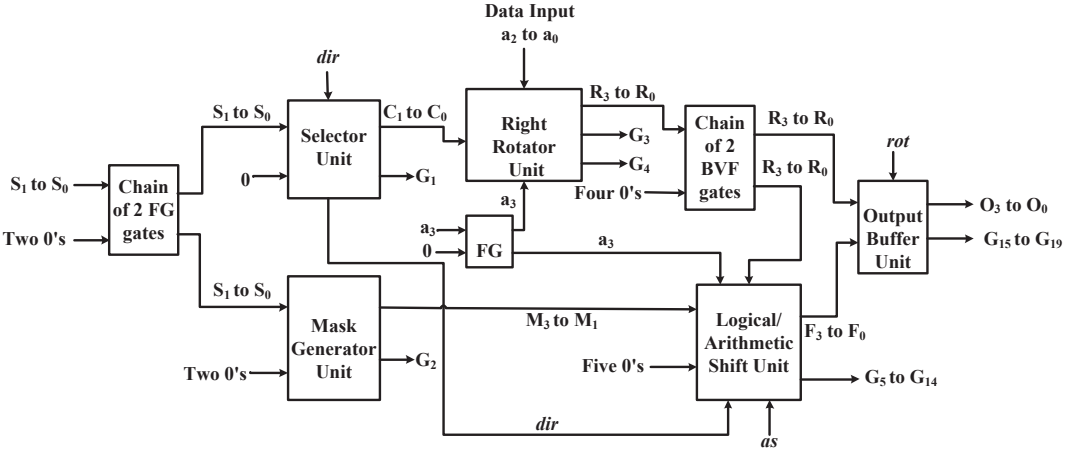

FIGURE 5.8: Block Diagram of Reversible Bidirectional Barrel Shifter

When $M = 0$, $Sel_2 = 1$, $Sel_1 = 1$, $Sel_0 = 1$ the proposed RALU performs the comparison between A and B. To perform the comparison operation, the proposed reversible circuit that was discussed in Section 4.2.3 is used in the RALU.

As the arithmetic unit has eight different types of circuits, a 8-to-1 reversible multiplexer is required to select the desired output. Figure 5.5 shows the architecture of the arithmetic unit.

Theorem 5.4. *A reversible n-bit arithmetic unit can be realized with* $\frac{1}{4}$ $(19n^2 + 86n + 28)$ $+ (\frac{3n}{4} - 7)log_2n$ *gates; where n is the number of data bits.*

Proof: A reversible n-bit arithmetic units consist of reversible CLA adder, subtractor, incrementer, decrementer, multiplier, divider, comparator, barrel shifter and multiplexer circuit.

Theorem 4.1 proved that a reversible n-bit CLA adder requires $n^2 + n$ gates. A reversible n-bit subtractor requires n number of HNG gates and n number of NOT gates. So, a reversible n-bit subtractor requires $2n$ gates. A reversible n-bit incrementer requires n HNG gates. Whereas, a reversible n-bit decrementer requires total $2n$ gates. A reversible $n \times n$ multiplier can be realized with $n(2n-1)$ gates [80]. From Theorem 4.4, it is found that a reversible n-bit divider requires $\frac{3(n+2)^2+2n}{4}$ gates. Theorem 4.10 proved that a reversible n-bit comparator can be realized with $3n$ gates. A reversible barrel shifter requires total $(\frac{3n}{4} - 7)log_2n + \frac{7n}{2} + 2$reversible gates [81]. These eight reversible circuits were combined using a 8-to-1 reversible multiplexer which requires total 6 gates.

So, total number of gates of a reversible n-bit arithmetic unit is
$n^2+n+2n+n+2n+n(3n-1)+\frac{3(n+2)^2+2n}{4}+3n+(\frac{3n}{4}-7)log_2n+\frac{7n}{2}+2+3n$
$=\frac{1}{4}(19n^2+86n+28)+(\frac{3n}{4}-7)log_2n$

Theorem 5.5. *A reversible n-bit arithmetic unit produces $\frac{1}{2}(11n^2+28n-6)+(\frac{n}{2}-3)log_2n$ garbage outputs; where n is the number of data bits.*

Proof: A reversible n-bit arithmetic units consist of reversible CLA adder, subtractor, incrementer, decrementer, multiplier, divider, comparator, barrel shifter and multiplexer circuit.

Theorem 4.3 proved that a reversible n-bit CLA adder produces n^2+n-6 garbage outputs. A reversible n-bit subtractor requires n number of HNG gates and n number of NOT gates. Each HNG gate produces 2 garbage outputs. So, a reversible n-bit subtractor produces total $2n$ garbage outputs. Similarly, a reversible n-bit incrementer and a reversible n-bit decrementer produce $2n$ garbage outputs each. A reversible $n \times n$ multiplier produces $4n(n-1)+1$ garbage outputs [80]. From Theorem 4.5, it is found that a reversible n-bit divider produces $\frac{(n+2)^2}{2}$ garbage oututs. Theorem 4.11 proved that a reversible n-bit comparator produces $4n-3$ garbage outputs. A reversible barrel shifter produces total $(\frac{n}{2}-3)log_2n+n+2$ garbage outputs [81]. These eight reversible circuits were combined using a 8-to-1 reversible multiplexer. The reversible multiplexer produces $4n+1$ garbage outputs.

So, total number of garbage outputs produces by a reversible n-bit arithmetic unit is
$n^2+n-6+2n+2n+2n+4n(n-1)+1+\frac{(n+2)^2}{2}+4n-3+(\frac{n}{2}-3)log_2n+n+2+4n+1$
$=\frac{1}{2}(11n^2+28n-6)+(\frac{n}{2}-3)log_2n$

Theorem 5.6. *A reversible n-bit arithmetic unit requires $\frac{1}{2}(10n^2+111n+20)+(3n-30)log_2n$ quantum cost; where n is the number of data bits.*

Proof: A reversible n-bit arithmetic units consist of reversible CLA adder, subtractor, incrementer, decrementer, multiplier, divider, comparator, barrel shifter and multiplexer circuit.

Theorem 4.2 proved that the quantum cost of a reversible n-bit CLA adder is $3n^2 + 2n$ gates. A reversible n-bit subtractor requires n number of HNG gates and n number of NOT gates. The quantum cost of HNG gate and NOT gate are 6 and 0, respectively. So, the quantum cost of a reversible n-bit subtractor is $6n$. Similarly, the quantum cost of n-bit incrementer and n-bit decrementer is $6n$ both as they both require n gates each. The quantum cost of a reversible $n \times n$ multiplier is $17n(n-1)+1$ [80]. From Theorem 4.6, it is found that the quantum cost of a reversible n-bit divider is $\frac{4(n+2)^2+n}{2}$. Theorem 4.12 proved that the quantum cost of a reversible n-bit comparator is $13n-5$. Quantum cost of the barrel shifter [81] is $(3n-30)log_2n + 14n + 7$. These eight reversible circuits were combined using a 8-to-1 reversible multiplexer whose quantum cost is $15n$.

So, total quantum cost of a reversible n-bit arithmetic unit is
$3n^2+2n+6n+6n+6n+17n(n-1)+1+\frac{4(n+2)^2+n}{2}+13n-5+(3n-30)log_2n+$
$14n+7+15n$
$= \frac{1}{2}\ (10n^2+111n+20) + (3n-30)log_2n$

5.1.3 Proposed Reversible Logic Unit

The proposed RALU can perform AND, OR, NOT, NAND, NOR, EX-OR and EX-NOR operation. When $M = 1$, $Sel_2 = 0$, $Sel_1 = 0$, $Sel_0 = 0$, the proposed ALU performs EX-OR operation. When $M = 1$, $Sel_2 = 0$, $Sel_1 = 0$, $Sel_0 = 1$, it performs the OR operation. When $M = 1$, $Sel_2 = 0$, $Sel_1 = 1$, $Sel_0 = 0$, it calculates the AND of the input bits. The RALU calculates then NOT of the input bits when $M = 1$, $Sel_2 = 0$, $Sel_1 = 1$, $Sel_0 = 1$.

When $M = 1$, $Sel_2 = 1$, $Sel_1 = 0$, $Sel_0 = 0$, the proposed RALU performs EX-NOR operation. Just changing the Sel_0 input, i.e., for $M = 1$, $Sel_2 = 1$, $Sel_1 = 0$, $Sel_0 = 1$, it calculates the NOR operation. $M = 1$, $Sel_2 = 0$, $Sel_1 = 1$, $Sel_0 = 0$, it calculates the NAND operation of input A. The RALU just transfers A when the select inputs $M = 1$, $Sel_2 = 0$, $Sel_1 = 1$, $Sel_0 = 1$. Algorithm 6 presents the procedure to construct the logic module. Figure 5.9 shows the detail design of the logic module.

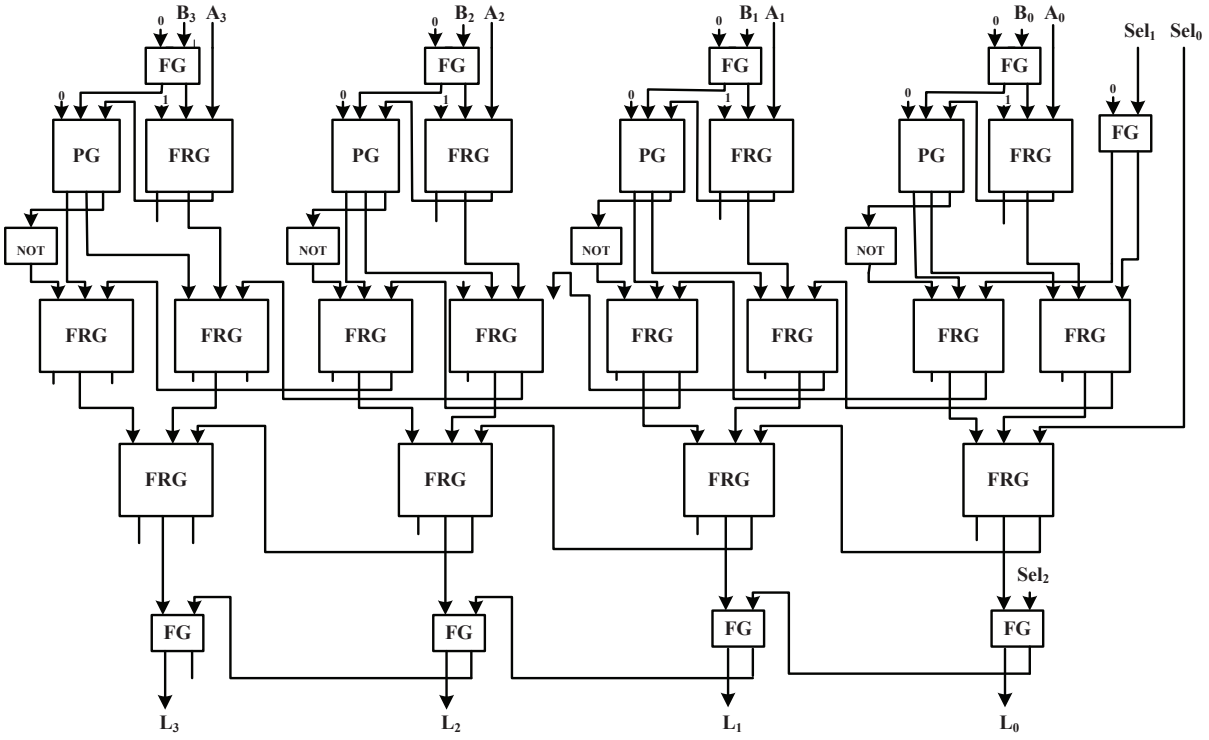

FIGURE 5.9: Proposed Reversible Logic Unit

Algorithm 6 Logic Unit

Input: First Operand A_i, Second Operand B_i, Select Input Sel_2, Sel_1, Sel_0
Output: L_i
fed Sel_1 and 0 to one FG gate
for $i = 1 \rightarrow n$
take one FG, one PG, one FRG and one NOT gate
execute $A_i + B_i$ using FRG; $A_i \oplus B_i$ and A_iB_i using PG and A' using NOT gate
take two FRG gates
fed $A_i + B_i$ and $A_i \oplus B_i$ to one FRG gate
fed AB and A' to another FRG gate
fed the second outputs of previous two steps to a new FRG gate
fed the second output of the last FRG gate and Sel_2 to a FG gate
end for

Theorem 5.7. *Let GC be the minimum number of gates, GO be the minimum number of garbage outputs and QC be the minimum number of quantum cost of the proposed reversible n-bit logic unit, then* $GC = 8n + 1$, $GO = 4n + 4$ *and* $QC = 26n + 1$.

Proof: A 1-bit reversible logic unit requires 3 FG, 4 FRG, 1 PG and 1 NOT gate. So, total number of gates for 1-bit logic unit is 9. If the logic unit is expanded for next consecutive bits the same circuit is required except the first FG gate. So, for each bit expansion, the logic unit requires additional 8 reversible gates.

So, the total number of gates for n-bit logic unit, $GC = 8n + 1$.

For n-bit reversible logic unit, the MSB part of the circuit generates $8(= 4 + 4)$ garbage outputs whereas, the other bits part generates 4 garbage outputs. So, the total number of garbage outputs, $GO = 4n + 4$.

The n-bit reversible logic unit requires $2n+1$ numbers of FG, $4n$ numbers of FRG, n numbers of PG and n numbers of NOT gate. The quantum cost of FG, FRG, PG and NOT gates are 1, 5, 4 and 0, respectively. So, the total quantum cost of the n-bit logic unit is $QC = 2(2n + 1) + 5 \times 4n + 4n + 0 = 28n + 2$.

5.1.4 Proposed Reversible Arithmetic Logic Unit

Figure 5.10 summarizes the overall design of the proposed ALU. The reversible arithmetic unit (designed in Section 5.1.2) and the reversible logic unit (designed in Section 5.1.3) takes input and calculates the operation depending on the control inputs Sel_2, Sel_1 and Sel_0. The outputs of the arithmetic unit and logic unit are fed to a 2-to-1 reversible multiplexer. The control input of the multiplexer is the control bit M. When $M = 0$, the output of the arithmetic unit is the final output of the RALU. When $M = 1$, the output of the logic unit is the final output of the RALU. The final output of the RALU is then stored in the register for future reference.

Theorem 5.8. *A reversible n-bit reversible arithmetic logic unit can be realized with* $\frac{1}{4}\{15n^2 + 89n + 64 + (3n - 28)log_2 n\}$ *gates; where n is the number of data bits.*

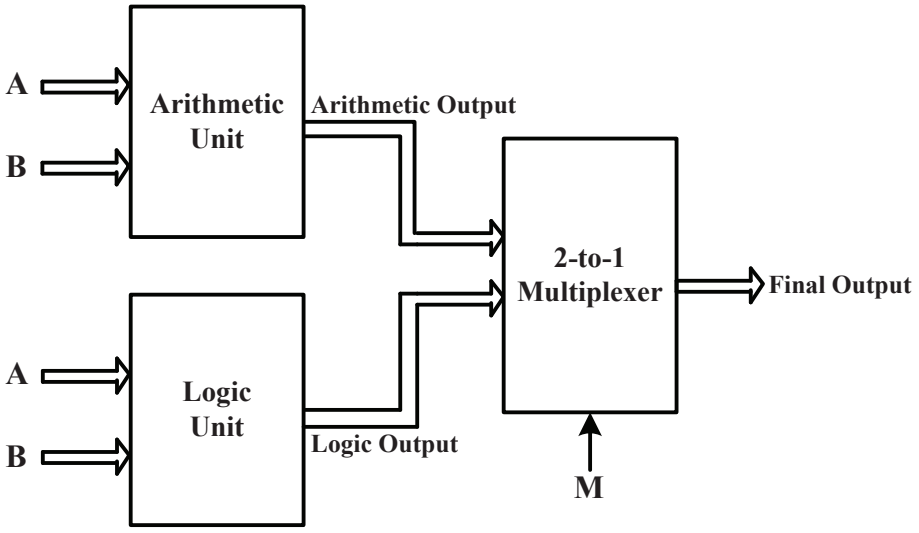

FIGURE 5.10: Proposed Reversible Arithmetic Logic Unit

Proof: From Theorem 5.8, it is found that a reversible arithmetic unit can be constructed from $\frac{1}{4}\{67n + 15n^2 + 56 + (3n - 28)log_2 n\}$ reversible gates, where n is the number of data bits. Theorem 5.7 proved that a reversible logic unit requires $8n + 1$ reversible gates. A 2-to-1 reversible multiplexer can be constructed from a Fredkin gate.

So, total number of gates of the reversible arithmetic logic unit is
$\frac{1}{4}\{67n + 15n^2 + 56 + (3n - 28)log_2 n\} + 8n + 1 + 1$
$= \frac{1}{4}\{15n^2 + 89n + 64 + (3n - 28)log_2 n\}$

Theorem 5.9. *A reversible n-bit reversible arithmetic unit produces* $\frac{1}{2}$ $(11n^2 + 29n + 28)$ $+$ $(\frac{n}{2} - 3)log_2 n$ *garbage outputs; where n is the number of data bits.*

Proof: From Theorem 5.9, it is found that the proposed arithmetic unit produces $\frac{1}{2}$ $(11n^2 + 21n + 16)$ $+$ $(\frac{n}{2} - 3)log_2 n$ garbage outputs, where n is the number of data bits. Theorem 5.7 proved that the proposed reversible logic unit produces $4n + 4$ garbage outputs. A 2-to-1 reversible multiplexer can be constructed from a Fredkin gate which produces 2 garbage outputs.

So, total garbage outputs prodcued by the reversible arithmetic logic unit is
$\frac{1}{2}$ $(11n^2+21n+16)$ $+$ $(\frac{n}{2}-3)log_2 n + 4n+4 +2 = \frac{1}{2}$ $(11n^2+29n+28)$ $+$ $(\frac{n}{2}-3)log_2 n$

Theorem 5.10. *The quantum cost of a reversible n-bit reversible arithmetic unit is* $22n^2 + 86n + 38 + (3n - 29)log_2n$ *quantum cost; where n is the number of data bits.*

Proof: Theorem 5.10 proved that the quantum cost of the proposed reversible arithmetic unit is $22n^2 + 60n + 32 + (3n - 29)log_2n$. From theorem 5.7, it is found that the quantum cost of the the proposed reversible logic unit is $26n + 1$. The quantum cost of a FRG gate, which is used to construct a 2-to-1 reversible multiplexer, is 5.

So, the total quantum cost of the proposed reversible arithmetic logic unit is
$22n^2 + 60n + 32 + (3n - 29)log_2n + 26n + 1 + 5$
$= 22n^2 + 86n + 38 + (3n - 29)log_2n$

5.2 Proposed Reversible Control Unit

A control unit is a circuit that directs operations within the computer's processor by directing the input and output of a computer system. A control unit consists of two decoders, a sequence counter, and a number of control logic gates [63]. It fetches the instruction from instruction register (IR). For example, the block diagram of a 16-bit control unit is shown in Figure 5.11. Here, the instruction register consists of 16 bits. The operation code (bit 12 to bit 14) is decoded by the 3-to-8 decoder. The outputs of the decoder are $D_0, D_1, ..., D_7$. Bit 0 to bit 11 and bit 15 are fed to the control logic gates. The 4-bit sequence counter counts from 0 to 15. The outputs of the counter are decoded into 16 timing signals $T_0, T_1, ..., T_{15}$ by the 4-to-16 decoder. In this section, the detail design of the various components of the Proposed Reversible Control Unit has been presented.

5.2.1 Proposed Reversible Sequence Counter

In this section, a 4-bit reversible sequence counter has been proposed which counts from 0 to 15. The sequence counter is designed using four J-K flip-flops, that was designed in Section 5.2.1, and four Feynman gates. The J-K FFs change

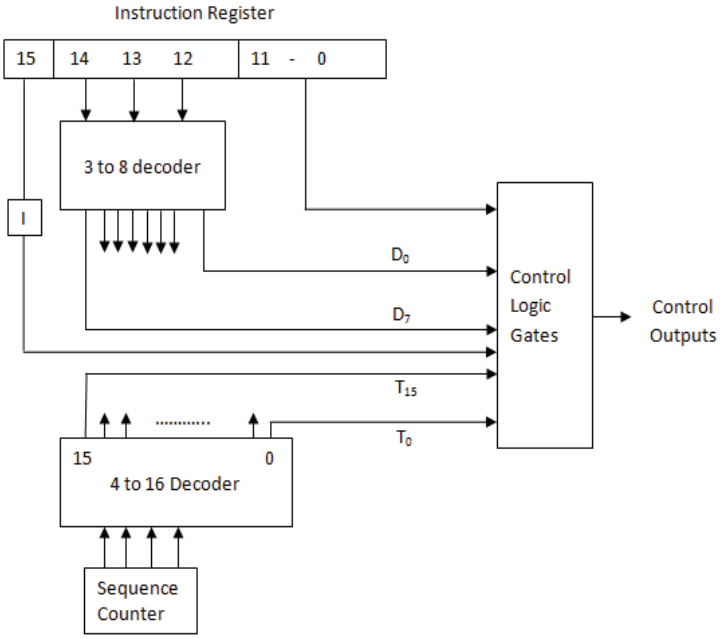

FIGURE 5.11: Block Diagram of the Proposed Control Unit

states with the positive clock edge and the counter counts from 0 to 15. This optimized design of sequence counter produces 4 garbage bits (shown in Figure 5.12). The performance comparison of the counters using proposed and existing techniques has been shown in the Table 5.3. From Table 5.3, it is found that the proposed design outperforms the existing ones in terms of numbers of gates (improvements are 53.33%, 63.12% and 83.72% with respect to [55], [56] and [57], respectively), garbage outputs (improvements are 66.66%, 66.66% and 91.66% with respect to [55], [56] and [57], respectively), delay (improvements are 53.33%, 63.12% and 83.72% with respect to [55], [56] and [57], respectively) and quantum cost (improvements are 13.6%, 32% and 58.54% with respect to [55], [56] and [57], respectively). The sequence counter can be expanded for any number of bits. An n-bit sequence counter requires n J-K FFs and $n-1$ Feynman gates (total $2n-1$ gates), it produces n garbage outputs and it requires $12n + (n-1) = 13n - 1$ quantum cost.

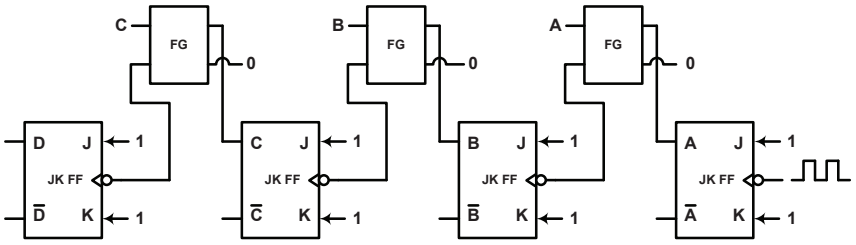

FIGURE 5.12: Proposed Reversible Sequence Counter

TABLE 5.3: Comparison of Different Sequence Counters

Sequence Counter	No. of Gates	Garbage Outputs	Delay	Quantum Cost
Proposed Method	7	4	7	51
Existing Design [55]	15	12	15	59
Existing Design [56]	19	12	19	75
Existing Design [57]	43	48	43	123
Improvement(%) w.r.t [55]	53.33	66.66	53.33	13.56
Improvement(%) w.r.t [56]	63.16	66.66	63.16	32
Improvement(%) w.r.t [57]	83.72	91.66	83.72	58.54

5.2.2 Proposed Reversible Decoder

A decoder is a logic circuit that accepts a set of inputs that represents a binary number and activates only the output that corresponds to that input number. In other words, a decoder circuit looks at its inputs, determines which binary number is present there, and activates the one output that corresponds to that number; all other outputs remain inactive [54]. A decoder with n inputs has 2^n outputs.

To design an efficient reversible decoder circuit, a new reversible gate, namely HL Gate, has been proposed. HL gate is a 4×4 reversible gate with the mapping of (A, B, C, D) to ($P = AB' \oplus B'C \oplus BD'$, $Q = AB \oplus B'C \oplus BD$, $R = A'B \oplus B'C \oplus BD$, $S = AB' \oplus BC \oplus B'D$), where A, B, C, D are the inputs and P, Q, R, S are the outputs. The block diagram of the proposed gate is shown in Figure 5.13. It can be verified from the truth table (shown in Table 5.4) that the input pattern corresponding to a particular output pattern can be uniquely determined and vice versa. Figure 5.14 shows that the quantum cost of the proposed HL gate is seven.

TABLE 5.4: Truth Table of 4×4 Reversible HL gate

INPUT				OUTPUT			
A	B	C	D	P	Q	R	S
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	1	0	1	0
0	0	1	1	1	1	1	1
0	1	0	0	1	1	1	0
0	1	0	1	0	1	0	0
0	1	1	0	1	0	1	1
0	1	1	1	0	1	0	1
1	0	0	0	1	0	0	1
1	0	0	1	1	0	0	0
1	0	1	0	0	1	1	1
1	0	1	1	0	1	1	0
1	1	0	0	1	1	0	0
1	1	0	1	0	0	1	0
1	1	1	0	1	1	0	1
1	1	1	1	0	0	1	1

Here, $P = AB' \oplus B'C \oplus BD'$, $Q = AB \oplus B'C \oplus BD$, $R = A'B \oplus B'C \oplus BD$, $S = AB' \oplus BC \oplus B'D$

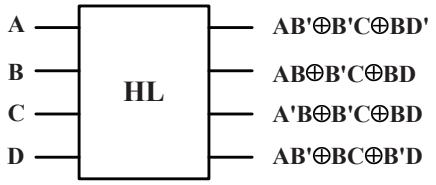

FIGURE 5.13: Block Diagram of the Proposed Reversible HL Gate

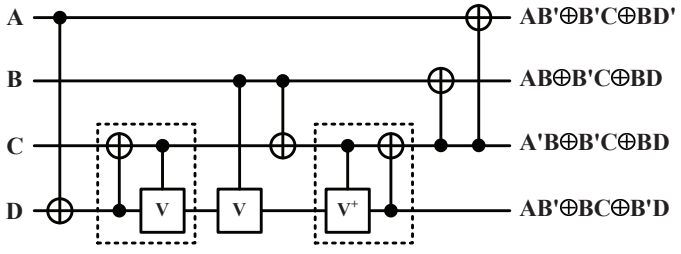

FIGURE 5.14: Quantum Realization of the Proposed Reversible HL Gate

5.2.2.1 Proposed Reversible 2-to-4 Decoder

A 2-to-4 Decoder has two inputs A and B and two outputs $A'B'$, $A'B$, AB' and AB. A 2-to-4 decoder can be constructed from only one HL gate. Figure 5.15 shows the proposed design of a 2-to-4 decoder using one HL gate. The quantum cost of the proposed decoder is seven.

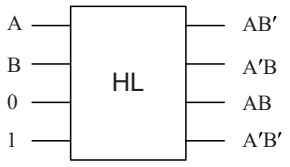

FIGURE 5.15: Proposed Reversible 2-to-4 Decoder

The proposed design of the reversible 2-to-4 decoder is simulated using Microwind DSCH 3.0 [38] software on a computer, which has Intel(R) Core(TM) i7-4790 CPU with 3.60 GHz Clock Speed and 4.00 GB RAM. The timing diagram, shown in Figure 5.16, verifies the correctness of the proposed reversible 2-to-4 decoder. Here, two inputs are A and B. C and D are two constant inputs of HL gate. The outputs are $P = A'B'$, $Q = A'B$, $R = AB'$ and $S = AB$. At time 10 ns, $A = 0$, $B = 0$; so, $P = 1$. At time 30 ns, $A = 0$, $B = 0$; so, $Q = 1$. At time 50 ns, $A = 1$, $B = 0$; so, $R = 1$. Finally at time 70 ns, $A = 1$, $B = 1$; so, $S = 1$.

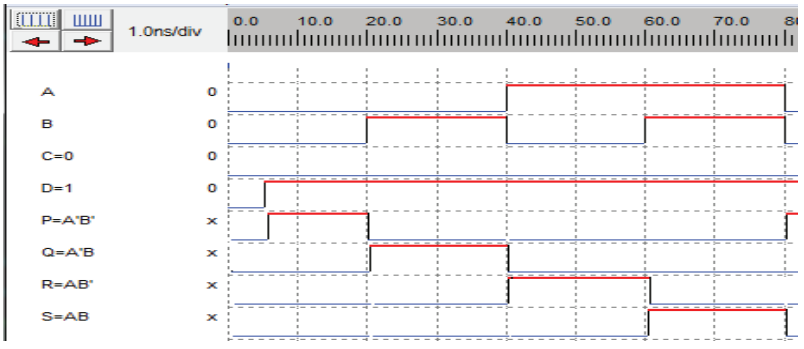

FIGURE 5.16: Simulation Result of Proposed Reversible 2-to-4 Decoder

The performance comparison of the proposed and the existing 2-to-4 decoders [82], [83], [84] has been shown in Table 5.5. From Table 5.5, it is clear that the proposed design is better than [82] and [83] in respect of all cost parameters. The proposed design has same number of gates, garbage output and delay of design

[84], but the quantum cost of the proposed design is less than the design of [84]. So, the proposed design is the best among all designs in terms of all parameters of reversible circuit design.

TABLE 5.5: Comparison of Different Reversible 2-to-4 Decoders

2-to-4 Decoder	No. of Gates	Garbage Outputs	Delay	Quantum Cost
Proposed Method	1	0	1	7
Existing Design [82]	4	2	4	14
Existing Design [83]	3	2	3	15
Existing Design [84]	1	0	1	8
Improvement(%) w.r.t [82]	75	100	75	50
Improvement(%) w.r.t [83]	66.66	100	66.66	53.33
Improvement(%) w.r.t [84]	Same	Same	Same	12.5

5.2.2.2 Proposed Reversible 3-to-8 Decoder

A reversible 3-to-8 decoder can be designed using one 2-to-4 reversible decoder and four Fredkin gates. The proposed design of the 3-to-8 decoder is shown in Figure 5.17. The proposed reversible 3-to-8 decoder requires 5 reversible gates which produces one garbage output and requires 27 quantum cost. The performance comparison of the proposed and the existing 3-to-8 decoder [82], [83] has been shown in Table 5.5. The table doesn't show any comparison with [84] as this paper shows the design of 2-to-4 decoder only.

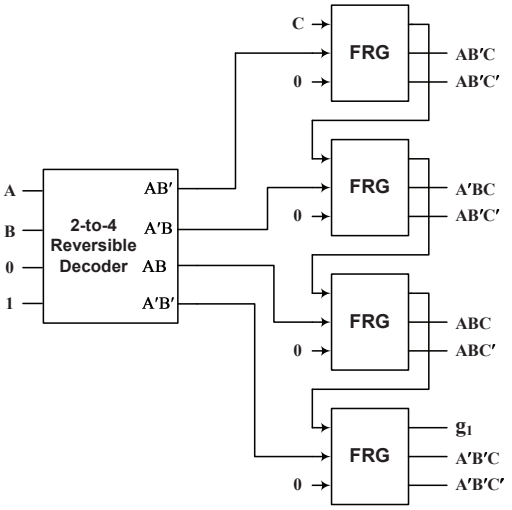

FIGURE 5.17: Proposed Reversible 3-to-8 Decoder

TABLE 5.6: Comparison of Different Reversible 3-to-8 Decoders

2-to-4 Decoder	No. of Gates	Garbage Outputs	Delay	Quantum Cost
Proposed Method	5	1	4	27
Existing Design [82]	11	6	11	38
Existing Design [83]	7	3	14	32
Improvement(%) w.r.t [82]	54.54	83.33	63.63	28.95
Improvement(%) w.r.t [83]	28.57	66.66	63.63	28.95

5.2.2.3 Proposed Reversible n-to-2^n Decoder

A reversible n-to-2^n decoder is designed with a $(n-1)$-to-$2^{(n-1)}$ reversible decoder and 2^{n-1} Fredkin gates. Figure 5.18 shows the proposed design of the n-to-2^n decoder.

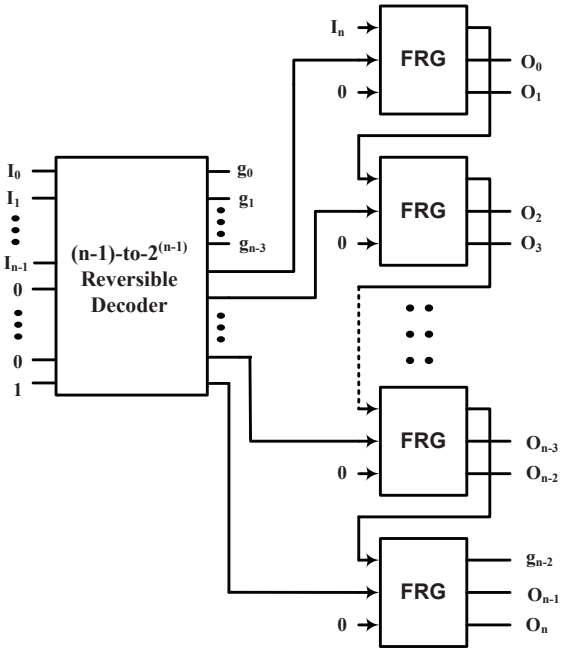

FIGURE 5.18: Proposed Reversible n-to-2^n Decoder

Theorem 5.11. *An n-to-2^n reversible decoder can be realized by at least 2^{n-3} reversible gates, where n is the number of bits and $n \geq 2$.*

Proof: The above statement is proved by mathematical induction.

A 2-to-4 ($n = 2$) decoder is constructed using one HL gate. So, a 2-to-4 decoder requires at least $1(= 2^2 - 3)$ reversible gates. So, the statement holds for the base

case $n = 2$.

Assume that, the statement holds for $n = k$. So, a k-to-2^k decoder can be realized by at least $2^k - 3$ reversible gates.

A $(k+1)$-to-$2^{(k+1)}$ decoder is constructed using k-to-2^k decoder and 2^k FRG gates. So, total number of gates required to construct a $(k+1)$-to-$2^{(k+1)}$ decoder is at least
$2^k - 3 + 2^k$
$= 2 \times 2^k - 3$
$= 2^{(k+1)} - 3$

So, the statement holds for $n = k + 1$.

Therefore, an n-to-2^n reversible decoder can be realized by at least 2^{n-3} reversible gates.

Theorem 5.12. *An n-to-2^n reversible decoder generates at least $n - 2$ garbage outputs, where n is the number of bits and $n \geq 2$.*

Proof: The above statement is proved by induction.

A 2-to-4 ($n = 2$) decoder requires only one HL gate. All of the outputs of the HL gates are used to design a 2-to-4 decoder. So, no garbage output is produced by the HL gate (shown in Figure 5.15). So, a 2-to-4 decoder generates at least $0(= 2 - 2)$ garbage output. So, the statement holds for the base case $n = 2$.
Assume that, the statement holds for $n = k$. So, a k-to-2^k decoder generates at least $k - 2$ garbage outputs.

A $(k+1)$-to-$2^{(k+1)}$ decoder is constructed using k-to-2^k decoder and 2^k FRG gates. k-to-2^k decoder generates at least $k - 2$ garbage outputs and only the last FRG

gate produces one garbage output. So, total number of garbage outputs generated by a $(k+1)$-to-$2^{(k+1)}$) decoder is at least $k-2+1=(k+1)-2$
So, the statement holds for $n=k+1$.

Therefore, an n-to-2^n reversible decoder generates at least $n-2$ garbage outputs, where $n \geq 2$.

5.2.3 Proposed Reversible Control Gates

The control inputs of the registers are LD (load), INR (increment) and CLR (clear). This section presents the gate structure associated with the control inputs of Address Register (AR). The control functions that change the content of AR are as follows:
$R'T_0 : AR \leftarrow PC$
$R'T_2 : AR \leftarrow IR(0-11)$
$D'_7IT_3 : AR \leftarrow M[AR]$
$RT_0 : AR \leftarrow 0$
$D_5T_4 : AR \leftarrow AR+1$
where, D_5 and D_7 are decoder operations and T_0, T_2, T_3 and T_4 are timing signals [20].
The above control functions can be combined into the following Boolean operations:
$LD(AR) = R'T_0 + R'T_2 + D_7IT_3$
$CLR(AR) = RT_0$
$INC(AR) = D_5T_4$
Here, the first statement specifies transfer of information from a register or memory to AR. The second statement clears AR and the last statement increments AR by 1 [63].
The gate structure associated with the above equations is shown in Figure 5.19. In a similar fashion the control gates for the other register can be derived as well as the logic needed to control the read and write inputs of memory.

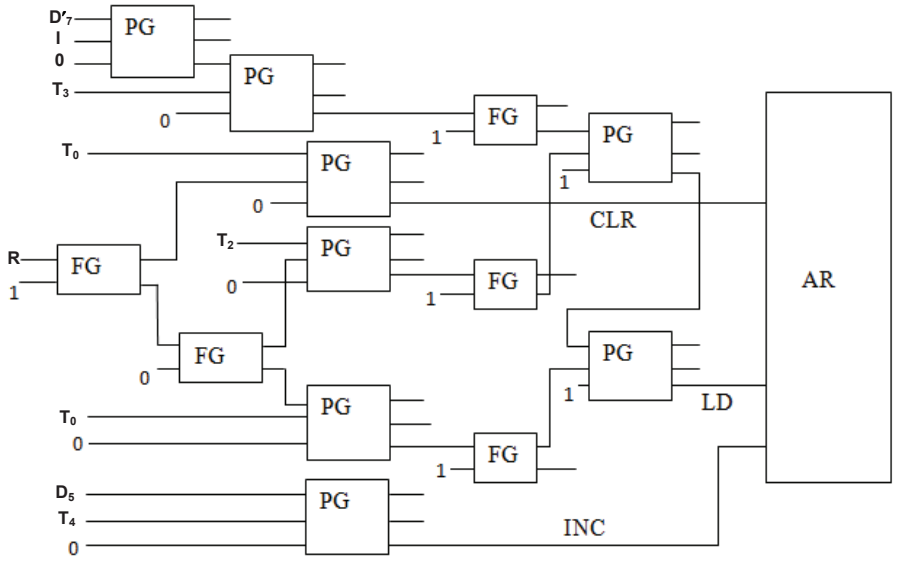

FIGURE 5.19: Proposed Reversible Control Gates

5.2.4 Construction Procedure and Complexities of the Proposed Reversible Control Unit

Section 5.2 shows the block diagram of a control unit. The control unit consists of two decoders, a sequence counter, and a number of control logic gates. It fetches the instruction from instruction register. The inputs to the control logic gates come from the two decoders, the I flip-flop and bits 0 through 11 of IR. The outputs of the control logic circuit are:

- Signals to control the inputs of the registers.
- Signals to control the read and write inputs of memory.
- Signals to set, clear or complement the flip-flops.

The steps to construct a 16-bit control unit are described in Algorithm 7.

Algorithm 7 Construction of Reversible Control Unit

Construct a 3-to-8 decoder

Take the inputs of the 3-to-8 decoder from the instruction register (bit 12 to bit 15)

Construct a 4-bit sequence counter

Construct a 4-to-16 bit decoder

Take the inputs of the 4-to-16 decoder from the outputs of the sequence counter

Construct a control logic gate

Take the inputs of the control logic gate from the instruction register (bit 0 to bit 11 and bit 15), 3-to-8 decoder (D_0 to D_7) and 4-to-16 decoder (T_0 to T_{15})

The outputs of the control register provide necessary signals to carry out the operation of the instruction register

Theorem 5.13. *Let ga_{cu} be the number of gates required to realize a reversible n-bit control unit (where, n is the number of bits) of a processor, ga_{ir} be the number of gates required to realize an n-bit instruction register, ga_{d1} be the number of gates required to realize an m-to-2^m decoder (where $m < n$), ga_{sc} be the number of gates required to realize a (log2n)-bit sequence counter, ga_{d2} be the number of gates required to realize a log_2n-to-n decoder and ga_{cl} be the number of gates required to realize control logic associated to AR. Then, $ga_{cu} >= ga_{ir} + ga_{d1} + ga_{sc} + ga_{d2} + ga_{cl} = 2log_2n + 2m + 2n - 9$.*

Proof: According to Section 4.1.2, an n-bit instruction register requires at least $2n$ reversible gates. So, $ga_{ir} \geq n$. An n-bit instruction has m-bit opcode (where $m < n$). An m-to-2^m reversible decoder is required to decode the opcode. According to Theorem 5.11, an m-to-2^m decoder requires at least $2^m - 3$ reversible gates. So, $ga_{d1} \geq 2^m - 3$. A (log_2n)-bit reversible sequence counter counts from 0 to $n - 1$. According to Section 5.2.1, a (log_2n)-bit sequence counter requires at least $2log_2n - 1$ reversible gates. Thus, $ga_{sc} \geq 2log_2n - 1$. A log_2n-to-n decoder is required to decode the output of the sequence counter. A $log_2n - to - n$ decoder requires at least $2log_2n - 3 = n - 3$ reversible gates, so $ga_{d2} \geq n - 3$. The number of gates of the control logic associated to IR depends on the instruction set of a computer. In the proposed design, the number of gates for the given instruction set is 13, where, $ga_{d2} \geq 15$. Therefore, the total number of gates for reversible

n-bit control unit of a processor is $ga_{cu} >= ga_{ir} + ga_{d1} + ga_{sc} + ga_{d2} + ga_{cl} = n + (2m - 3) + (2log_2n - 1) + (n - 3) + 13 = 2log_2n + 2m + 2n - 9$.

Theorem 5.14. *Let go_{cu} be the number of garbage outputs generated by a reversible control unit of a processor, go_{ir} be the number of garbage outputs produced by the n-bit instruction register, go_{d1} be the number of garbage outputs for an m-to-2^m decoder (where $m < n$), go_{sc} be the number of garbage outputs for a (log_2n)-bit sequence counter, go_{d2} be the number of garbage outputs produced by a log_2n-to-n decoder and go_{cl} be the number of garbage outputs generated by the control logic gates, then $go_{cu} \geq go_{ir} + go_{d1} + go_{sc} + go_{d2} + go_{cl}, = 2log_2n + m + n + 16$.*

Proof: An n-bit instruction register produces at least $n + 1$ garbage outputs (according to Section 4.1.2). So, $go_{ir} \geq n + 1$. An n-bit instruction has m-bit opcode (where $m < n$). According to Theorem 5.12, an m-to-2^m reversible decoder generates at least $m - 2$ garbage outputs. So, $go_{d1} \geq m - 2$. According to Section 5.2.1, a (log_2n)-bit sequence counter generates at least (log_2n) garbage bits. So, $go_{sc} \geq log_2n$. A log_2n-to-n decoder is required to decode the output of the sequence counter. A log_2n-to-n decoder produces at least $log_2n - 2$ garbage outputs. Thus, $go_{d2} \geq log_2n - 2$. The number of garbage outputs produced by the control logic associated to IR depends on the instruction set of a computer. In the proposed design, the number of garbage outputs for the given instruction set is 19, so $go_{d2} \geq= 19$. Therefore, the total number of garbage bits generated by a reversible control unit of a processor is $go_{cu} \geq go_{ir} + go_{d1} + go_{sc} + go_{d2} + go_{cl} = (n + 1) + (m - 2) + log_2n + (log_2n - 2) + 19 = 2log_2n + m + n + 16$.

5.2.5 Proposed Reversible Central Processing Unit

Datapath architectures are extremely important in the organization of the processor [85]. Datapath is a set of functional units that carry out data processing operations. Datapaths, along with a reversible control unit, make up the reversible CPU. Without the path architecture, the interrelationships between various components of the reversible processor can not be established.

Figure 5.20 shows the block diagram of the proposed reversible central processing unit. The accumulator register stores intermediate answers or it may be used to

store an operand for a binary operation performed by the ALU. Temporary register stores the second operand of any binary operation performed by the ALU. The status register stores the overflow information. Instruction register is a part of the control unit. It stores the instruction fetched from main memory. The program stores the address of the next instruction to be executed. The data register takes input from memory. It is directly connected to the data bus. The processor control unit directs the data register to take input or put the stored data from or to the memory modules. Register file is the array of registers. The processor can stored several information regarding an instruction in register file to speed up the computation. The register file contains total sixteen registers. The arithmetic logic unit performs arithmetic and logic operations based on the selections chosen by the processors control unit. Control unit of the processor controls all components of the processor. The control unit must be designed efficiently so that the computer can take full advantage of the processors features through it.

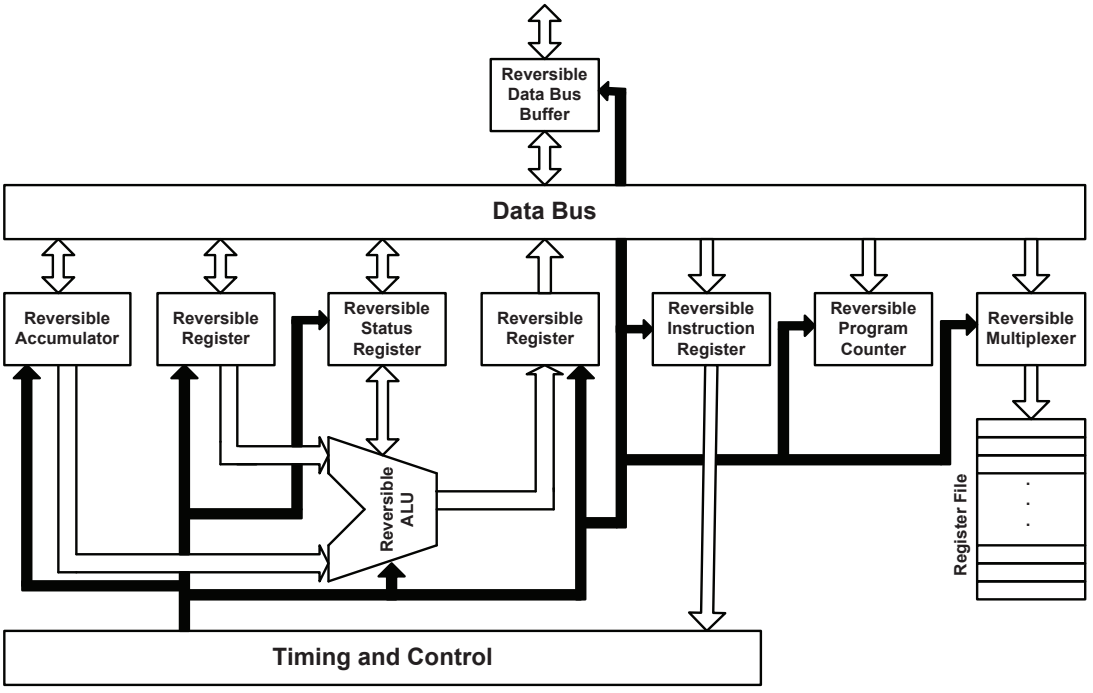

FIGURE 5.20: Outline of the Design of the Proposed Reversible Processor

Path architectures are extremely important in the organization of the processor. Without the path architecture, the interrelationships between various components of the reversible processor can not be established. Arithmetic and Logic Unit (ALU) operates central role in processor operations. ALU can operate arithmetic

operations where operands can be fetched from several registers from the register pool in the register file. So the ALU must have proper and accurate way to access the data bus.

The datapath of the proposed reversible processor is shown in Figure 5.21. The control signals are not shown in the diagram for simplicity.

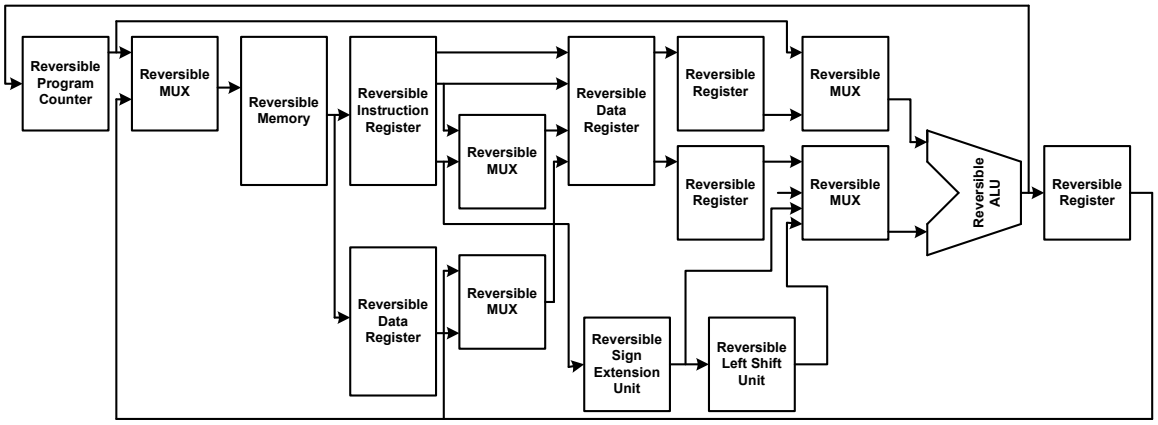

FIGURE 5.21: Datapath Design of the Proposed Reversible Processor

While constructing the datapath and integrating the RALU and RCU, the following steps were followed to get an optimized datapath:
1. Initially all the components were placed. The position of the component varies depending on the inter component connectivity.
2. The components were grouped into disjoint subsets.
3. The components with same characteristics are merged using Boolean Matching Algorithm [86].
4. The components are placed efficiently to avoid any physical overlapping.

Floorplan design is an important step in the physical design process of VLSI circuits [87]. A floorplan F is a subdivision of an enclosing rectangle by horizontal and vertical line segments into nonoverlapping rectangles. A rectangle not subdivided by any line segment is called a basic block. To optimize the layout, it is allowed to move the line segments and to choose the dimensions of the basic blocks. Two floorplans $F1$ and $F2$ are equivalent if for every pair of basic blocks, the above-below relation and left-right relation of the pair of blocks are the same in $F1$ and $F2$ [88]. While designing the datapath and the placement of the blocks

of the proposed reversible central processing unit, a block placing algorithm (Algorithm 8) was used.

Algorithm 8 Block Placement of Reversible Central Processing Unit

if A block is a basic block **then**
  Place the block in the floor
end if
if The floor has vertically sliced two sub floorplans $F1$ and $F2$ **then**
  Find nonredundant realizations of $F1$ and $F2$ and place the blocks
  Find the floor whose height is determined by $F2$
  Find the floor whose height is determined by $F1$
end if
if The floor has horizontally sliced two sub floorplans $F1$ and $F2$ **then**
  Find nonredundant realizations of $F1$ and $F2$ and place the blocks
  Find the floor whose width is determined by $F2$
  Find the floor whose width is determined by $F1$
end if

Combining both of the units (RALU and RCU), the compact design of the proposed reversible central processing unit can be obtained (shown in Figure 5.22).

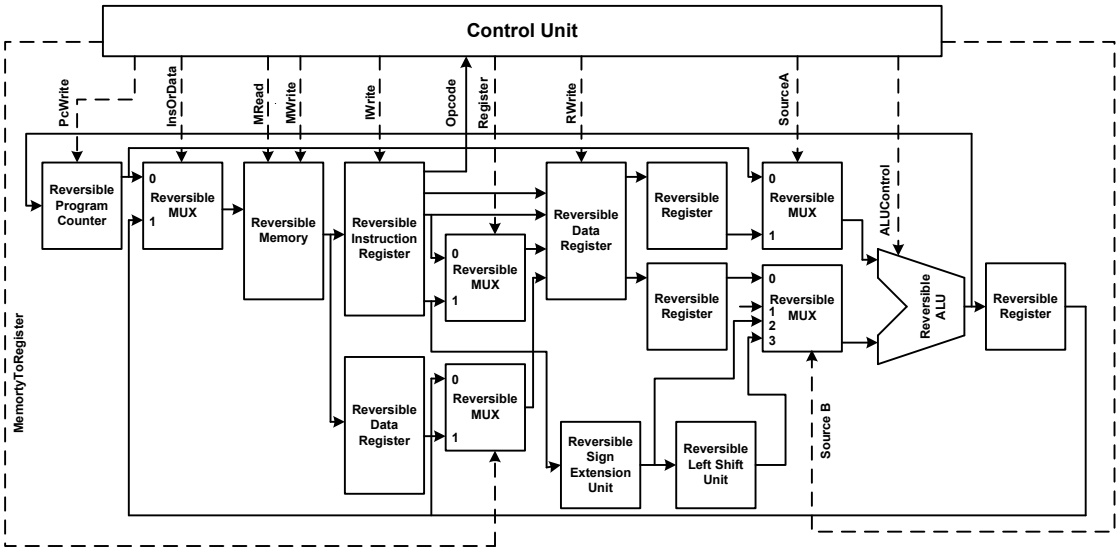

FIGURE 5.22: Integrated Design of the Proposed Reversible Central Processing Unit

Upon receiving the control signal (PCWrite) from the control unit, the program counter sends its content to the multiplexer. The multiplexer gets control signal (InsOrData) from the control unit. Depending on this control signal, it passes the instruction address to the memory or the data to the memory. The control unit sends control signal (MRead or MWrite) to the memory. The memory sends the

corresponding instruction to the instruction to the instruction register or corresponding data to the data register. The instruction register is updated when it gets the control signal (IWrite) from the instruction register. The corresponding opcode of the instruction is sent to the control unit. The address of the operands are sent to the data register. The destination address is determined from the control signal (Register). The data resister is updated according to the control signal (RWrite) from the control unit. The data register may also get the data from the left bottom multiplexer depending on the control signal (MemoryToRegister). The data register sends the corresponding data to next two reversible registers. The control unit sends control signal (SourceA and SourceB) to next two multiplexers to send desired data to the arithmetic logic unit. The ALU may get the first operand from the data register or from the program counter. It may get the second operand from the data register or from the sign extension unit or from the left shift unit through the second multiplexer. The source of the ALU depends on the control signals (SourceA and SourceB). The control unit sends control signal to the ALU (ALUControl) based on the opcode of the instruction. The ALU performs the arithmetic or logic operation and sends the corresponding result to the register. The register sends back the result to the multiplexer next to the program counter. If the data needs to be stored in the data register, the control unit issues a control signal (InsOrData) and the desired data is stored in the data register, if required. Upon completion of an instruction execution, the CPU fetches the next instruction.

5.2.6 Summary

This chapter discusses the detail design procedure of the reversible central processing unit. The RCPU is divided into two major parts: RALU and RCU. The detail design of the proposed RALU and RCU have been discussed. The cost complexities of the proposed circuits were proved.

Chapter 6

Conclusions and Future Works

The promising computing paradigm of reversible computing has applications in the world of nanocomputing and is attracting the attention of researchers from architecture, design, synthesis and test perspective. The research efforts reported in this dissertation represent a solid contribution towards the advancement of design, synthesis and test of reversible logic circuits for emerging nanotechnologies. In this dissertation, an efficient design of a reversible central processing unit considering metrics of ancilla inputs, garbage outputs, quantum cost, delay, area and power were presented. The functional verification of all the proposed reversible design methodologies for the reversible central processing unit are done using DSCH [38] and CMOS 45 nm Open Cell Library [37].

On the way to design the proposed reversible central processing unit (RCPU), several components of the RCPU were designed separately. Efficient reversible flip-flops such as JK FF and D FF were designed. To design those FFs, two new reversible gates, BJ gate and NP gate, were proposed. The design of data register and instruction register were devised using proposed JK FF and D FF.

Different reversible arithmetic circuits, such as, carry look-ahead adder, comparator, divider etc. were proposed. Different reversible logic operations were implemented in the logic unit of the proposed RCPU. The logic unit performs AND, OR, NOT, AND, NAND, EX-OR and EX-NOR operations. The proposed reversible

arithmetic unit (RALU) can perform the desired operation according to the control inputs. The control unit generates the necessary control signals. Depending on the control signals, the RALU completes the desired calculation.

It has been established with the help of comparisons that the proposed designs are compact and efficient. The proposed designs are also superior and optimized over the existing works in terms of quantum cost, number of gates, number of garbage outputs, delay, area and power. The complexities of the proposed reversible circuits were derived and proved with mathematical functions in terms of numbers of gates, garbage outputs and quantum cost and ancilla inputs. For example, a reversible n-bit arithmetic unit can be realized with $\frac{1}{4}\{67n+15n^2+56+(3n-28)log_2n\}$ reversible gates, it produces $\frac{1}{2}$ $(11n^2+21n+16)$ $+$ $(\frac{n}{2}-3)log_2n$ garbage outputs and it requires $22n^2+60n+32+(3n-29)log_2n$ quantum cost; where n is the number of data bits. The area and power is calculated using CMOS 45 nm Open Cell Library [37]. Simulations using Microwind DSCH software [38] have been done to verify the correctness of the proposed circuits. The timing diagram of the proposed circuits were provided to show the outputs for some possible input combinations.

The proposed reversible central processing unit solve the power dissipation problem of conventional irreversible circuits. Power analysis is a physical attack to cryptosystems. It exploits the fact that the power dissipation of an electronic circuit depends on the actions performed in it. Simple Power Analysis (SPA) and Differential Power Analysis (DPA) attacks use the power-dissipation characteristics as a provider of side-channel information [89]. Using DPA, an attacker can extract information on secret keys by statistically analyzing power consumption measurements from multiple cryptographic operations performed by a crypto-processor. DPA is more difficult to prevent, since even small biases in the power consumption can lead to exploitable weaknesses. Reversible central processing unit protects the crypto-systems from power analysis attacks (SPA and DPA). As the reversible circuits require very small power than the corresponding irreversible circuits, it is extremely hard to analyze the difference of power consumptions with the change of the functions in reversible circuit. For this reason, the characteristics of the reversible functions cannot be determined from power analysis. Thus the proposed reversible processor will prevent any type of power analysis attack

(SPA and DPA), since theoretically no energy will be dissipated from reversible circuits. The proposed RCPU can make a significant contribution to design a crypto-processor [17] which can be used in smart card, storage devices, embedded systems, network routers, firewalls, etc.

The reversible central processing unit presented in this dissertation have wide applicability in the various emerging nanotechnologies such as, Low power CMOS design [90], Field Programmable Gate Arrays (FPGAs) in CMOS technology [91], Optical information processing [92], DNA computing [93], Quantum computing and nanotechnology [94] .

The proposed designs of reversible circuits are yet to be fabricated in a chip. The future plan of this study is to realize the proposed design practically and work with quantum computation and quantum synthesis of the proposed processor circuit as well as its components and improve the proposed ALU to perform more number of operations. In future, there is a scope to design a fault tolerant reversible central processing unit. There is another scope to design reversible central processing unit in quantum dot cellular automata and in multiple-valued logic.

The proposed reversible central processing unit can make a significant contribution in the field of low power reversible computing and quantum computing.

Bibliography

[1] *The Story of the Intel 4004*, 2016 (accessed February 6, 2017). [Online]. Available: http://www.intel.com/content/www/us/en/history/museum-story-of-intel-4004.html

[2] *Hardware Design: Archives - Intel Pentium III Processor*, 2016 (accessed February 6, 2017). [Online]. Available: http://www.intel.com/design/archives/processors/PentiumIII

[3] *Intel Core i7 Processor Series Datasheet*, 2016 (accessed February 6, 2017). [Online]. Available: http://www.intel.com/content/www/us/en/processors/core/core-i7-900-ee-and-desktop-pr

[4] R. Zhou, Y. Li, M. Zhang, and B. Hu, "Novel design for reversible arithmetic logic unit," *International Journal of Theoretical Physics*, vol. 54, no. 2, pp. 630–644, 2015.

[5] M. Morrison and N. Ranganathan, "Design of a reversible alu based on novel programmable reversible logic gate structures," in *2011 IEEE Computer Society Annual Symposium on VLSI*, July 2011, pp. 126–131.

[6] M. Haghparast and A. Bolhassani, "Optimization approaches for designing quantum reversible arithmetic logic unit," *International Journal of Theoretical Physics*, vol. 55, no. 3, pp. 1423–1437, 2016.

[7] H. R. Aradhya, B. P. Kumar, and K. Muralidhara, "Design of control unit for low power {AU} using reversible logic," *Procedia Engineering*, vol. 30, pp. 631 – 638, 2012, international Conference on Communication Technology and System Design 2011.

[8] M. K. Thomsen, H. B. Axelsen, and R. Glück, *A Reversible Processor Architecture and Its Reversible Logic Design*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 30–42.

[9] I. Ilammathi and K. Prasanna, "Design and implementation of a processor using reversible logic," *International Journal of Engineering Research and Technology (IJERT)*, vol. 3, no. 11, pp. 1139 – 1142, November.

[10] R. Wille, M. Soeken, D. Groe, E. Schnborn, and R. Drechsler, "Designing a risc cpu in reversible logic," in *2011 41st IEEE International Symposium on Multiple-Valued Logic*, May 2011, pp. 170–175.

[11] R. G. M. Thomsen and H. Axelsenr, "Towards designing a reversible processor architecture," in *Reversible Computation*, 2009.

[12] M. Haghparast and K. Navi, "A novel reversible full adder circuit for nanotechnology based systems," *Journal of Applied Sciences*, vol. 7, no. 24, pp. 3995–4000, 2007.

[13] K. Kant, *Microprocessors And Microcontrollers: Architecture Programming And System Design*. PHI Learning Pvt. Ltd., 2007.

[14] E. Fredkin and T. Toffoli, "Conservative logic," *International Journal of Theoretical Physics*, vol. 21, no. 3-4, pp. 219–253, 1982.

[15] R. Landauer, "Irreversibility and heat generation in the computing process," *IBM journal of research and development*, vol. 5, no. 3, pp. 183–191, July 1961.

[16] J. Hall, "Nanocomputers and reversible logic," *Nanotechnology*, vol. 5, no. 3, pp. 157–167, 1994.

[17] R. Buchty, N. Heintze, and D. Oliva, *Cryptonite – A Programmable Crypto Processor Architecture for High-Bandwidth Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 184–198. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-24714-2_15

[18] V. Vedral, A. Barenco, and A. Ekert, "Quantum networks for elementary arithmetic operations," *Phys. Rev. A*, vol. 54, pp. 147–153, Jul 1996. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevA.54.147

[19] C. Bennett, "Logical reversibility of computation," *IBM journal of Research and Development*, vol. 17, no. 6, pp. 525–532, November 1973.

[20] N. Kostinski, M. P. Fok, and P. R. Prucnal, "Experimental demonstration of an all-optical fiber-based fredkin gate," *Opt. Lett.*, vol. 34, no. 18, pp. 2766–2768, Sep 2009. [Online]. Available: http://ol.osa.org/abstract.cfm?URI=ol-34-18-2766

[21] S. E. Frost-Murphy, M. Ottavi, M. P. Frank, and E. P. DeBenedictis, "On the design of reversible qdca systems."

[22] J. Ren, V. K. Semenov, Y. A. Polyakov, D. V. Averin, and J. S. Tsai, "Progress towards reversible computing with nsquid arrays," *IEEE Transactions on Applied Superconductivity*, vol. 19, no. 3, pp. 961–967, June 2009.

[23] J. Ren and V. K. Semenov, "Progress with physically and logically reversible superconducting digital circuits," *IEEE Transactions on Applied Superconductivity*, vol. 21, no. 3, pp. 780–786, June 2011.

[24] V. K. Semenov, G. V. Danilov, and D. V. Averin, "Classical and quantum operation modes of the reversible josephson-junction logic circuits," *IEEE Transactions on Applied Superconductivity*, vol. 17, no. 2, pp. 455–461, June 2007.

[25] S. Kim and S.-I. Chae, "Implementation of a simple 8-bit microprocessor with reversible energy recovery logic," in *Proceedings of the 2Nd Conference on Computing Frontiers*, ser. CF '05. New York, NY, USA: ACM, 2005, pp. 421–426. [Online]. Available: http://doi.acm.org/10.1145/1062261.1062332

[26] K. N. Patel, J. P. Hayes, and I. L. Markov, "Fault testing for reversible circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 8, pp. 1220–1230, Aug 2004.

[27] M. R. Betker, J. S. Fernando, and S. P. Whalen, "The history of the microprocessor," *Bell Labs Technical Journal*, vol. 2, no. 4, pp. 29–56, Autumn 1997.

[28] B. B. Brey, *The Intel Microprocessors*. Pearson Publication, 2008.

[29] D. V. Hall, *Microprocessors and Interfacing: Programming and Hardware*. Glencoe McGraw-Hill, 1991.

[30] *Intel Core i7-920 Processor*, 2016 (accessed February 6, 2017). [Online]. Available: http://ark.intel.com/products/37147/

[31] O. Lempel, "2nd generation intel core processor family: Intel core i7, i5 and i3," in *2011 IEEE Hot Chips 23 Symposium (HCS)*, Aug 2011, pp. 1–48.

[32] G. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, no. 8, pp. 56–59, April 1965.

[33] P. Kerntopf, "A new heuristic algorithm for reversible logic synthesis," in *Proceedings of the 41st annual Design Automation Conference*, pp. 834–837, 2004.

[34] D. Maslov and G. Dueck, "Garbage in reversible design of multiple output functions," in *6th International Symposium on Representations and Methodology of Future Computing Technologies*, pp. 162–170, 2003.

[35] R. Feynman, "Quantum mechanical computers," *Foundations of Physics*, vol. 16, no. 6, pp. 985–986, 1985.

[36] M. Mohammadi and M. Eshghi, "On figures of merit in reversible and quantum logic designs," *Quantum Information Processing*, vol. 8, no. 4, pp. 297–318, August 2009.

[37] *CMOS 45 nm Open Cell Library*, 2016 (accessed March 6, 2017). [Online]. Available: http://www.microwind.org

[38] *Microwind and dsch information page*, 2016 (accessed March 6, 2017). [Online]. Available: http://www.microwind.org

[39] M. Nielsen and I. Chuang, *Quantum computation and quantum information*. Cambridge university press, 2010.

[40] R. Wille and R. Drechsler, "Bdd-based synthesis of reversible logic for large functions," in *Proceedings of 46th Annual Design Automation Conference*, pp. 270–275, 2009.

[41] J. Smolin and D. DiVincenzo, "Five two-bit quantum gates are sufficient to implement the quantum fredkin gate," *Physical Review A*, vol. 53, no. 4, pp. 2855–2856, 1996.

[42] C. P. Williams, *Explorations in quantum computing*. Springer Science & Business Media, 2010.

[43] B. Parhami, "Fault tolerant reversible circuits," in *40th Asilomar Conference on Signals, Systems and Computers*, pp. 1726–1729, 2006.

[44] A. Mamun, M. Selim, M. Indrani, and M. Hasanuzzaman, "Design of universal shift register using reversible logic," *International Journal of Engineering and Technology*, vol. 2, no. 9, pp. 1620–1625, 2012.

[45] A. Peres, "Reversible logic and quantum computers," *Physical review A*, vol. 32, no. 6, pp. 3266–3276, 1985.

[46] W. Hung, X. Song, G. Yang, J. Yang, and M. Perkowski, "Optimal synthesis of multiple output boolean functions using a set of quantum gates by symbolic reachability analysis," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 25, no. 9, pp. 1652 –1663, sept. 2006.

[47] M. Haghparast and K. Navi, "A novel reversible bcd adder for nanotechnology based systems," *American Journal of Applied Sciences*, vol. 5, no. 3, pp. 282–288, 2008.

[48] A. Nagamani, H. Jayashree, and H. Bhagyalakshmi, "Novel low power comparator design using reversible logic gates," *Indian J Computer Science and Engineering*, vol. 2, pp. 576–574, 2011.

[49] M. S. Islam, M. M. Rahman, Z. Begum, and M. Z. Hafiz, "Fault tolerant reversible logic synthesis: Carry look-ahead and carry-skip adders," in *2009 International Conference on Advances in Computational Tools for Engineering Applications*, July 2009, pp. 396–401.

[50] H. Thapliyal and N. Ranganathan, "Design of efficient reversible binary subtractors based on a new reversible gate," in *Proceedings of IEEE Computer Society Annual Symposium on VLSI*, pp. 229–234, 2009.

[51] B. Sharath and K. Suhas, "A new approach to the design and implementation of multipliers using reversible logic," *International Journal of Engineering Science Invention*, vol. 2, no. 10, pp. 20–23, October 2013.

[52] H. Bhagyalakshmi and M. Venkatesha, "An improved design of a multiplier using reversible logic gates," *International journal of engineering science and technology*, vol. 2, no. 8, pp. 3838–3845, 2010.

[53] J. S. Hall, "An electroid switching model for reversible computer architectures," in *Workshop on Physics and Computation*, Oct 1992, pp. 237–247.

[54] R. Tocci, *Digital Systems: principles and applications*. Pearson Education India, 1980.

[55] A. S. M. Sayem and M. Ueda, "Optimization of reversible sequential circuits," *Journal of Computing*, vol. 2, no. 6, pp. 208–214, June 2010.

[56] H. Thapliyal and N. Ranganathan, "Design of reversible latches optimized for quantum cost, delay and garbage outputs," in *VLSID '10. 23rd International Conference on*, 2010, pp. 235–240.

[57] H. Thapliyal, M. Srinivas, and M. Zwolinski, "A beginning in the reversible logic synthesis of sequential circuits," in *Military and Aerospace Applications of Programmable Devices and Technologies International Conference (MAPLD)*, September 2005. [Online]. Available: http://eprints.soton.ac.uk/381081/

[58] M. S. Al Mamun, I. Manda, and M. Hasanuzzaman, "Design of universal shift register using reversible logic," *International Journal of Engineering and Technology*, vol. 2, no. 9, pp. 1620–1625, September 2012.

[59] H. Thapliyal and A. Vinod, "Design of reversible sequential elements with feasibility of transistor implementation," in *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, 2007, pp. 625–628.

[60] L. Jamal, F. Sharmin, M. A. Mottalib, and H. M. H. Babu, "Design and minimization of reversible circuits for a data acquisition and storage system," *International Journal of Engineering and Technology*, vol. 2, no. 1, pp. 9–15, January 2012.

[61] N. Nayeem, M. Hossain, L. Jamal, and H. Babu, "Efficient design of shift registers using reversible logic," in *2009 International Conference on Signal Processing Systems*, 2009, pp. 474–478.

[62] J. P. Hayes, *Computer Architecture and Organization; (2Nd Ed.)*. New York, NY, USA: McGraw-Hill, Inc., 1988.

[63] M. M. Mano, *Computer System Architecture (3rd Ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.

[64] C.-W. Kai and C.-C. Tseng, "Quantum plain and carry look-ahead adders," in *National Symposium on telecommunications*, vol. 12, 2002.

[65] H. Thapliyal and M. B. Srinivas, *A Novel Reversible TSG Gate and Its Application for Designing Reversible Carry Look-Ahead and Other Adder Architectures*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 805–817. [Online]. Available: http://dx.doi.org/10.1007/11572961_66

[66] M. Lu, *Arithmetic and Logic in Computer Systems*, 1st ed. New Jersey: Wiley-Interscience, 2004.

[67] H. Guild, "Some cellular logic arrays for non-restoring binary division," *Radio and Electronic Engineer*, vol. 39, no. 6, pp. 345–348, june 1970.

[68] J. Majithia, "Nonrestoring binary division using a cellular array," *Electronics Letters*, vol. 6, no. 10, pp. 303–304, 1970.

[69] M. Cappa and V. Hamacher, "An augmented iterative array for high-speed binary division," *Computers, IEEE Transactions on*, vol. C-22, no. 2, pp. 172–175, feb. 1973.

[70] N. Nayeem, A. Hossain, M. Haque, L. Jamal, and H. Babu, "Novel reversible division hardware," in *Circuits and Systems, 2009. MWSCAS '09. 52nd IEEE International Midwest Symposium on*, 2009, pp. 1134 –1138.

[71] F. Dastan, , and M. Haghparast, "A novel nanometric fault tolerant reversible divider," *International Journal of the Physical Sciences*, vol. 6(24), pp. 5671–5681, 2011.

[72] F. Dastan and M. Haghparast, "A novel nanometric reversible signed divider with overflow checking capability," *Research Journal of Applied Sciences, Engineering and Technology*, vol. 4(6), pp. 535–543, 2012.

[73] M. L. Mui, K. Banerjee, and A. Mehrotra, "A global interconnect optimization scheme for nanometer scale vlsi with implications for latency, bandwidth, and power dissipation," *IEEE Transactions on Electron Devices*, vol. 51, no. 2, pp. 195–203, Feb 2004.

[74] H. Rangaraju, V. Hegde, K. B. Raja, and K. N. Muralidhara, "Design of efficient reversible binary comparator," *Procedia Engineering*, vol. 30, pp. 897 – 904, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877705812009538

[75] H. Thapliyal, N. Ranganathan, and R. Ferreira, "Design of a comparator tree based on reversible logic," in *10th IEEE International Conference on Nanotechnology*, Aug 2010, pp. 1113–1116.

[76] C. Vudadha, P. S. Phaneendra, V. Sreehari, S. E. Ahmed, N. M. Muthukrishnan, and M. B. Srinivas, "Design of prefix-based optimal reversible comparator," in *2012 IEEE Computer Society Annual Symposium on VLSI*, Aug 2012, pp. 201–206.

[77] M. Morrison, M. Lewandowski, and N. Ranganathan, "Design of a tree-based comparator and memory unit based on a novel reversible logic structure," in *2012 IEEE Computer Society Annual Symposium on VLSI*, Aug 2012, pp. 231–236.

[78] A. S. Sayem, M. M. A. Polash, and H. M. Hasan Babu, "Design of a reversible logic block of field programmable gate array," in *Silver Jubilee Conference on Communication Technologies and VLSI design (CommV '09), VIT University, Vellore, India.*, 2009, pp. 500–501.

[79] M. M. A. Polash and S. Sultana, "Design of a lut-based reversible field programmable gate array," *Journal of Computing*, vol. 2, pp. 103–108, October 2010. [Online]. Available: http://www.scribd.com/doc/43093238

[80] L. Jamal, M. M. Rahman, and H. M. H. Babu, "An optimal design of a fault tolerant reversible multiplier," in *2013 IEEE International SOC Conference*, Sept 2013, pp. 37–42.

[81] S. Nowrin, L. Jamal, and H. M. H. Babu, "Design of an optimized reversible bidirectional barrel shifter," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2016, pp. 730–733.

[82] N. Huda, S. Anwar, L. Jamal, and H. Babu, "Design of a reversible random access memory," *Dhaka University Journal of Applied Science and Engineering*, vol. 2, no. 1, pp. 31–38, 2011.

[83] M. Shamsujjoha and H. M. H. Babu, "A low power fault tolerant reversible decoder using mos transistors," in *2013 26th International Conference on VLSI Design and 2013 12th International Conference on Embedded Systems*, Jan 2013, pp. 368–373.

[84] M. Nachtigal and N. Ranganathan, "Design and analysis of a novel reversible encoder/decoder," in *2011 11th IEEE International Conference on Nanotechnology*, Aug 2011, pp. 1543–1546.

[85] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007.

[86] J. R. Burch and D. E. Long, "Efficient boolean function matching," in *Proceedings of the 1992 IEEE/ACM International Conference on Computer-aided Design*, ser. ICCAD '92. Los Alamitos, CA, USA: IEEE Computer Society Press, 1992, pp. 408–411. [Online]. Available: http://dl.acm.org/citation.cfm?id=304032.304143

[87] T. Lengauer, *Combinatorial algorithms for integrated circuit layout*. Springer Science & Business Media, 2012.

[88] W. Shi, "A fast algorithm for area minimization of slicing floorplans," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 12, pp. 1525–1532, Dec 1996.

[89] J. J. P. Kocher and B. Jun, "Differential power analysis," in *Annual International Cryptology Conference*, 1999, pp. 388–397.

[90] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power cmos digital design," *IEICE Transactions on Electronics*, vol. 75, no. 4, pp. 371–382, 1992.

[91] S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic, *Field-programmable gate arrays*. Springer Science & Business Media, 2012, vol. 180.

[92] R. Cuykendall and D. Andersen, "Reversible optical computing circuits," *Optics Letters*, vol. 12, no. 7, pp. 542–544, 1987.

[93] A. Sarker, T. Ahmed, S. M. M. Rashid, S. Anwar, L. Jaman, N. Tara, M. M. Alam, and H. M. H. Babu, "Realization of reversible logic in dna computing," in *2011 IEEE 11th International Conference on Bioinformatics and Bioengineering*, Oct 2011, pp. 261–265.

[94] L. Bogani, "Quantum nanoelectronicsan introduction to electronic nanotechnology and quantum computing. by el wolf." 2010.

Appendices

Appendix A: List of Acronyms

CPU	**C**entral **P**rocessing **U**nit
ALU	**A**rithmetic **L**ogic **U**nit
CU	**C**ontrol **U**nit
RCPU	**R**eversible **C**entral **P**rocessing **U**nit
RALU	**R**eversible **A**rithmetic **L**ogic **U**nit
RCU	**R**eversible **C**ontrol **U**nit
VLSI	**V**ery **L**arge **S**cale **I**ntegration
CMOS	**C**omplementary **M**etal **O**xide **S**emiconductor
QCA	**Q**uantum Dot **C**ellular **A**utomata
DNA	**D**eoxyribo **N**ucleic **A**cid
SIMD	**S**ingle **I**nstruction **M**ultiple **D**ata
SDRAM	**S**ynchronous **D**ynamic **R**andom **A**ccess **M**emory
QC	**Q**uantum **C**ost
CLA	**C**arry **L**ook-aheadt **A**dder
CNOT	**C**ontrolled **NOT** Gate
PPG	**P**artial **P**roduct **G**eneration
MOA	**M**ulti **O**perand **A**ddition
HNG	**H**aghparast **N**avi **G**ate
PG	**P**eres **G**ate
FG	**F**eynman **G**ate
F2G	**F**eynman **D**ouble **G**ate
FRG	**FR**edkin **G**ate
MFG	**M**odified **F**redkin **G**ate
BJN	**B**hagyalakshmi **J**ayashree **N**agamoni
TR	**T**hapliyal **R**anganathan

Appendix B: List of Publications

PhD Competition Paper:

1. Lafifa Jamal and Hafiz Md. Hasan Babu, "Design and Implementation of a Reversible Central Processing Unit", *IEEE Computer Society Annual Symposium on VLSI (ISVLSI 2015)*, Montpellier, France, pp. 187-190, July 2015.

International Journal Papers:

2. Lafifa Jamal, Hafiz Md. Hasan Babu and Sadia Nowrin, "Design of a Reversible Barrel Shifter", *Elsevier Journal of Microelectronics* (The revision has been submitted after the reviewers' comment).

3. Lafifa Jamal, Hafiz Md. Hasan Babu and Md. Masbaul Alam, "An Efficient Approach to Design a Reversible Control Unit of a Processor", *Elsevier Journal of Sustainable Computing: Informatics and Systems*, Vol. 3, Issue 4, pp. 286-294, December 2013, ISSN 2210-5379.

4. Hafiz Md. Hasan Babu, Nazir Saleheen, Lafifa Jamal, Sheikh Muhammad Sarwar and Tsutomu Sasao, "Approach to Design a Compact Reversible Low Power Binary Comparator", *IET Computers & Digital Techniques*, Vol. 8, Issue 3, pp. 129-139, May 2014, ISSN 1751-8601.

5. Lafifa Jamal, Md. Masbaul Alam Polash, M. A. Mottalib and Hafiz Md. Hasan Babu, "On the Compact Designs of Low Power Reversible Decoders and Sequential Circuits", *16th International Symposium on VLSI Design and Test (VDAT 2012)*, Shibpur, India, July 2012, Published in Lecture Notes in Computer Science, Springer Berlin / Heidelberg, vol. 7373, pp. 281-288, 2012.

6. Lafifa Jamal, Md. Shamsujjoha and Hafiz Md. Hasan Babu, "Design of Optimal Reversible Carry Look-Ahead Adder with Optimal Garbage and Quantum Cost", *The International Journal of Engineering and Technology*, Vol. 2, No. 1, pp. 44-50, January 2012, ISSN 2049-3444.

7. Md. Shamsujjoha, Hafiz Md. Hasan Babu and Lafifa Jamal, "Design of a Compact Reversible Fault Tolerant Field Programmable Gate Array: A Novel Approach in Reversible Logic Synthesis", *Elsevier Journal of Microelectronics*, Vol. 44, Issue 6, pp. 519-537, June 2013, ISSN 0026-2692.

International Conference Papers:

8. Lafifa Jamal and Hafiz Md. Hasan Babu, "Efficient Approaches to Design a Reversible Floating Point Divider", *The IEEE International Symposium on Circuits and Systems (ISCAS 2013)*, Beijing, China, pp. 3004-3007, May 2013.

9. Lafifa Jamal, Md. Mushfiqur Rahman and Hafiz Md. Hasan Babu, "An Optimal Design of a Fault Tolerant Reversible Multiplier", *26th IEEE International System-on-Chip Conference (SOCC 2013)*, Erlangen, Germany, pp. 37-42, September 2013.

10. Hafiz Md. Hasan Babu, Lafifa Jamal and Nazir Saleheen, "An Efficient Approach for Designing a Reversible Fault Tolerant n-Bit Carry Look-Ahead Adder", *26th IEEE International System-on-Chip Conference (SOCC 2013)*, Erlangen,

Germany, pp. 98-103, September 2013.

11. Hafiz Md. Hasan Babu, Md. Shamsujjoha, Lafifa Jamal and Ahsan Raja Chowdhury, "Design of a Fault Tolerant Reversible Compact Unidirectional Barrel Shifter", *26th International Conference on VLSI Design (VLSID 2013)*, Pune, India, pp. 103-108, January 2013.

12. Sadia Nowrin, Lafifa Jamal and Hafiz Md. Hasan Babu, "Design of an Optimized Reversible Bidirectional Barrel Shifter", *International Symposium on Circuit and Systems (ISCAS 2016)*, Montreal, Canada, pp. 730-733, May 2016.

13. Sayanton Vhaduri Dibbo, Hafiz Md. Hasan Babu and Lafifa Jamal, "An Efficient Design Technique of a Quantum Divider Circuit", *International Symposium on Circuit and Systems (ISCAS 2016)*, Montreal, Canada, pp. 2102-2105, May 2016.

14. Sadia Nowrin, Lafifa Jamal and Humayra Tasnim, 'An Efficient Approach to Design A Reversible Signed Multiplier", *TENCON 2014 - 2014 IEEE Region 10 Conference*, Bangkok, Thailand, pp. 1-6, October 2014.