An FPGA-based Emulation Process for Dynamic Quantum Circuits

Yicheng Song

Department of Electrical & Computer Engineering

McGill University

Montréal, Québec, Canada

May 24, 2024

A thesis presented for the degree of Masters of Electrical Engineering

Abstract

The current quantum computation on real, physical devices has predominantly been constrained to basic, time-ordered sequences of unitary quantum operations culminating in a final projective measurement. As quantum computing hardware evolves in scale and functionality, it becomes crucial to facilitate the construction of quantum circuits beyond their traditional confines. Recent progress in quantum hardware has brought about mid-circuit measurements and resets, allowing for the recycling of measured qubits and notably decreasing the qubit demands for running quantum algorithms.

In this thesis, we propose an FPGA-based dynamic quantum circuit emulation process that integrates quantum bit storage, quantum validation checking, quantum operation processing, quantum state measurement and probabilistic execution prediction to provide an emulation platform for designing and verifying dynamic quantum circuits.

Each functional block in the proposed emulation process was analyzed and evaluated using the Vivado environment and programmed onto the Digilent Cmod A7-35T FPGA board. The quantum validation checking process blocked all the invalid qubits and achieved

a 99.98% pass rate for valid qubits with the suitable threshold set. The ring-oscillator-based true random number generator used for the quantum state measurement process provides a 99.986% of 0-1 ratio. The testing results of the probabilistic execution predictor show that the average miss rate is as low as 25%, while the time saved by the process depends on the specific emulated quantum circuits. A case study is provided to present the comprehensive workflow of the emulation.

Abrégé

Le calcul quantique actuel sur des dispositifs physiques réels a été principalement limité à des séquences de base ordonnées dans le temps d'opérations quantiques unitaires aboutissant à une mesure projective finale. À mesure que le matériel informatique quantique évolue en termes d'échelle et de fonctionnalités, il devient crucial de faciliter la construction de circuits quantiques au-delà de leurs limites traditionnelles. Les progrès récents dans le domaine du matériel quantique ont permis des mesures et des réinitialisations à mi-circuit, permettant le recyclage des qubits mesurés et réduisant considérablement les demandes de qubits pour l'exécution d'algorithmes quantiques.

Dans cette thèse, nous avons proposé un processus d'émulation de circuits quantiques dynamiques basé sur FPGA qui intègre le stockage de bits quantiques, la vérification de validation quantique, le traitement des opérations quantiques, la mesure d'état quantique et la prédiction d'exécution probabiliste pour fournir une plate-forme d'émulation pour la conception et la vérification de circuits quantiques dynamiques.

Chaque bloc fonctionnel du processus d'émulation proposé a été analysé et évalué à

l'aide de l'environnement Vivado et programmé sur la carte FPGA Digilent Cmod A7-35T. Le processus de vérification de validation quantique a bloqué tous les qubits invalides et a atteint un taux de réussite de 99,98% pour les qubits valides avec le seuil approprié défini. Le générateur de nombres aléatoires réels basé sur un oscillateur en anneau utilisé pour le processus de mesure de l'état quantique fournit un rapport 0-1 de 99,986%. Les résultats des tests du prédicteur d'exécution probabiliste montrent que le taux d'échec moyen est aussi faible que 25%, tandis que le temps gagné par le processus dépend des circuits quantiques émulés spécifiques. Une étude de cas est fournie pour présenter le flux de travail complet de l'émulation.

Acknowledgements

I extend my deepest gratitude to my supervisor, Zeljko Zilic, for his invaluable guidance, support, and mentorship throughout this journey. His expertise, encouragement, and unwavering commitment have been instrumental in shaping this thesis.

I am profoundly thankful to my parents for their endless love, encouragement, and sacrifices. Their unwavering belief in my abilities has been my source of strength and motivation.

I am also indebted to my friends Xiangyun (Alfred) Wang and Purui (Raymond) Chen for their unwavering support, understanding, and encouragement during this challenging yet rewarding academic endeavour. Their presence has made this journey not only academically enriching but also enjoyable.

Finally, I express my gratitude to all those who have supported me in various ways, directly or indirectly, in the completion of this thesis.

Contents

List of Figures

List of Tables

List of Acronyms

CNOT	Controlled NOT gate.
DFF	D-type Flip Flop.
DSP	Digital Signal Processor.
FPGA	Field Programmable Gate Array.
LFSR	Linear Feedback Shift Register.
LSB	least significant bit.
MSB	most significant bit.
Qubit	Quantum bit.
RAM	Random Access Memory.
RNG	random number generator.
RO	ring oscillator.
UART	Universal Asynchronous Receiver-Transmitter.

Chapter 1

Introduction

1.1 Motivation

Quantum computing has arisen as a promising path for solving tricky problems that classical computing cannot address, such as integer factoring [3], chemistry simulation [4], large database search [5] and machine learning [6]. Many quantum algorithms outperform their classical counterparts through parallelism which is impossible in classical computing, some have been successfully used in practical applications such as data encryption and communication [7] [8] [9].

With IBM introduced the first circuit-based commercial quantum computer, IBM Quantum System One, in January 2019, the expectation to solve computationally expensive problems seemed to come true. Nevertheless, the expense of constructing a real

quantum computer remains prohibitive for most research institutions, presenting a formidable challenge due to the exceedingly harsh operating conditions. The lack of fully functional quantum computers is thus impeding the implementation of quantum algorithms on a practical scale. This research gap has shifted attention toward the emulation of quantum computing based on the classical computing method. With considerable effort dedicated, various hardware and software approaches have been developed to simulate or emulate the quantum circuits, such as [10], [11], [12], [13] and [14].

While most of the current research on quantum circuit emulation focused on static quantum circuits, the concept of dynamic quantum circuits was brought to the public eye. Numerous quantum algorithms have conventionally been formulated as static quantum circuits, where the computations operate on an initially prepared quantum state, and all measurements are executed at the end of the circuit to derive the computational outcomes. However, recent advancements in the research of quantum hardware have opened a door for a more flexible approach, enabling measurements and qubit resets to be performed midway through a quantum circuit. These circuits permit the real-time evolution of the quantum circuit based on prior measurement outcomes [15] [16] [17]. This emerging paradigm of quantum computation, distinguished by its ability to adapt the circuit dynamically, is termed a dynamic quantum circuit. It plays a pivotal role in exploring quantum error correction [18] and measurement-based quantum computation [19].

This work aims to investigate the emulation of dynamic quantum circuits on FPGA

devices, where parallelism is leveraged to provide emulation in a time-efficient manner. It is motivated by the need to augment the existing quantum circuit emulation approach that is centred on static quantum circuits.

1.2 Contribution to Knowledge

This thesis introduces an FPGA-based emulation process for the dynamic quantum circuit, including quantum bit storage, quantum gate operations, quantum state measurement and probabilistic execution prediction. The objective of this research is to provide an FPGA-based emulation platform that could be used for the design and verification of dynamic quantum circuits. The contribution to the existing body of knowledge occurs through the following ways:

- Presented a comprehensive emulation process of the dynamic quantum circuit.
- Implemented a quantum state measurement process based on a true random number generator. Compare the true random number generator with the pseudo-random number generator in quantum state measurement.
- Proposed a probabilistic execution prediction process that saves processing time when the midway measurement is being executed.

1.3 Document Structure

The structure of this thesis is composed of five chapters. Chapter 2 introduces the background knowledge and examines the literature on the strategies adopted in this work. Chapter 3 describes the implementation details of the proposed dynamic quantum circuit emulation process. Chapter 4 reports and discusses the experimental results of each functional block. The conclusions and future work are presented in Chapter 5.

Chapter 2

Background and Literature Review

This chapter presents the definitions and concepts in quantum computing necessary to understand the details of the investigation. A comprehensive review of the existing work is presented in the second section.

2.1 Quantum Computing Basics

2.1.1 Quantum Information and Quantum Bits

In conventional computing, the bit is the smallest unit of information describing a classical system where each bit represents a single value of either 0 or 1. The manipulation and combination of such binary bits is the deterministic information process. Probabilistic computation introduces probabilistic bits, which return 0 or 1 with probabilities P_0 and P_1,

respectively. The quantum bit, or qubit, is the basic unit of information in a quantum system utilizing the superposition and the entanglement phenomenon held by quantum physics.

The superposition indicates that a quantum bit could be represented as a mixture of two basis states. The Stern–Gerlach experiment defines the concept of a qubit with the help of the electron spin [20]. The mathematical model of the experiment states that an electron exists in a superposition and that the probability of an electron following the up or down path is $|\alpha_0|^2$ and $|\alpha_1|^2$, respectively. An electron spin is then described as

$$spin(e) = \alpha_0 |\uparrow\rangle + \alpha_1 |\downarrow\rangle \tag{2.1}$$

In quantum computing, the spin directions are replaced by $|0\rangle$ and $|1\rangle$ to denote the basis state. The conventional state analogs are represented by a 2×1 matrix shown in 2.2. The superposition allows the qubit to be $|0\rangle$ and $|1\rangle$ simultaneously, but only one of these values will be returned when the qubit is measured.

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{2.2}$$

A qubit is mathematically defined as a combination of orthogonal states, shown in 2.3, where α and β are complex numbers, $|\alpha|^2$ expresses the probability of a qubit landing as $|0\rangle$ and $|\beta|^2$ represents the probability of being $|1\rangle$. The probability of being $|0\rangle$ or $|1\rangle$ perfectly

satisfies the concept of probabilistic computation, which can be viewed as manipulating those probabilities for every output bit.

$$|\psi\rangle = \begin{bmatrix} \alpha \\ \\ \beta \end{bmatrix} = \alpha \begin{bmatrix} 1 \\ \\ 0 \end{bmatrix} + \beta \begin{bmatrix} 0 \\ \\ 1 \end{bmatrix} \tag{2.3}$$

2.1.2 Quantum Circuit Model

Information processing can be viewed as a series of operations performed by a set of gates on a group of parallel binary inputs, flowing through wires. A quantum circuit is a computational routine consisting of coherent quantum operations on quantum data, such as qubits, and concurrent real-time classical computation. A quantum circuit model is a mathematical and visual model used in quantum computing to represent and perform quantum computations. It is analogous to classical digital circuits used in classical computing. A quantum circuit consists of quantum gates operating on one or more qubits, which envision quantum algorithms as a series of unitary transformations on the quantum state register.

A quantum circuit is static if it does not contain mid-circuit reset operations and all measurements are performed at the end of the circuit, as shown in Figure 2.1.

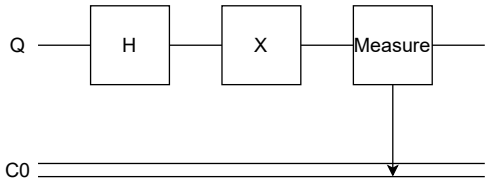

Figure 2.1: A static quantum circuit sample.

A quantum circuit is dynamic if it contains mid-circuit measurements and reset operations, as shown in Figure 2.2.

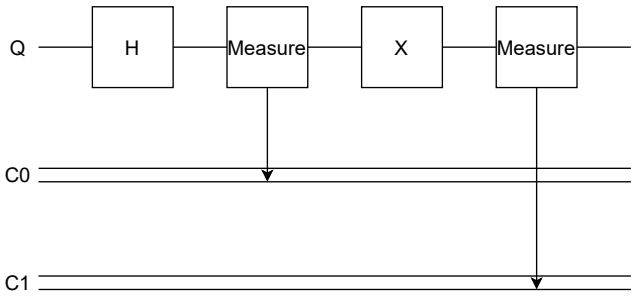

Figure 2.2: A dynamic quantum circuit sample.

Since the quantum gates represent unitary operations, the reversibility required by quantum computing, which requires the determination of input through output, is naturally fulfilled. In mathematical notation, we may represent a quantum state with a

vector of complex values and quantum gates as unitary matrices. Therefore, the computation may be viewed as a series of matrix-vector multiplications.

It should be noted that the measurement operations also change the state of the quantum bits that are gauged. After being measured, the state of the quantum bit will collapse, and the superposition of the basis states will vanish [21]. Approaches are developed to help with qubit measurement reversal through entanglement [22], which is similar to generating a hard copy of the pre-measurement qubit states. In contrast to the quantum gate operations, the measurement operation is irreversible, and the information stored will be eliminated.

2.2 Literature Review

There are various existing approaches to emulate or simulate the quantum computer. Most of them focused on reducing the time consumed by simulating the quantum algorithm based on classical computing architecture with the help of GPU processors [23] [24] or multiprocessor systems [25]. However, as the number of quantum bits required by the quantum algorithms grows exponentially, resource management and parallelism in quantum computing have become an issue for those solutions [26].

FPGA technology provides an attractive opportunity to leverage its massive parallelism to completely emulate the time-speedup of quantum machines. There are many proposed emulator architectures based on the FPGA platform, but all of them are only focused on the emulation of static quantum circuits and ignore the dynamic quantum circuit.

Khalid et al. [27] proposed a VHDL library of quantum gate primitives, which allows the simple construction of static quantum circuits using the primary blocks from the component library. At the same time, it emulates the parallelism present in quantum computing by constructing parallel paths for each quantum bit on the FPGA. The quantum circuit to be emulated must be known before synthesis so that the synthesis software will take responsibility for arranging the hardware resources. However, this solution only provides the method of building a quantum circuit based on quantum gates, where the emulation of quantum measurement is missing.

Goto and Fujishima [28] designed a solution based on unitary macro-operations, allowing memory-efficient simulation of quantum circuits on FPGA. These operations, which are sequences of elementary quantum gates, are decomposed in software to pre-designed controlled-NOT represented by hardware assembler instructions. By optimizing the storage and computation of the instruction matrices, the emulation system achieves better performance compared to traditional approaches. The emulator proposed consists of hardware (an instruction processor written into an FPGA) and software (a C++ program converts macro-operation to emulator instruction), accelerating the process of emulating quantum algorithms.

As the above two approaches proved the feasibility of implementing the quantum computing emulator, the exponential growth of space requirement and time complexity for larger quantum circuits arose. Lee et al. [29] took an innovative approach of mixing parallel

and serial processing on FPGA, which allowed them to achieve desired speedups without hardware resources' exponential growth. While this design achieves a linear reduction in resource utilization compared to previous works and provides a process for quantum algorithm analysis, the cost was the ignoring of parallelized behaviours of quantum physics in hardware.

Aminian et al. [30] described a universal and efficient method of emulating quantum circuits on FPGAs. The authors proposed an efficient way to emulate a universal set of quantum gates on FPGA hardware by further decomposing the basic quantum gate into the basis and sign operations. Multiple algorithms constructed from gates in the set are tested with the performance analyzed. While this approach uses fewer logic cells for implementing a quantum circuit compared to the work previously described, it does not allow the mixed usage between the direct usage of any desired quantum gate and the gate operation they proposed.

Negovetic et al. [31] proposed a software-hardware system for emulating static quantum circuits on FPGAs. They presented an approach where a software preprocessor converts quantum netlists into HDL that can then be synthesized and run on FPGA. The emulation of the observation process was proposed with the introduction of the pseudo-random number generator, where a RAM with 8 addresses is required. While the solution utilizes the FPGA parallelism to speed up the emulation of quantum computation, it requires re-synthesizing of hardware each time emulating a quantum circuit.

Khalid and Mujahid et al. [1] present an architecture for hardware abstraction of single-input quantum systems, where the proposed abstraction is to provide an FPGA-based platform as the fundamental subblock for designing quantum circuits. They proposed a complete emulation process for a single qubit static quantum circuit, including qubit storage, arithmetic logic unit and quantum state measurement. The emulation of measurement is based on the permutation-based shuffling (PbS) function used for a pseudo-random number generator. While the PbS perfectly satisfies the function of a pseudo-random number generator, the hardware resources it requires are tremendously high.

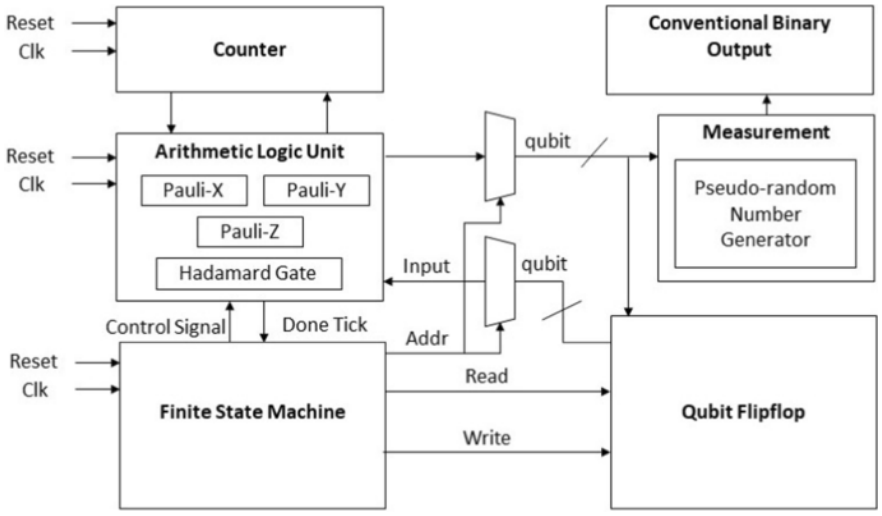

Figure 2.3: The SISO quantum circuit emulator proposed by Khalid and Mujahid et al. [1].

Pilch and Długopolski [2] proposed a generalized programmable quantum computer emulator, including a computing core, a communication module and a processor. The emulator combined software with hardware implementation, where some software was

introduced to compile the quantum circuit to instructions and send it to the hardware, and the FPGA was only used to deal with matrix-vector multiplications, where the quantum information held by the quantum bits is stored in a large-scale matrix. While the complexity and the importance of quantum state measurement are especially mentioned, the pseudo-random number generator used was not specified.

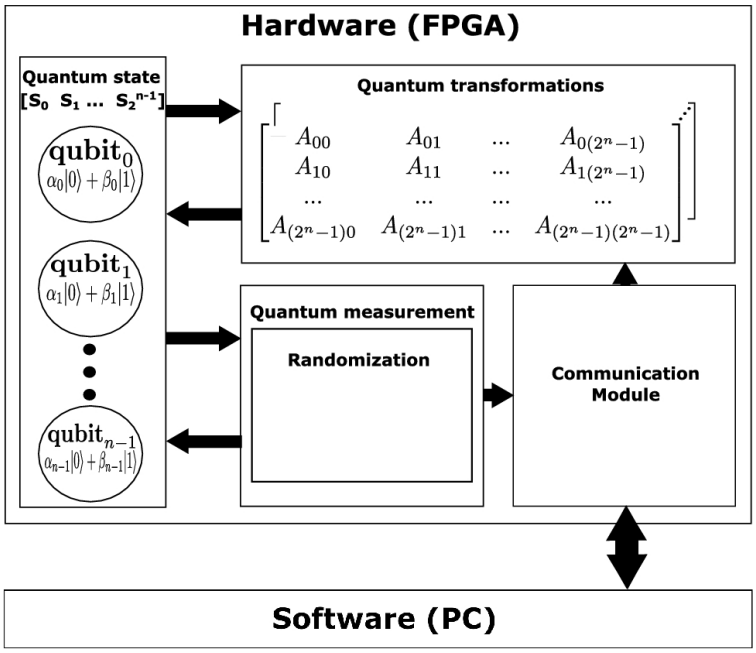

Figure 2.4: The quantum system overview proposed by Pilch and Długopolski [2].

In general, the existing approaches of quantum circuit emulation could be divided into two main categories:

- Emulating static quantum circuits using a pre-built set of blocks focused on efficient time or hardware usage. The proposed designs include HDL libraries and the quantum processor based on classical architecture.

- Emulating the natural behaviours of physical quantum circuits such as the parallelism of quantum computing and measurement of quantum bits. Those solutions were mostly focused on reflecting the quantum phenomenon of a selected group of synthesized circuits, built from tools provided in HDL libraries, rather than constructing a processing unit with a universal set of instructions.

Córcoles et al. [15] proposed a protocol that focuses on the synergy between classical and quantum hardware in complex dynamic circuits. As quantum systems get increasingly accurate, longer-lived, and faster queried, it is important to consider pathways for the processing of their classical outputs that do not limit the capability of the quantum system to compute, neither in time nor in breadth of resources. By implementing the quantum phase estimation algorithm [32] [33] with dynamic quantum circuits, the number of quantum resources used has economized and the need for classical postprocessing is eliminated. The experiments presented show that quantum computing hardware has reached a level of maturity where it can benefit from dynamic circuits.

While the approaches to emulate static quantum circuits are mature, there is hardly any effort paid to emulating dynamic quantum circuits. Recent progress in quantum hardware has brought about mid-circuit measurements and resets, which allow for the recycling of measured qubits and substantially reduce the number of qubits needed to execute quantum algorithms. Fang et al. [34] noticed the significance of dynamic quantum circuits and presented a systematic study of dynamic quantum circuit compilation. With

the application of graph representation of the quantum circuit, they proposed a process that converts the static quantum circuit composition to instructions and compiles them, which would be used to generate a possible equivalent dynamic quantum circuit. In contrast to the static quantum circuit, dynamic circuit compilation is centered around the reordering of instructions and the reassignment of logical qubits, while preserving both the number and type of circuit instructions. This approach demonstrates the advantages of dynamic quantum circuits and bridging the gap between theoretical quantum algorithms and their physical implementation.

Chapter 3

Proposed Methodology

This chapter outlined a detailed account of the methodology and implementation employed in emulating dynamic quantum circuits with probabilistic execution prediction on FPGA is outlined below.

- **Emulation Process Overview:** This section presents the complete architecture of the dynamic quantum circuits emulation process, composed of 4 primary components: validation check of quantum states, quantum gate operations, quantum state measurement, and probabilistic execution predictor.

- **Quantum State and Validation Check:** In this section, we demonstrate the implementation of quantum state registers that store the states of qubits. Moreover, the quantum state validation methodology and the validation criteria are presented.

- **Quantum Gate Operations:** In this section, we describe the implementation of several quantum gates, including the Pauli-X gate, the Pauli-Y gate, the Pauli-Z gate, the Hadamard gate and the CNOT gate.

- **Quantum State Measurement:** This section explores the architecture of the quantum state measurement block. We describe the quantum state measurement methodology based on the random number generator for both a single qubit and multiple qubits. The implementation of an LFSR-based pseudo-random number generator and a ring-oscillator-based true random number generator is presented.

- **Probabilistic Execution Predictor:** This section presents the architecture of the probabilistic execution predictor block. We describe the process flow of jumping midway measurement and how to deal with branch misprediction. Temporary state registers are introduced to serve an essential function by holding intermediate quantum states. These states can be leveraged to overwrite misprediction results.

3.1 Emulation Process Overview

An elaborated overview of the proposed emulation process is shown in Fig. 3.1. The quantum state registers hold the information of Qubit. They will be accessed and overwritten every time a quantum gate operation, a probabilistic execution prediction, or a quantum state measurement is performed. For the first time a Qubit is loaded, the validation check of

Qubits will be activated to make sure the Qubit is in the pure state. Once the validation check is passed, the following quantum gate operation can be executed on the qubit.

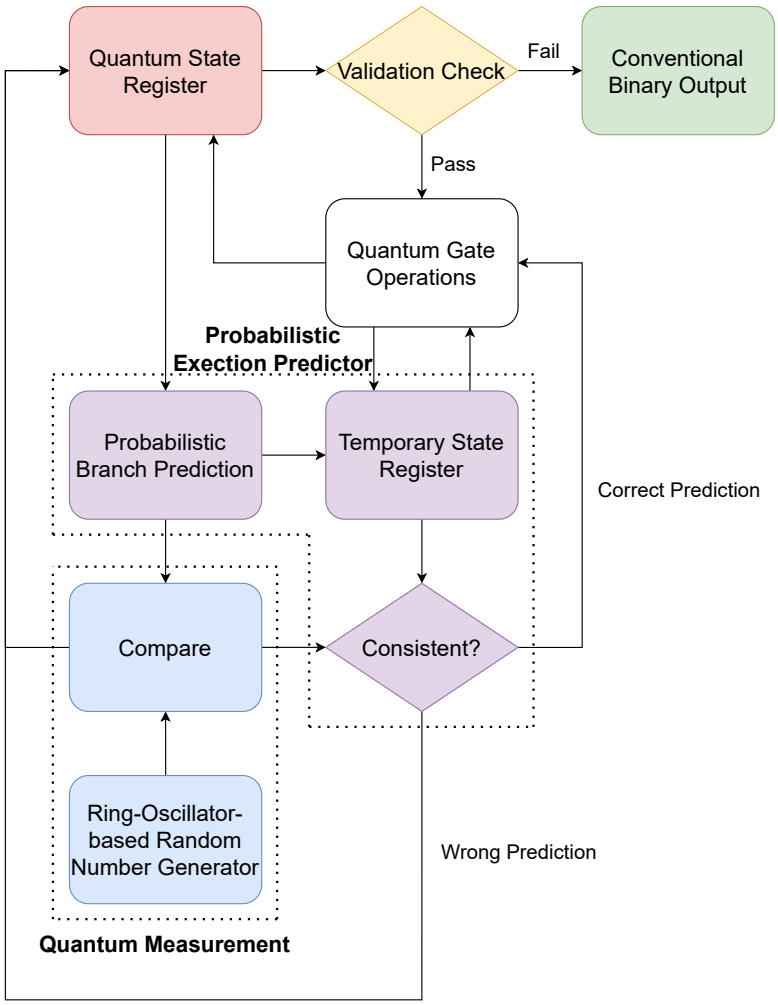

Figure 3.1: Process flow of the dynamic quantum circuit emulation with probabilistic execution prediction.

In the practical dynamic quantum circuit, the quantum gate operations to be performed depend on the midway quantum state measurement result, similar to the conditional statement in the traditional digital circuit. In the proposed emulation process, when encountering a midway quantum state measurement, the current qubit information that is being processed will be stored in the temporary state registers, and the probabilistic execution predictor unit will automatically choose the conditional statement with the higher probability, with the following quantum gate operations being executed using the quantum information stored in the temporary state register. At the same time, the quantum measurement unit will measure the qubit and check whether the measured result is consistent with the prediction. If the measured result is the same as predicted, the temporary state will overwrite the corresponding qubit in the quantum state register. Otherwise, the temporary state will be eliminated, and the correct branch will be executed instead.

A similar emulation process without probabilistic execution prediction is also introduced to compare with the process proposed above, as shown in Fig. 3.2. The whole process would stall whenever a measurement appears until the measurement is done.

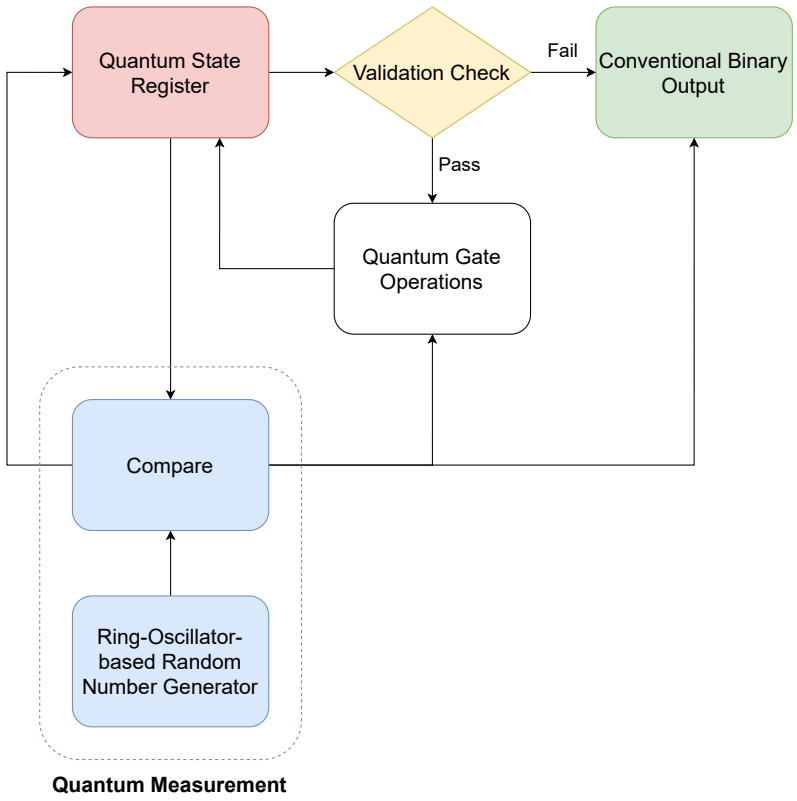

Figure 3.2: Process flow of the dynamic quantum circuit emulation.

3.2 Quantum State Representation

Binary qubits have two computational base states denoted as $|0\rangle$ and $|1\rangle$. Unlike classical bits, quantum bits are in a superposition of the basis states, represented as Equation 3.1.

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{3.1}$$

Quantum state registers store α and β of all qubits and fetch or store the exact data when operations are performed on a specific qubit. Figure 3.3 demonstrates the architecture of a quantum state register. Considering that the coefficients are complex numbers, the real and imaginary parts of each coefficient are stored in two separate signed binaries.

We decided to choose 16-bit signed binaries to represent each coefficient after taking both the accuracy and resources on FPGA into consideration. The conventional signed binary representation is used instead of a fixed-point representation since the real and imaginary parts of all coefficients are constrained between -1 and 1. As a result, the accuracy of these coefficients is directly related to the length of the binaries. A 16-bit signed binary could be used to represent from -32768 to 32767. The fraction number represented by the 16-bit binary is calculated by dividing the signed integer by 32768.

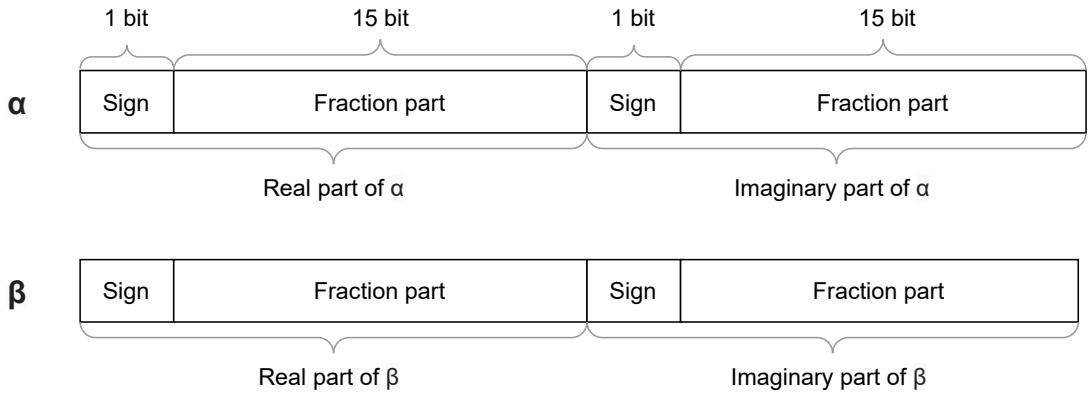

Figure 3.3: The architecture of a quantum state register

Quantum gate operations also affect the accuracy of qubits. Some quantum gates involve

signed multiplication and addition when processing the qubit, which would generate errors. When performing the multiplication of two binaries, the length of the result would be the sum of the length of inputs. To prevent the exponentially increasing number of bits, the result would be truncated and only the MSBs will be preserved. That is to say, the output of some quantum gates is less accurate, especially when a shorter input is applied.

3.3 Validation Check

The quantum state coefficients α and β are complex numbers subject to the condition shown in Equation 3.2. Qubits that meet the requirement are in a pure state; otherwise, they are in a mixed state. Before performing operations on the qubit, it is necessary to ensure that the coefficients of the qubit meet the requirement.

$$|\alpha|^2 + |\beta|^2 = 1 \tag{3.2}$$

After being loaded, the squares of both real and imaginary parts of each coefficient would be calculated and summed up. As mentioned in section 3.2, multiplying and truncating will introduce errors to the result, which implies that the sum of squares of all coefficients of a valid qubit may not be exactly 1 but close to 1. Fig. 3.4 shows the data flow of the validation block. If the sum of the square of the coefficients falls into the range from hex number 3ffe9c88 to hex number 40000000, the qubit is treated as valid and will be transmitted to the next

step, and the validation flag is raised as pass. Otherwise, the validation flag is raised as a failure, and the emulation process will terminate.

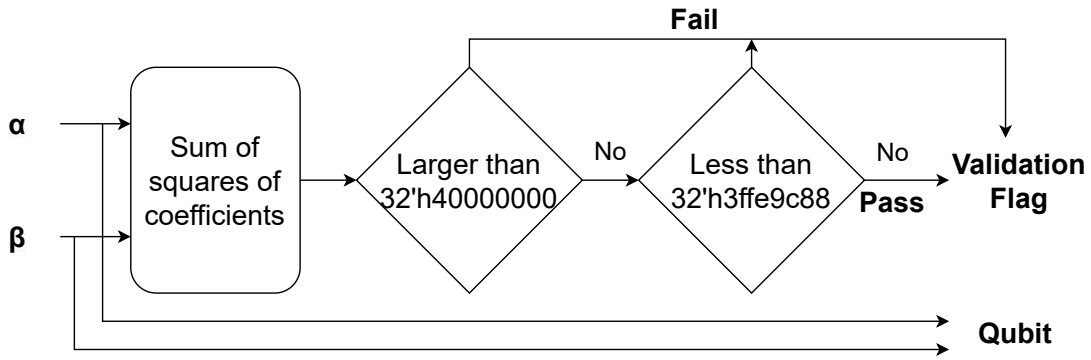

Figure 3.4: The data flow of validation check block.

After generating 1000 random numbers from 0 to 1 as a set of magnitude of α, we could easily get their corresponding magnitude of β with the relationship shown in Equation 3.2. After multiplying 2^{15} and rounding down, the 16-bit signed binary representation of all αs and βs could be converted from the decimal. Therefore, the 16-bit binary would be slightly less than the actual fraction number, which means that after being squared, the sum is impossible to exceed 1. Since the square of 2^{15} is 2^{30}, the upper bound of the threshold is 2^{30}, which is 40000000 in hex. To determine the lower bound, we calculated the sum of the generated α square and β square and found the minimum value which is 3ffe9c88 in hex. The determination and evaluation of this threshold is discussed in detail in chapter 4.

3.4 Quantum Gate Operations

In quantum circuit emulation, a quantum gate is a transformation applied to the input qubit, similar to the logic gates in a classical circuit model. Mathematically, a quantum gate can be represented as a unitary matrix. A gate that acts on n qubits is represented by a $2^n \times 2^n$ matrix. In this section, we give some basic gates that are useful in developing quantum algorithms.

3.4.1 Pauli-X Gate

The Pauli-X gate applies a flipping rotation around the x-axis on the Bloch sphere, which is implemented by swapping the α and β of the input qubit. Equation 3.3 presents the mathematical description of the Pauli-X gate.

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{3.3}$$

For a qubit represented by Equation. 3.1, Pauli-X gate will produce the following output.

$$X|\psi\rangle = \beta|0\rangle + \alpha|1\rangle \tag{3.4}$$

Note that the gate does not introduce any error to the output, since it simply swaps the bits of α and β.

3.4.2 Pauli-Y Gate

The Pauli-Y gate applies a flipping rotation around the y-axis on the Bloch sphere. The implementation of the Pauli-Y gate first splits the complex coefficients into a real part and an imaginary part, and then the 2's complements of each part will be reorganized. Equation 3.5 presents the mathematical description of the Pauli-Y gate.

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \tag{3.5}$$

For a qubit represented by Equation. 3.1, Pauli-Y gate will produce the following output.

$$Y\left|\psi\right\rangle = (\beta_i - i\beta_r)\left|0\right\rangle + (-\alpha_i + i\alpha_r)\left|1\right\rangle \tag{3.6}$$

Where α_r, α_i, β_r and β_i are real numbers denote the real and imaginary part of α and β respectively. Note that the gate does not introduce any error to the output, since the gate operation only contains bit swapping and sign inversion.

3.4.3 Pauli-Z Gate

The Pauli-Z gate applies a flipping rotation around the z-axis on the Bloch sphere, which is implemented by inverting the input qubit's sign of β. Equation 3.7 presents the mathematical description of the Pauli-Z gate.

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{3.7}$$

For a qubit represented by Equation. 3.1, Pauli-Z gate will produce the following output.

$$Z\left|\psi\right\rangle = \alpha\left|0\right\rangle - \beta\left|1\right\rangle \tag{3.8}$$

Note that the gate does not introduce any error to the output, since it simply performs a sign inversion on β.

3.4.4 Hadamard Gate

Unlike the Pauli gates listed above, the Hadamard gate is described as a 180° rotation around the bisector between the x-axis and the z-axis on the Bloch sphere. Equation 3.9 presents the mathematical description of the Hadamard gate.

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{3.9}$$

For a qubit represented by Equation. 3.1, Hadamard gate will produce the following output.

$$H\left|\psi\right\rangle = \frac{1}{\sqrt{2}}(\alpha+\beta)\left|0\right\rangle + \frac{1}{\sqrt{2}}(\alpha-\beta)\left|1\right\rangle \tag{3.10}$$

Considering that conducting division is resource-consuming on FPGA, converting $\frac{1}{\sqrt{2}}$ to a 16-bit signed binary and performing multiplication is an alternative approach. As a result, the implementation of the Hadamard gate involves 4 multiplications and 4 additions, which would incur a discretization error on both complex coefficients of the output qubit.

The Hadamard gate's major functionality is adding randomness to the input qubit. Setting the input qubit to $|\psi\rangle = |1\rangle$ as an example, the output qubit would be $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ which is treated as the superposition of $|0\rangle$ and $|1\rangle$. The basis state $|0\rangle$ is added to the qubit with the same probability of state $|1\rangle$.

3.4.5 CNOT Gate

The Controlled-NOT gate acts on 2 input qubits, where one qubit acts as a control for the Pauli-X operation on another. Only when the first qubit is $|1\rangle$, the target qubit will be flipped, otherwise, it will remain unchanged. The input of the mathematical description of CNOT gate shown in Equation 3.11 are the complex coefficients with respect to the basis $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$.

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.11}$$

The CNOT gate is often used to generate entanglement between two qubits. In the emulation process we proposed, the inputs of the CNOT gate are the complex coefficients of two qubits, denoted as α_1, β_1, α_2 and β_2 respectively. The entanglement state raised by the inputs is shown in Equation 3.15, and the output of the CNOT gate is shown in Equation 3.16.

$$|\psi_1\rangle = \alpha_1 |0\rangle + \beta_1 |1\rangle \tag{3.12}$$

$$|\psi_2\rangle = \alpha_2 |0\rangle + \beta_2 |1\rangle \tag{3.13}$$

$$|\psi_{entangled}\rangle = |\psi_1\rangle \bigotimes |\psi_2\rangle \tag{3.14}$$

$$|\psi_{entangled}\rangle = \alpha_1\alpha_2 |00\rangle + \alpha_1\beta_2 |01\rangle + \beta_1\alpha_2 |10\rangle + \beta_1\beta_2 |11\rangle \tag{3.15}$$

$$CNOT\,|\psi_{entangled}\rangle = \alpha_1\alpha_2 |00\rangle + \alpha_1\beta_2 |01\rangle + \beta_1\beta_2 |10\rangle + \beta_1\alpha_2 |11\rangle \tag{3.16}$$

It could be noticed that the entanglement contains 16 multiplications, thus the CNOT gate would introduce errors.

3.5 Quantum State Measurement

In quantum physics, a qubit could be in the superposition of two basic states $|0\rangle$ and $|1\rangle$. When people try to measure which state the qubit is exactly in, it will collapse to either $|0\rangle$ state or $|1\rangle$ state. Equation 3.2 describes the condition of a qubit in the pure state, where

$|\alpha|^2$ expresses the probability of a qubit landing as $|0\rangle$, while $|\beta|^2$ represents the probability of being measured as $|1\rangle$.

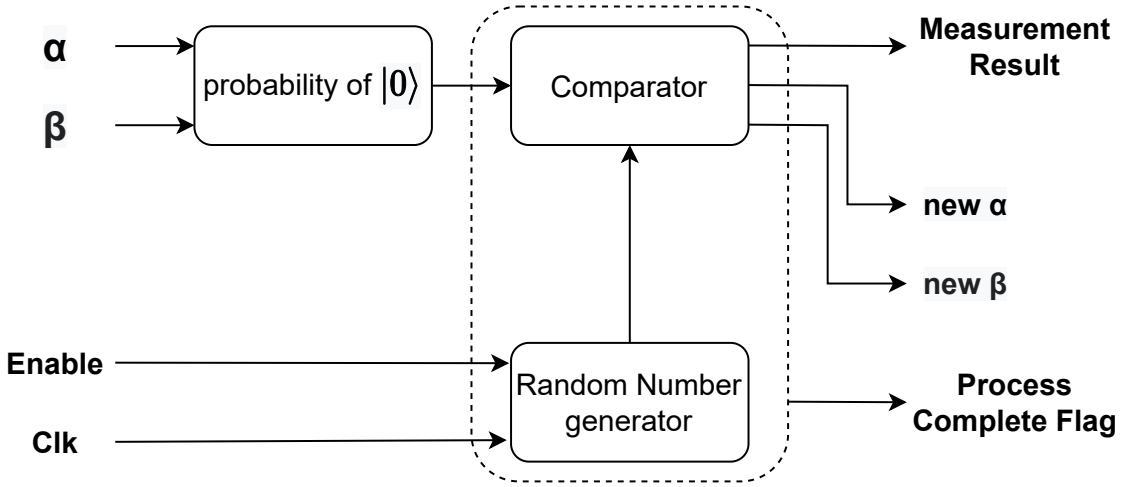

Figure 3.5: The data flow of quantum state measurement block.

Figure 3.5 presents the data flow of the quantum state measurement block. When measuring a single qubit, we only need to calculate $|\alpha|^2$. In the proposed design, the measurement block acquires a sequence of random bits from the RNG and compares it with the probability calculated by the complex coefficient. If the 32-bit bitstream generated by the RNG is less than $|\alpha|^2$, the measurement result is 0. Otherwise, the result should be 1. At the same time, to simulate the collapse effect of measurement, the corresponding quantum state coefficients will be overwritten to match the measured result.

We implemented 2 RNGs, with details shown in the following subsections. The LFSR-based pseudo-random number generator is for testing the measurement block, but the result

is pre-determined, thus the RO-based true random number generator is implemented in the emulation process we proposed.

3.5.1 LFSR-based Pseudo Random Number Generator

The linear feedback shift register is implemented as a series of DFFs that are wired together as a shift register. 4 taps points of the shift register chain are used as inputs to an XNOR gate. The output of the XNOR gate is then used as the input to the beginning of the shift register chain, hence formulating feedback. Meanwhile, all bits except the last one shift forward, and the feedback input will take the first bit.

Longer LFSR will take a longer time to run through all iterations. The longest possible number of iterations for an LFSR of N bits is $2^N - 1$. To create the maximum possible LFSR length for each bit width, the tap points are chosen according to (reference). For a 4-bit LFSR, the period of output is $2^4 - 1 = 15$ clock cycles. Considering that both the real part and imaginary part of α are represented by a 16-bit binary, the calculated possibility of $|0\rangle$ is a 32-bit binary. As a result, the length of LFSR is set to 32 bits in the proposed design. Figure 3.6 shows the architecture of a 32-bit LFSR-based PRNG, where the tap points are located at bits 1, 2, 22 and 32.

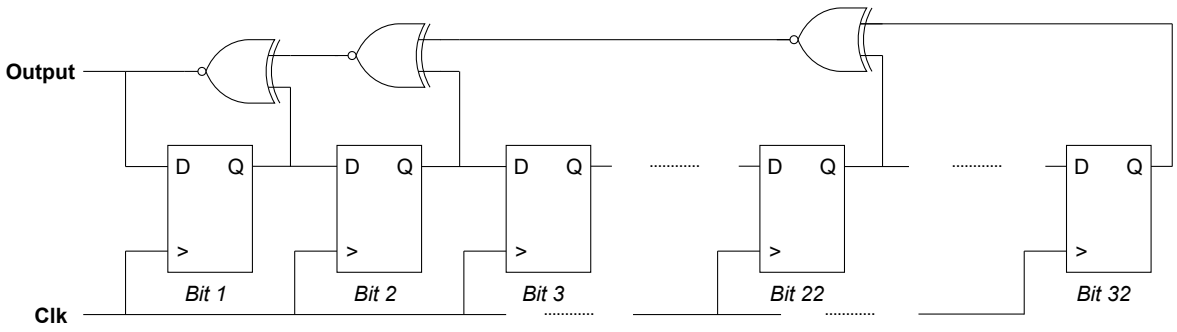

Figure 3.6: The architecture of the LFSR-based PRNG.

When an LFSR is running, the pattern that is being generated by the individual DFFs is pseudo-random because the next state could be predicted from the current state of the LFSR pattern at any time. That is to say, with identical default input, the produced bit stream after 32 clock cycles is always the same. Thus, we modified the clock cycle sampling to add some randomness to the output. The 32-bit probability is divided into 8 4-bit binaries and converted to decimal integers, which would be used as the clock cycles between each generated consecutive output. Figure 3.7 shows a sample of the sampling, after 9 clock cycles, the LFSR will provide the first bit of the output, and then after 5 clock cycles, the second output bit will be generated. A total of 8 bits will be generated through this method and will be used as the 8 MSB to compare with the calculated probability. No more bit is needed since the probability of two 8-bit binary being the same is as low as 0.3906%.

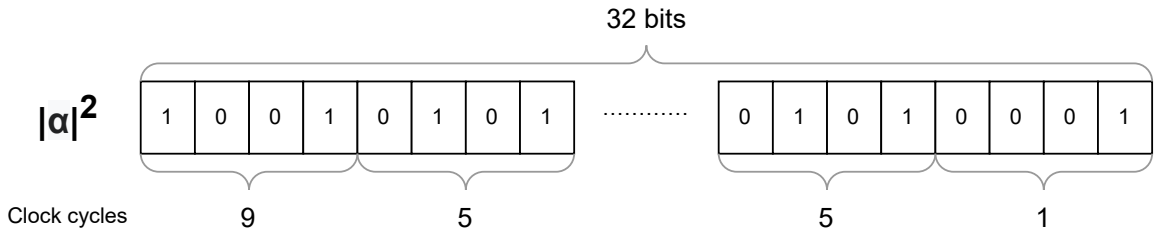

Figure 3.7: An example of modified LFSR output sampling.

3.5.2 RO-based Random Number Generator

Although the modified LFSR-based PRNG fulfills the requirement of generating random numbers, the produced outputs are still deterministic. That is to say, the sequence of random bits generated by a given input will always be the same, which will affect the analysis of the performance of the quantum gates and quantum algorithms. As a result, a true random number generator that produces random bits with a given probability is irreplaceable.

In this section, we propose a ring oscillator-based true random number generator. The ring oscillator is composed of an odd number of NOT gates, and the last NOT gate is attached to the AND gate to formulate a ring, as shown in Figure 3.8. When the Enable signal is set to high, the ring oscillator will be activated, and the output will oscillate between 0 and 1. Note that the number of inverters should always be odd, otherwise, the output will always be the same as the input to the first inverter, which means that the output will never change.

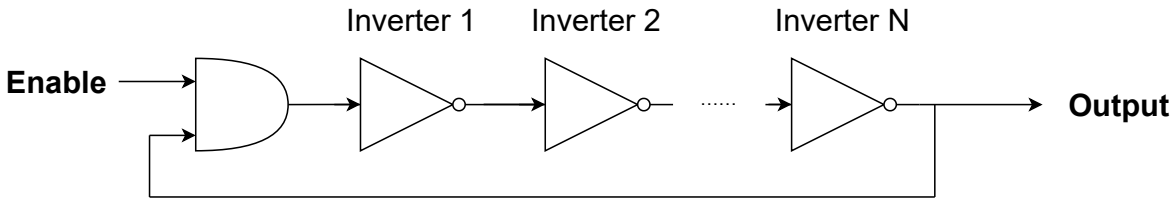

Figure 3.8: The architecture of ring oscillator with enable signal.

In the practical world, the gate will not respond instantaneously to the signal being processed. The output of any inverter within an RO takes a specific duration to process the input after it has been updated. As is shown in Figure 3.9, the output signal's phase relative to the sample clock drifts over time due to power supply variations, cross talks, semiconductor noise, temperature variations and propagation delays.

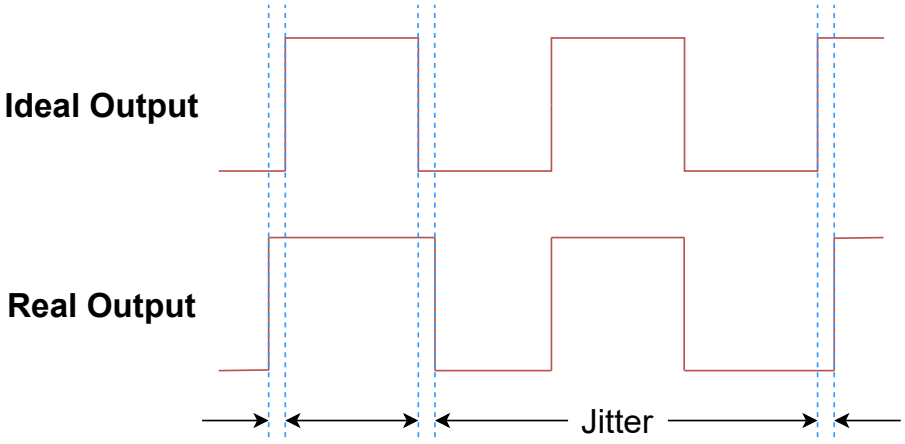

Figure 3.9: Representation of jitter of the output of ring oscillator.

The arrangement of gates presents a specific oscillation pattern, the period of which

depends on the accumulated delays in the feedback loop. Equation 3.17 demonstrates the calculation of the RO's output changing period, where T_d refers to the delay of an inverter, N refers to the number of consecutive inverters and $\sum T_j$ refers to the jitter accumulated through the inverters of a running RO. The jitter period can be used as the entropy source for generating random bits by sampling the output using DFFs.

$$T_{osc} = 2NT_d + \sum T_j \tag{3.17}$$

Figure 3.10 shows an example of DFF sampling. At the rising edge of the sampling clock, the output of RO is recorded by DFF. With the clock of DFF set to a relatively low frequency compared to the RO output, the randomness generated by jitters is preserved. Otherwise, the DFF will always hold the same value, and the randomness will be eliminated.

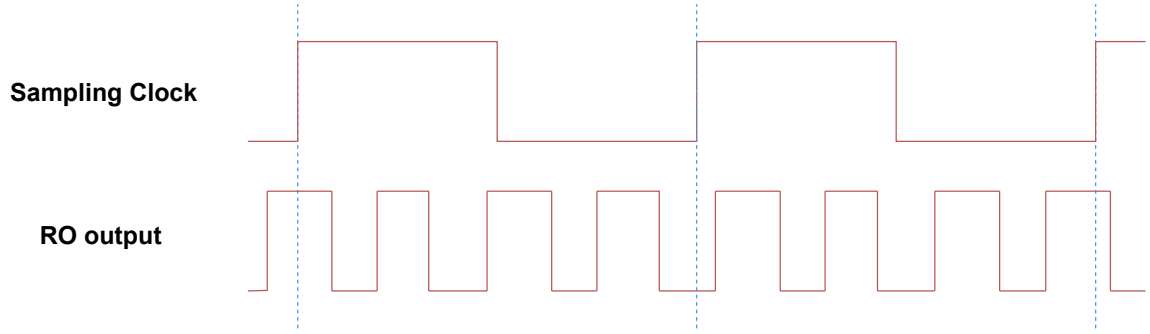

Figure 3.10: An example of DFF sampling.

The quality of generated random bits can be improved by employing multiple ROs, and feeding the outputs to an XOR gate. Figure 3.11 demonstrates the architecture of RO-based

RNG. Several ROs will be running parallel when the enable signal is set to high. At the output of each RO, a sampling DFF will periodically record the random bits generated by RO, controlled by the same sampling clock signal. After being XORed, the random bit stream will be loaded to compare with the calculated probability.

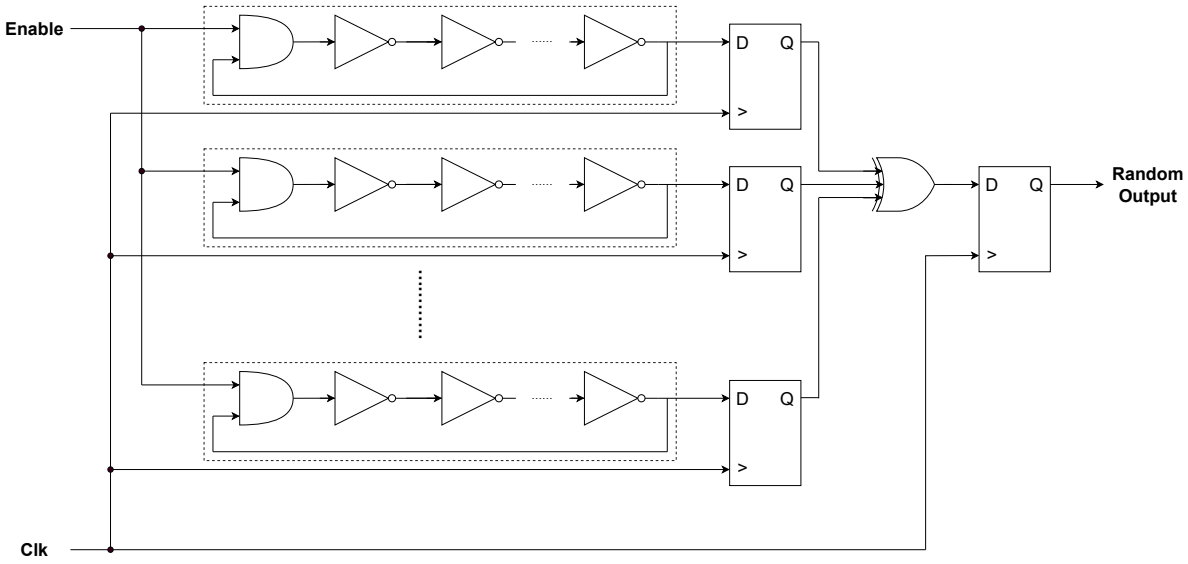

Figure 3.11: The architecture of ring oscillator-based random number generator.

Since the LUT resources are limited on an FPGA, the number of ROs and the number of inverters in an RO should be as small as possible, while the randomness of the generated bit stream should remain the same. In our proposed design, there are 4 RO stages which have only 5 inverters each. A detailed analysis of the RORNG is in the following chapter.

3.5.3 Multi-Qubit measurement

When there are multi-qubit gates such as CNOT applied in the quantum circuit, the qubits are entangled, and the measurement of such conditions is different from measuring a single qubit. As is shown in Equation 3.15, such entangled two qubits will have up to 4 possible states, whereas there are only 2 for a single qubit. As a result, the measurement of the single qubit does not apply to multi-qubit measurement, and we proposed a method for measuring entangled qubits.

As aforementioned in this chapter, when measuring a single qubit, the RNG in the measurement block will generate a 30-bit random bitstream, pad with 0 at the front, and compare it with the magnitude of the complex coefficient of $|0\rangle$ state. If the random bitstream is smaller, the qubit will be measured as $|0\rangle$, otherwise, it will be gauged as $|1\rangle$. However, this method is not applicable when the number of states is more than 2, since the number of states is inconsistent with the boolean states: there might be more than one state qualified as the measurement result. Thus, instead of measuring the state of entangled qubits, the state of each qubit will be measured and combined.

Equation 3.18 to 3.20 are the representations of qubits $|\psi_1\rangle$, $|\psi_2\rangle$ and their entangled qubits $|\psi_{entangled}\rangle$. The coefficients of the entangled qubits are derived from the coefficients of $|\psi_1\rangle$ and $|\psi_2\rangle$. Hence, the coefficient of both qubits could be retrieved.

$$|\psi_1\rangle = \alpha_1 |0\rangle + \beta_1 |1\rangle \tag{3.18}$$

$$|\psi_2\rangle = \alpha_2 |0\rangle + \beta_2 |1\rangle \tag{3.19}$$

$$|\psi_{entangled}\rangle = \alpha_1\alpha_2 |00\rangle + \alpha_1\beta_2 |01\rangle + \beta_1\alpha_2 |10\rangle + \beta_1\beta_2 |11\rangle \tag{3.20}$$

Equation 3.21 and 3.22 are the intrinsic constraints held by $|\psi_1\rangle$ and $|\psi_2\rangle$. If the square of the magnitude of α_1 is multiplied on both sides of Equation 3.22, Equation 3.23 could be obtained, where $\alpha_1\alpha_2$ and $\alpha_1\beta_2$ are the coefficient of the basis states of the entangled qubits $|\psi_{entangled}\rangle$. All 4 coefficients of the basis states of a single qubit could be acquired the same way, now that we only need to conduct qubit measurement 2 times.

$$|\alpha_1|^2 + |\beta_1|^2 = 1 \tag{3.21}$$

$$|\alpha_2|^2 + |\beta_2|^2 = 1 \tag{3.22}$$

$$|\alpha_1\alpha_2|^2 + |\alpha_1\beta_2|^2 = |\alpha_1|^2 \tag{3.23}$$

$$|\beta_1\alpha_2|^2 + |\beta_1\beta_2|^2 = |\beta_1|^2 \tag{3.24}$$

$$|\alpha_1\alpha_2|^2 + |\beta_1\alpha_2|^2 = |\alpha_2|^2 \tag{3.25}$$

$$|\alpha_1\beta_2|^2 + |\beta_1\beta_2|^2 = |\beta_2|^2 \tag{3.26}$$

This method could be extended to entangled n-qubit measurement. Equation 3.27 shows the state of entangled n-qubits represented by 2^n base states. For each base state, the complex coefficient is calculated by multiplying n corresponding coefficients of the single qubit.

$$|\psi_{entangled}\rangle = \alpha_1\alpha_2\ldots\alpha_n\,|00\ldots0\rangle + \alpha_1\alpha_2\ldots\beta_n\,|00\ldots1\rangle + \ldots + \beta_1\alpha_2\ldots\alpha_n\,|10\ldots0\rangle + \beta_1\alpha_2\ldots\beta_n\,|10\ldots1\rangle + \ldots + \beta_1\beta_2\ldots\alpha_n\,|11\ldots0\rangle + \beta_1\beta_2\ldots\beta_n\,|11\ldots1\rangle \tag{3.27}$$

To retrieve the probability of a single qubit from the entangled qubits, take α_1 as an example, 2^{n-1} multiplications are required, as is shown in Equation 3.28.

$$\begin{aligned}|\alpha_1|^2 = {} & |\alpha_1\alpha_2\ldots\alpha_{n-1}\alpha_n|^2 + |\alpha_1\alpha_2\ldots\alpha_{n-1}\beta_n|^2 + \ldots + |\alpha_1\alpha_2\ldots\beta_{n-1}\alpha_n|^2 + \\ & |\alpha_1\alpha_2\ldots\beta_{n-1}\beta_n|^2 + \ldots + |\alpha_1\beta_2\ldots\alpha_{n-1}\alpha_n|^2 + |\alpha_1\beta_2\ldots\alpha_{n-1}\beta_n|^2 + \\ & \ldots + |\alpha_1\beta_2\ldots\beta_{n-1}\alpha_n|^2 + |\alpha_1\beta_2\ldots\beta_{n-1}\beta_n|^2\end{aligned} \tag{3.28}$$

When simulating the measurement of a pair of entangled qubits, 2 qubit measurement blocks need to work at the same time to keep the time consistency of processing a single qubit measurement. In addition, the pre-processing of complex coefficients introduces extra time and logic gates, and the computing resources grow exponentially with the size of qubits.

3.6 Probabilistic Execution Predictor

The quantum measurement block requires at least 30 system clock cycles to generate a 30-bit random bit stream. In contrast, the time needed for a qubit to be processed through a quantum gate is less than 1 clock cycle. Considering the dynamic quantum circuit will always have at least 1 midway measurement on a qubit, the whole emulation process will stall and wait for the random number generation, thus wasting time.

The probabilistic execution predictor is designed to deal with this situation. Inspired by the static branch prediction technique in computer architecture which solely predicts the outcome based on the branch instruction, the probabilistic execution prediction block implemented in the emulation process chooses the branch based on the probability indicated by the input qubit state. Since the probability of each state is implied by the complex coefficient of each basis state, the prediction would be performed with pre-processing of the coefficients, and the correct prediction rate could be raised by choosing the state with a higher probability.

Figure 3.12 shows the data flow of the probabilistic execution prediction block. The complex coefficient of the qubit to be measured will be loaded to the quantum measurement block and the probabilistic execution prediction block in parallel. Before being transformed to the new coefficients and stored in the temporary state registers, the probability of both basis states will be calculated and compared, where $|\alpha|^2$ denotes the probability of $|0\rangle$ and $|\beta|^2$ refers to the probability of being measured as $|1\rangle$.

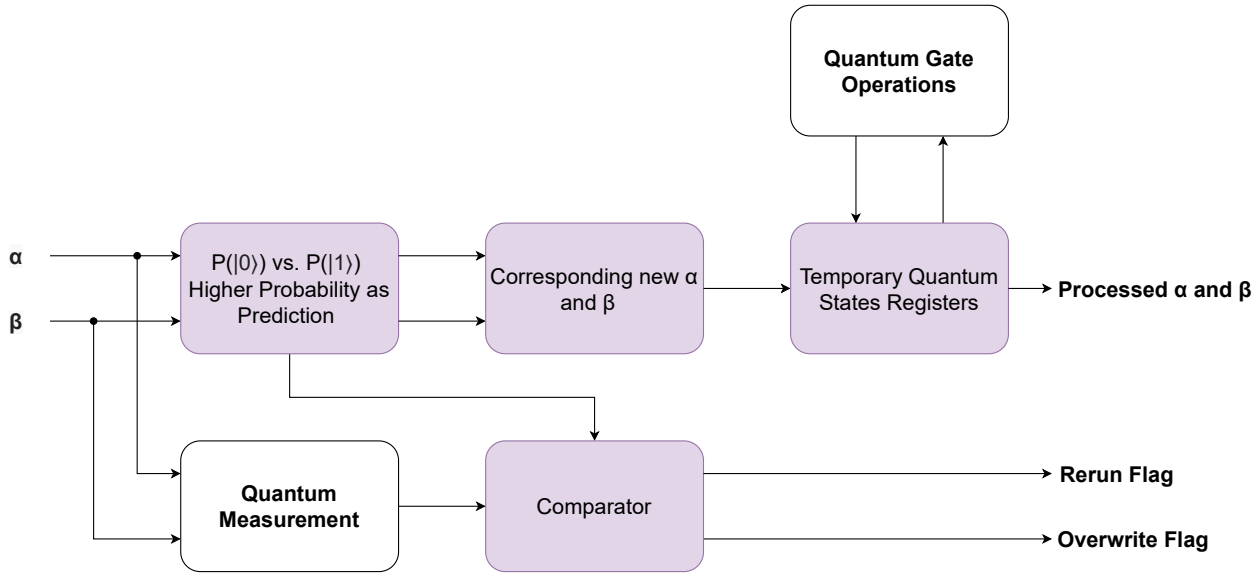

Figure 3.12: The data flow of probabilistic execution prediction block.

If $|\alpha|^2$ is larger than $|\beta|^2$, the predictor will take the $|0\rangle$ branch, set the temporary quantum state to $|0\rangle$, adjust the coefficients of the temporary α_{temp} and β_{new} to 16'h7fff and 16'h0000 respectively, vice versa. After that, the coefficients will be stored in the temporary quantum state register with the same structure as the quantum state register. The following quantum gate operations will be performed on this temporary qubit.

Once the quantum state measurement is done, the measured result will be compared with the prediction. If these two results are consistent, the prediction is correct, the overwrite flag will be set as high, and the processed temporary coefficients will overwrite the qubit stored in the quantum state register. Otherwise, the prediction is wrong, the rerun flag will be set as high for one clock cycle, the qubit stored in the temporary quantum state register

will be eliminated, and the quantum gate operations will be performed again on the correct quantum state.

The comparison between the probability of basis states is optimized to save hardware resources. According to Equation 3.2, the sum of two squares is always equal to 1. Hence, if one probability is larger than 0.5, it will take the majority and will be larger than the other. As a result, only one square calculation is mandatory, and the hardware resources could be saved up to 50%. To keep consistent with the quantum measurement block, the comparison is made between $|\alpha|^2$ and 32'h20000000 which refers to 0.5 in decimal.

The probabilistic execution prediction has no extra time costs compared to the original emulation process. Midway quantum measurements introduce a pause in the conventional quantum circuit emulation process. It's necessary to wait for the measurement outcome before proceeding further with quantum gate operations. With the probabilistic execution prediction block applied, the time of waiting for the quantum state measurement will be saved if the prediction is correct. Moreover, there will be no extra penalty for the wrong prediction, since the gate operations performed on the temporary qubit are parallel to the quantum measurement.

Chapter 4

Results and Discussion

In this chapter, the analysis and evaluation of the proposed quantum circuit emulation process have been performed extensively using the Vivado environment and programmed on the Digilent Cmod A7-35T FPGA board. Results related to each functional block are reported separately in the following subsections.

For the validation check block, a set of qubits in the pure state and a set of qubits in the mixed state are used to test the pass rate of different thresholds. For the quantum state measurement block, the performance of LFSR-based PRNG and RO-based RNGs are evaluated. For the probabilistic execution predictor, the relationship between the qubit and the miss rate is quantized and discussed. Two sample circuits are presented and analyzed to show the improvement compared to the conventional emulation process.

4.1 Validation Check Criteria Determination

This section presents the determination of the validation threshold mentioned in section 3.3. Since the fraction number will be estimated and rounded down when converted to binaries, errors will be generated. Since the design principle of the validation check is to exclude the qubits in the mixed state, some pure qubits may fail the test if the threshold is too tight due to the error. On the other hand, some qubits in the mixed state might pass the check if the threshold is too loose. Hence, a set of 10000 random qubits in the pure state is used on the validation check block to test the pass rate of the threshold. Moreover, a set of 10000 random qubits in the mixed state is also used to test the reliability of the threshold.

The coefficients of basis states of all qubits are random numbers in the range of (0,1) generated using Python. For the set of qubits in the pure state, the α is generated, and the β is calculated. Whereas for the set of qubits in the mixed state, all coefficients are random fractional numbers. The fractional decimals will then be multiplied by 2^{15}, rounded down and converted to binary numbers. Since the square of a 16-bit signed binary number is a 32-bit binary number with the first two bits always being 0, the lower bound of the threshold is chosen to be 32'h3fff0000 and 32'h3ffe0000 empirically. As described in section 3.3, 32'h3ffe9c88 is the lower bound used in the implementation which is related to the minimum value among the sum of the square of the coefficients of the qubits in the pure states. Figure 4.1 visualizes the bias of each qubit, where the 3 red lines refer to the lower bounds of different thresholds. The index of the y-axis refers to the bias between the hex

number 40000000 and the sum of squares.

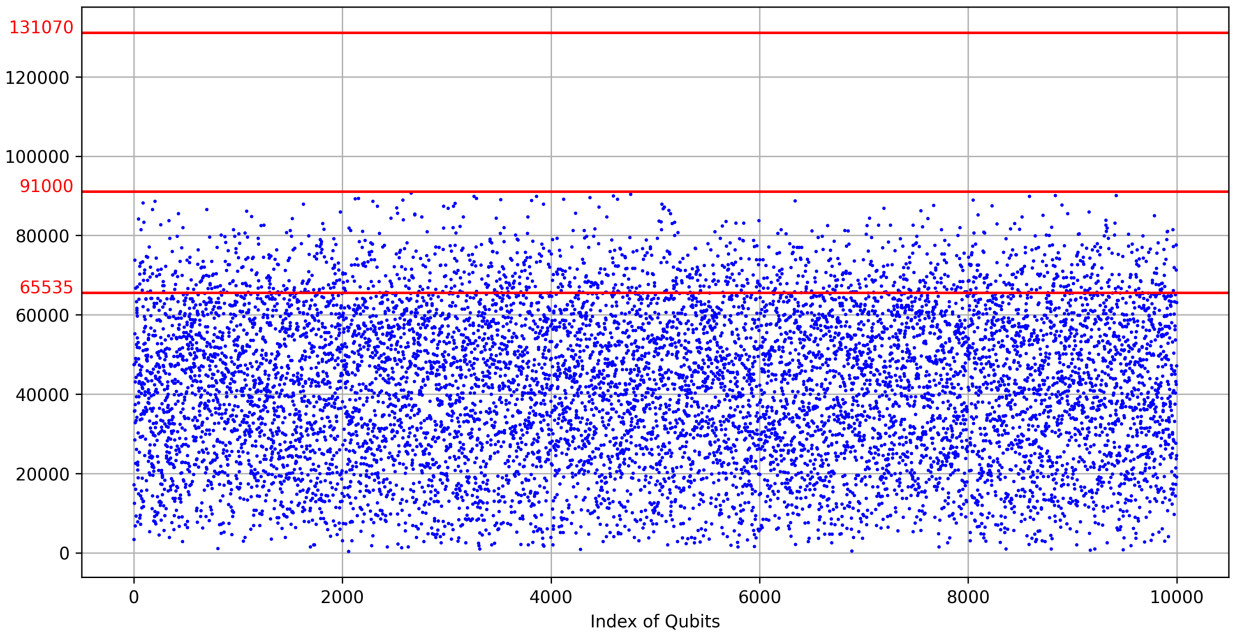

Figure 4.1: Bias between the sum of squares of coefficients and hex number 32'h40000000 of each qubit in the pure states.

Table 4.1 shows the performance of each threshold, with all higher bounds set to 32'h40000000. It could be observed that both empirical criteria have drawbacks, either failing to detect valid qubits or passing invalid qubits unintentionally. In contrast, the lower bound implemented in the proposed validation check block provides a competent performance, with no qubits in the mixed state passing the check and only 2 qubits in the pure state failing the test. The reason for the valid qubits to pass the test is to avoid the unexpected condition that some invalid qubits pass the validation check.

Table 4.1: Thresholds performance

Lower Bound	Failed Valid Qubit	Passed Rate	Passed Invalid Qubit	Failed Rate
3fff0000	1184	88.16%	0	100%
3ffe9c88	2	99.98%	0	100%
3ffe0000	0	100%	4	99.96%

4.2 Random Number Generator Evaluation

This section presents the performance of LFSR-based PRNG and different patterns of RO-based RNGs and how they are evaluated. Since the design principle of the quantum measurement block is to compare the calculated state probability with the generated bit stream, the affiliated RNG should provide an equal probability of 0s and 1s for the generated random bit.

4.2.1 LFSR-based Pseudo Random Number Generator

Provided that the LFSR-based PRNG requires a 32-bit binary number to set as the default value, 50 32-bit binaries are randomly generated. As mentioned in section 3.5.1, since the longest possible number of iterations for the 32-bit LFSR is $2^{32} - 1$, each 32-bit binary was iterated and recorded for 200000 rounds. Figure 4.2 shows the proportion of 0s among all bits generated by the LFSR using the random inputs, and it could be spotted that all results fall between the narrow band around 0.5. The mean of the proportions is 0.500036 and the variance is 1.245×10^{-6}, implying that the probability of 0 and probability of 1 generated by the LFSR-based PRNG is almost equal.

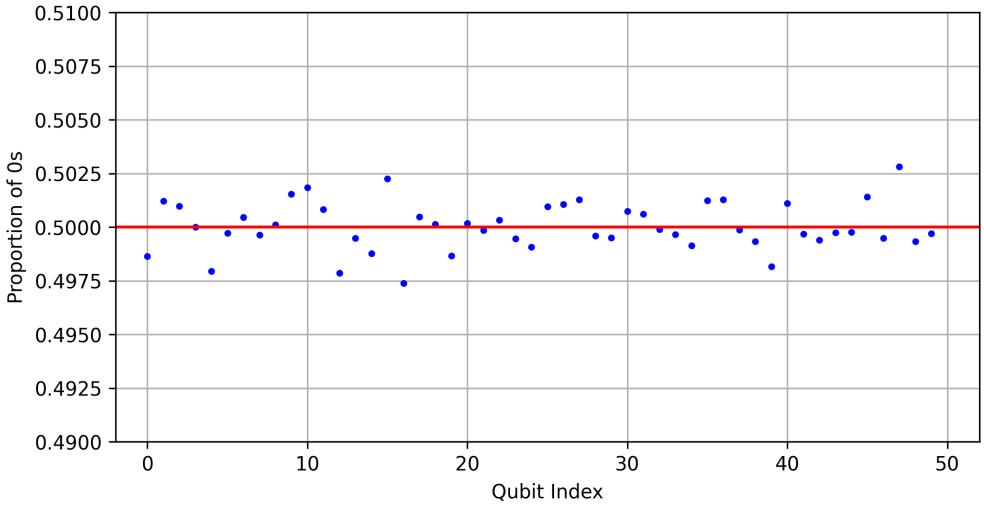

Figure 4.2: The single-bit performance of LFSR-based PRNG.

Considering that the purpose of RNG in the quantum measurement block is to generate consecutive bit streams whose distribution ought to be consistent with the calculated probability, 15 uniformly separated tap points were chosen to compare with the consecutive bits. As shown in Figure 4.3, the biases between the expected probability and the measured probability are less than 0.2%, which implies that the consecutive bits generated by the LFSR-based PRNG are almost uniformly distributed.

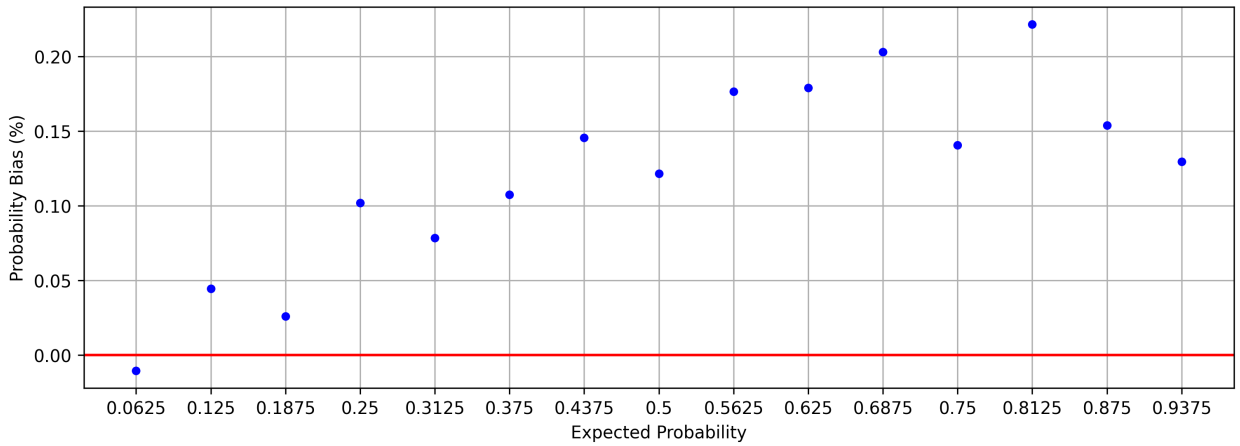

Figure 4.3: A sample of the consecutive bit stream performance of LFSR-based PRNG.

4.2.2 RO-based Random Number Generator

To analyze the performance of RO-based RNG, various layouts with different numbers of RO stages and inverters were implemented. A Digilent Cmod A7-35T FPGA board was used to implement the RNGs, with the operating clock frequency set to 12 MHz. The random bit sequence is extracted by a UART interface, with the Baud rate set to 115200 and the data size set to 8-bit without parity bit. For a fair comparison, we tested a 32-bit sequence among a 200,000-bit sequence for each RO-based RNG layout.

Figure 4.4 shows the generated random results of RO-based RNGs with fixed RO stages and 5, 15, 35 and 75 inverters separately. It could be observed that for 2 RO stages, the number of 0s among all results increases with the number of inverters. This relationship between the number of 0s and the number of inverters would vanish when the number of

RO stages increased. The output random bits for 4 RO stages and 6 RO stages show an approximate 50% chance of generating 0. The randomness gained from the jitter through a single RO is limited, and the randomness of each RO is accumulated with an XOR gate applied at the output of all ROs. However, the randomness is not linearly related to the number of RO stages. When the randomness reaches its limit, the randomness of the results will remain the same even if more RO stages are applied.

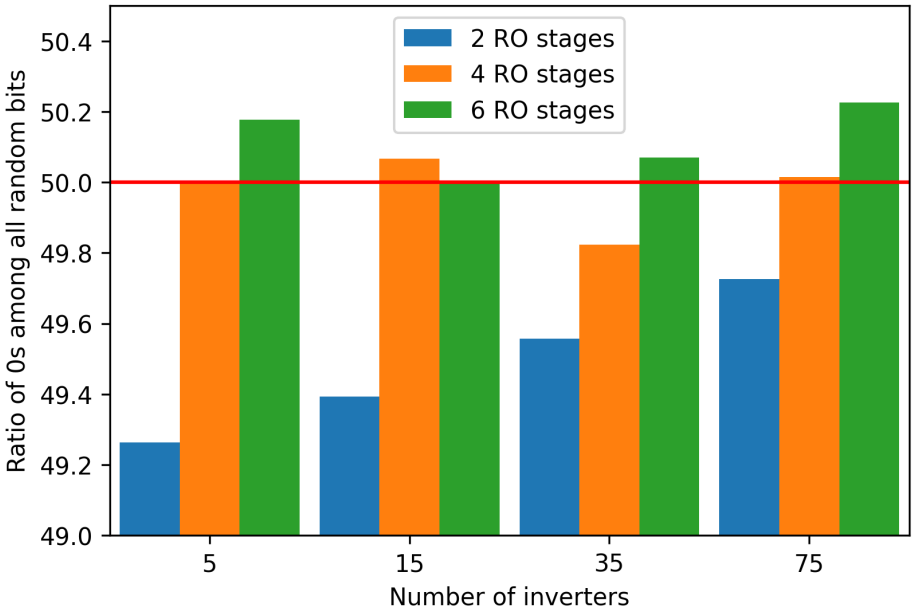

Figure 4.4: Generated random bits of 2, 4 and 6 RO stages with different numbers of inverters.

This explanation could be confirmed by the random results generated by more RO stages shown in Figure 4.5, where the ratio of 0s among the generated random bits are all around 50% for all layouts with more than 6 RO stages. As a result, the single-bit performance

of RO-based RNG meets the requirement of uniformly distributed 0s and 1s among the generated random bits.

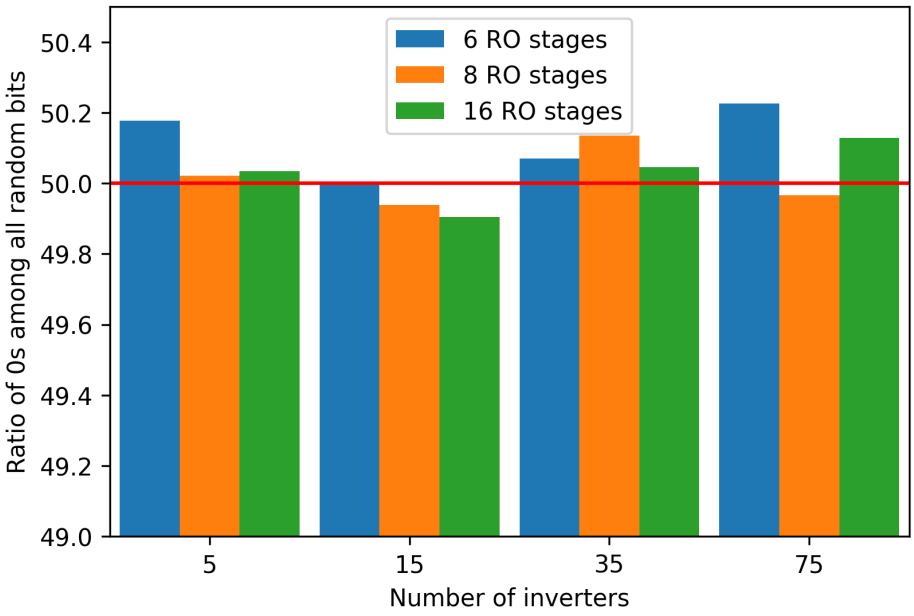

Figure 4.5: Generated random bits of 6, 8 and 16 RO stages with different numbers of inverters.

Similar to when testing the LFSR-based PRNG, 15 equally spaced points from 0 to 1 are set as the tap points to evaluate the goodness-to-fit between the observed distribution and the expected distribution, where the observed consecutive bits frequency of occurrence should be consistent with the chosen points. Figure 4.6 shows the probability biases of an RO-based RNG with 2 RO stages and various numbers of inverters. For a layout with 2 RO stages and 5 inverters, the bias could be as large as -0.86% . It could be observed that with the number of inverters in each RO stage increasing, the biases between the observed

probability and the expected probability decrease for all tap points, which further proves that the jitter accumulated through the inverters in an RO contributes to the randomness generation.

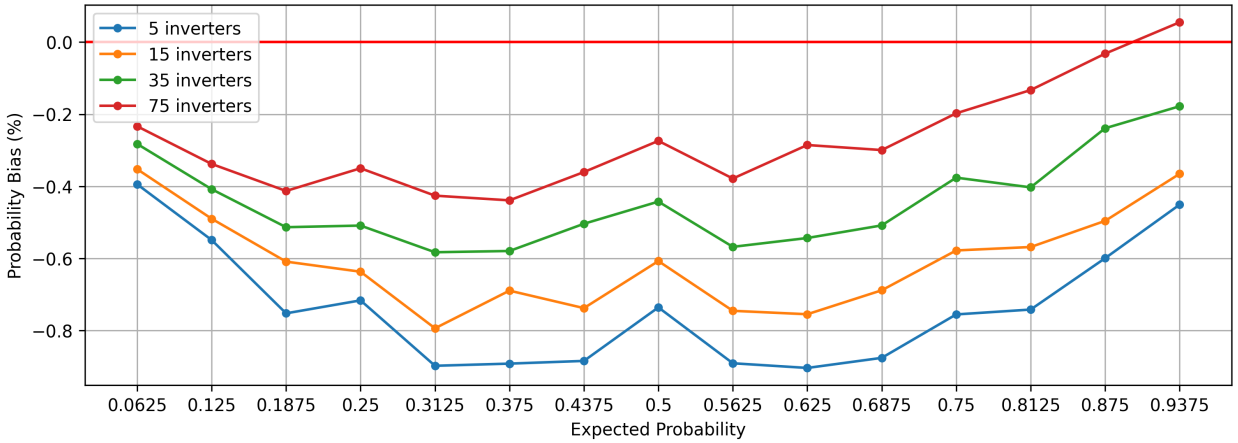

Figure 4.6: Probability biases of RO-based RNG with 2 RO stages

Table 4.2 provides the average probability biases of different layouts. The absolute value of the bias of RNG with more than 2 stages sharply decreases compared with the stats of the 2-stage RO-based RNG, which is consistent with the single-bit performance stated previously. To minimize the usage of hardware resources, the RO-based RNG implemented within the quantum measurement block has 4 RO stages with 5 inverters each.

Table 4.2: Averge probability biases of different RO-based RNG layouts

Number of Inverters	2 stages	4 stages	6 stages	8 stages	16 stages
5	-0.7361%	-0.0034%	0.1778%	0.0214%	0.0348%
15	-0.6076%	-0.0669%	-0.0022%	-0.0623%	-0.0953%
35	-0.4425%	-0.1771%	0.0703%	0.1355%	0.0468%
75	-0.2737%	-0.0545%	0.2257%	-0.0342%	0.1288%

4.3 Probabilistic Execution Predictor Evaluation

This section mainly focuses on the miss rate and the time analysis of the probabilistic execution predictor. Miss rate describes the accuracy of the predictor, lower the miss rate implies higher the accuracy of the prediction. The miss rate of the proposed predictor is associated with the state of the qubit to be measured.

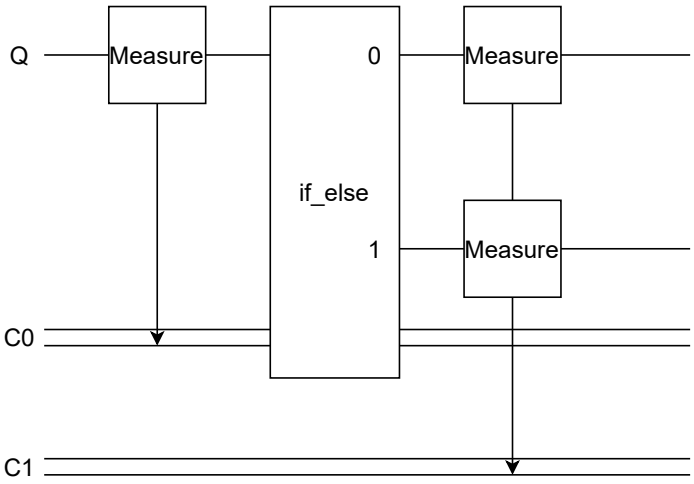

Figure 4.7: Miss rate test circuit

200 qubits in the pure state were randomly generated using Python, each qubit was tested 100 times by the quantum circuit shown in Figure 4.7. The RO-based RNG in the measurement block contains 4 RO stages with 5 inverters in each stage. The first measurement block indicates the dynamic circuit's midway measurement, which would invoke the predictor. Since the predictor will generate a temporary qubit state that is either $|0\rangle$

or $|1\rangle$, the measurement in the branches won't be processed. It will directly output the corresponding conventional bit and is used to show the prediction result. The prediction is correct if the conventional bits C1 and C0 are identical, otherwise it will be treated as a misprediction.

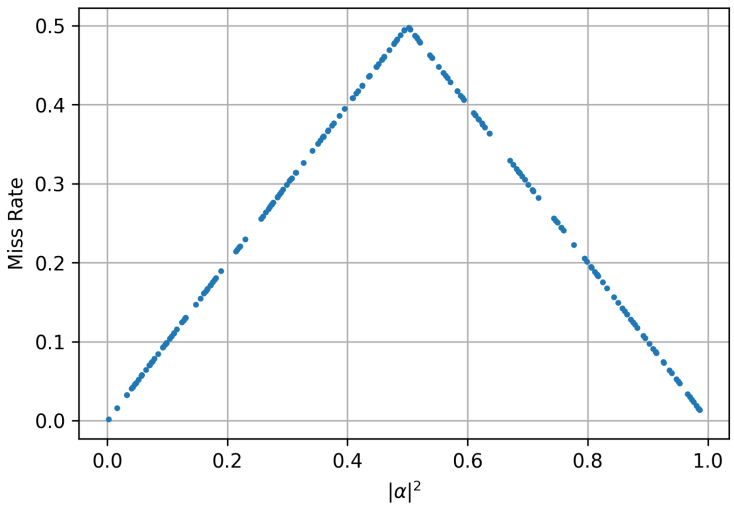

Figure 4.8: Relationship between miss rate and $|\alpha|^2$

Figure 4.8 shows the relationship between the square of the coefficient of the basis state $|0\rangle$ and the miss rate. Since the predictor takes the branch according to the comparing result of $|\alpha|^2$ and $|\beta|^2$, the miss rate peaks at 0.5 when the probability of $|0\rangle$ equals the probability of $|1\rangle$ and the miss rate is the same as the smaller one of the basis state probability. Thus, the average miss rate is 25%.

The time saved by the probabilistic execution predictor is related to the quantum circuit layout and could be as large as the time of qubit measurement. In the real world, when

people detect the qubit's state, the qubit will collapse to either of the basis states, and the measurement procedure is done instantly. The time saved by the predictor is the extra time needed to generate random bits when simulating the measurement of qubits. In the proposed emulation process, the time for measuring a qubit is designed to be fixed at 32 clock cycles. However, when the processing time of the following operations in the predicted branch is shorter than the quantum measuring time, the predictor still needs to wait for the measuring result.

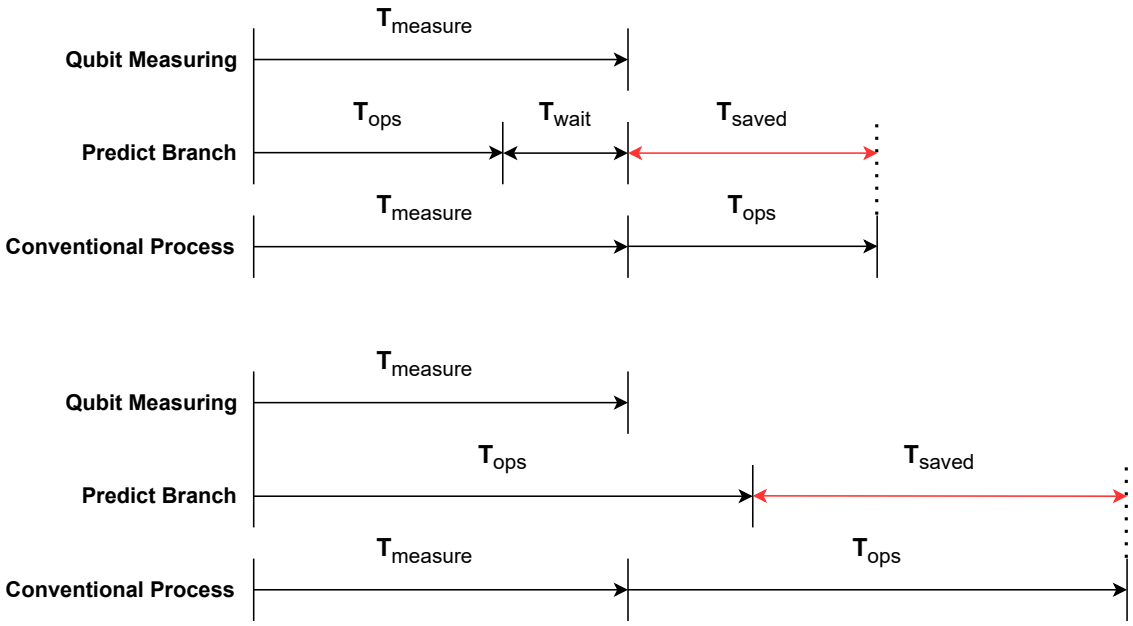

Figure 4.9: Time schematic of correct branch prediction

Figure 4.9 demonstrates the time schematic of correct branch predictions. When the processing time of the operations in the predicted branch is shorter than the qubit measuring

time, the time saved by the predictor is equal to the operation processing time. When the operation processing time is longer than the qubit measuring time, the time saved would be longer, equal to the qubit measuring time.

Figure 4.10 shows the time schematic of branch mispredictions. It could be observed that there's no extra penalty for the wrong prediction compared to the conventional emulation process, regardless of the operation processing time. When the operation processing time is longer than the qubit measuring time, the gate operations will be stopped once the qubit measurement indicates the prediction is wrong. Thus, the time could be saved to avoid the misprediction penalty.

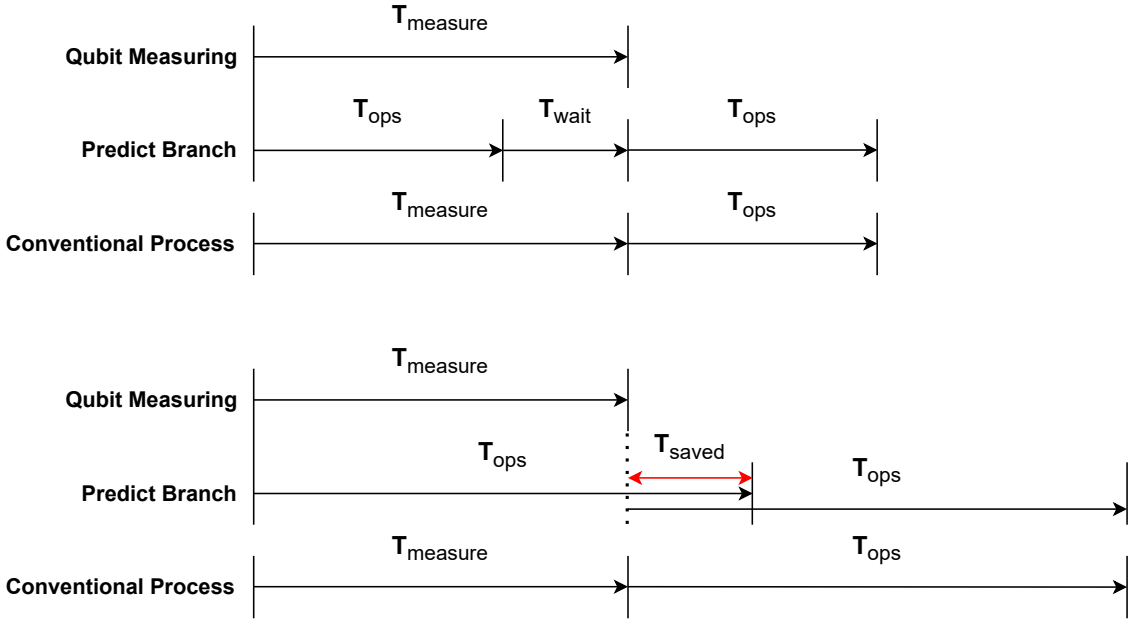

Figure 4.10: Time schematic of wrong branch prediction

4.4 Dynamic Quantum Circuit Case Study

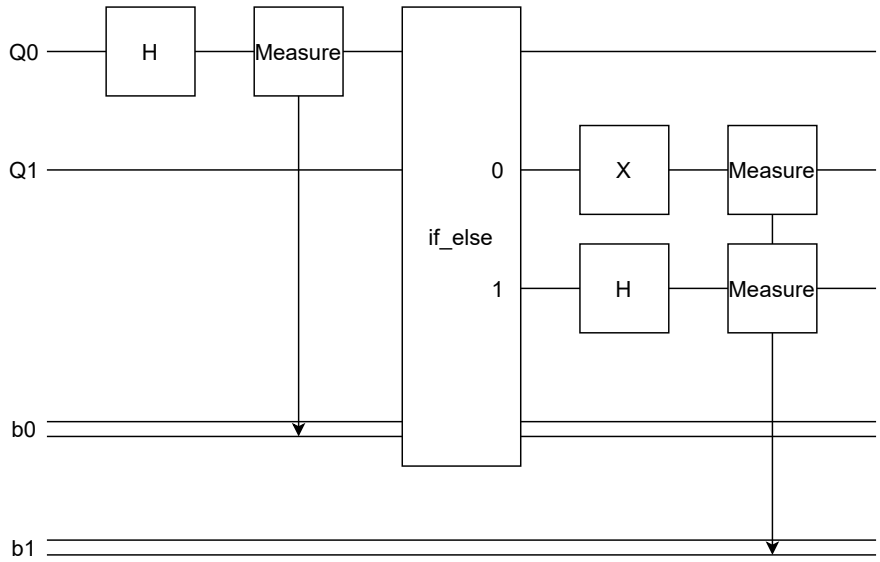

Figure 4.11: Dynamic quantum circuit example

In practice, the dynamic circuit is often used to control the qubit with another qubit. Figure 4.11 shows a dynamic quantum circuit example provided by IBM [35]. In this case, the aim will be to act differently on $|Q_1\rangle$ depending on the value of $|Q_0\rangle$. Qubit Q_0 is randomized by a Hadamard gate and measured to determine the gate operation to be performed on the qubit Q_1. If the measured result of Q_0 is 0, a Pauli-X gate will be applied to Q_1, otherwise a Hadamard gate will be performed, followed by the quantum state measurement process.

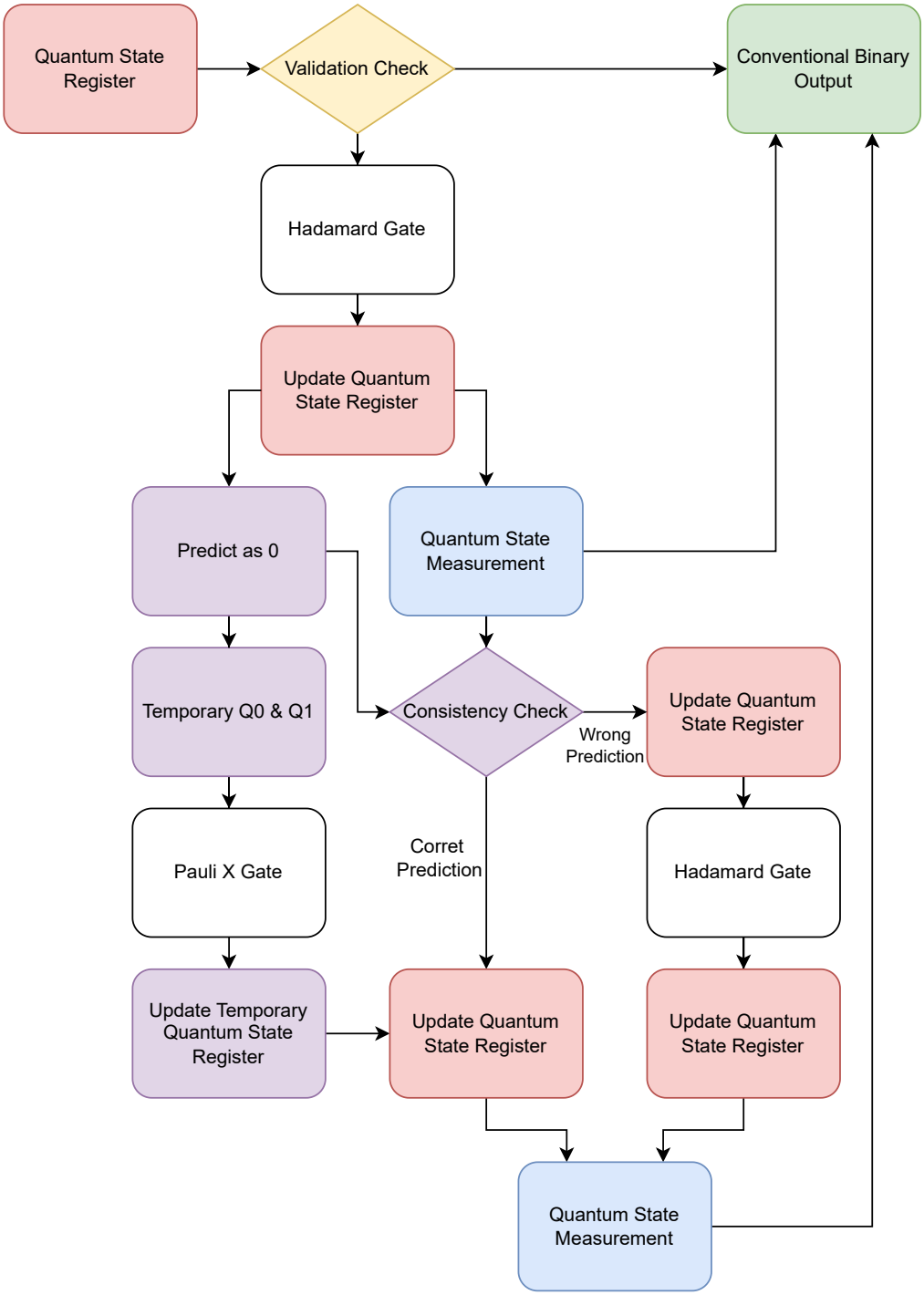

Figure 4.12: Emulation process flow of the dynamic quantum circuit example with midway measurement predicted as 0

The circuit was implemented using the emulation techniques stated in chapter 3. Figure 4.12 presents the comprehensive emulation process of the sample circuit, assuming the prediction result of Q_0 as 0. It could be observed that after each gate operation, the quantum state will be updated regardless of where the gate operation is performed. When the midway measurement of Q_0 occurs, the probabilistic execution predictor predicts the result of measurement based on the complex coefficient of the basis state of Q_0 stored in the quantum state register. In this case, the prediction is assumed as 0 for simplicity. Since the measurement of a qubit will change its state to a deterministic state of either $|0\rangle$ or $|1\rangle$, the Q_0 stored in the temporary quantum state register will be updated as $|0\rangle$. At the same time, the state information of Q_1 will be copied to the same place and applied to the Pauli-X Gate. The processed qubits will then wait for the practical measurement result to check the consistency with the prediction. If the prediction is correct, the qubits stored in the temporary state register will overwrite the regular quantum state register. Otherwise, the temporary qubit information will be eliminated, the Hadamard gate will be performed directly on the original qubits.

After being synthesized, the emulation is programmed on the Xilinx XC7A35T-1CPG236C chip. The emulation of this sample circuit requires 727 LUTS, 288 FFs and 12 DSPs. The DSPs are used for the comparator part of the quantum state measurement process and for realizing the multiplication in the quantum gate operation, especially the Hadamard gate. The FFs are responsible for storing the quantum state

information. In this context, the normal quantum state registers and the temporary quantum state registers each utilize half of the FFs. When a longer bits mantissa representation is applied to the quantum state coefficients, the number of FFs required will increase significantly. Specifically, if we change the architecture of the quantum state register from a 16-bit mantissa to a 32-bit mantissa, the number of FFs required would double.

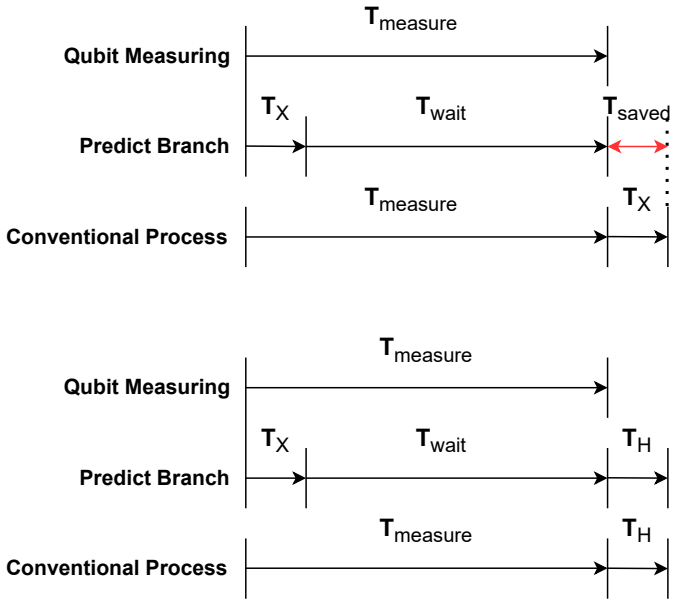

Figure 4.13: Timing schematic of correct (up) and wrong (down) prediction of the sample quantum circuit

Since the quantum gate operations on each quantum circuit branch is a single gate operation whose processing time is much shorter than that of the quantum state measurement

process, the time saved by the probabilistic execution predictor equals the processing time of the quantum gate. When referring to a timing schematic in Figure 4.13, and assuming that the predictor guesses the result is 0, the timing diagram would typically illustrate the sequence of processed operations and their transitions over time within the circuit. If the prediction is correct, 3.03% of the processing time would be saved.

Chapter 5

Conclusion and Future Work

When the midway measurement of a qubit is introduced, the quantum circuit becomes significantly complicated. Throughout this thesis, we proposed an FPGA-based emulation process that integrates quantum state storage, quantum gates, quantum state measurement and probabilistic execution predictor for such dynamic quantum circuits. Our work centers around the emulation of the quantum measurement and the midway measurement prediction. The RO-based RNG introduces true randomness to the measurement process and avoids the possible problem of deterministic results provided by pseudo-RNG. The probabilistic execution predictor saves the overall processing time when a conditional statement occurs within the quantum circuit. The proposed solution is a standalone system capable of exhibiting parallelism and probabilistic measurement.

The performance analysis reveals that the quantum state measuring method that utilizes

the RO-based true random number generator provides competent results compared to the LFSR-based pseudo RNG. The proportion of 0 among the random bits generated by an RO-based RNG with 4 stages and 5 inverters is 49.9966%, which meets the requirement of providing evenly distributed 0s and 1s. The probabilistic execution predictor also presents a high success rate, where the average miss rate for predicting the measuring result of a random qubit in the pure state could be as low as 25%. The time analysis of the predictor reveals that there's no extra penalty for the misprediction, while the time saved by the process is linearly related to the number of quantum gate operations on the following branches after the midway quantum state measurement.

5.1 Limitations and Future Work

This research mainly focused on the qubit measurement and probabilistic execution prediction. However, as mentioned in the background and the implementation of quantum gates, errors are generated through the process when multiplications occur. Quantum states are often damaged over time by several types of gate-specific and environmental errors, which experimental physicists find difficult to characterize. On the other hand, the hardware resource usage, especially the number of FFs and DSPs, would grow exponentially when longer mantissa is introduced to the quantum state representation. Thus, the errors should be quantized, and the corresponding impact could be analyzed. In this thesis, the errors generated throughout the emulation process are disregarded, further

work can focus on eliminating the errors raised by the gate process.

Another point that could be advanced is the multi-qubit midway measurement prediction. This work focused on the single-qubit measurement, where the measurement and prediction result is binary – it should be either $|0\rangle$ or $|1\rangle$. When a pair of entangled qubits need to be measured, there will be 4 possible results, thus the average miss rate of prediction will incredibly increase and the necessity of probabilistic execution prediction for such qubits needs to be considered. The extension to multiprocessors [36] will be considered as well.

Bibliography

[1] M. Khalid, U. Mujahid, A. Jafri, H. Choi, and N. u. I. Muhammad, "An FPGA-based hardware abstraction of quantum computing systems," *J. Comput. Electron.*, vol. 20, pp. 2001–2018, Oct. 2021.

[2] J. Pilch and J. Długopolski, "An fpga-based real quantum computer emulator," *Journal of Computational Electronics*, vol. 18, p. 329–342, Dec. 2018.

[3] P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pp. 124–134, 1994.

[4] B. P. Lanyon, J. D. Whitfield, G. G. Gillett, M. E. Goggin, M. P. Almeida, I. Kassal, J. D. Biamonte, M. Mohseni, B. J. Powell, M. Barbieri, A. Aspuru-Guzik, and A. G. White, "Towards quantum chemistry on a quantum computer," *Nature Chemistry*, vol. 2, p. 106–111, Jan. 2010.

[5] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, (New York, NY, USA), p. 212–219, Association for Computing Machinery, 1996.

[6] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, p. 195–202, Sept. 2017.

[7] G. Alagic, A. Broadbent, B. Fefferman, T. Gagliardoni, C. Schaffner, and M. St. Jules, *Computational Security of Quantum Encryption*, p. 47–71. Springer International Publishing, 2016.

[8] G. Alagic, T. Gagliardoni, and C. Majenz, *Unforgeable Quantum Encryption*, p. 489–519. Springer International Publishing, 2018.

[9] Z. Hu and S. Kais, "A quantum encryption design featuring confusion, diffusion, and mode of operation," *Scientific Reports*, vol. 11, Dec. 2021.

[10] T. Häner, D. S. Steiger, M. Smelyanskiy, and M. Troyer, "High performance emulation of quantum circuits," in *SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 866–874, 2016.

[11] Z.-Y. Chen, Q. Zhou, C. Xue, X. Yang, G.-C. Guo, and G.-P. Guo, "64-qubit quantum circuit simulation," *Science Bulletin*, vol. 63, no. 15, pp. 964–971, 2018.

[12] S. Mourya, B. R. L. Cour, and B. D. Sahoo, "Emulation of quantum algorithms using cmos analog circuits," *IEEE Transactions on Quantum Engineering*, vol. 4, pp. 1–16, 2023.

[13] A. Daskin, A. Grama, G. Kollias, and S. Kais, "Universal programmable quantum circuit schemes to emulate an operator," *The Journal of Chemical Physics*, vol. 137, Dec. 2012.

[14] G. G. Guerreschi, J. Hogaboam, F. Baruffa, and N. P. D. Sawaya, "Intel quantum simulator: a cloud-ready high-performance simulator of quantum circuits," *Quantum Science and Technology*, vol. 5, p. 034007, May 2020.

[15] A. D. Córcoles, M. Takita, K. Inoue, S. Lekuch, Z. K. Minev, J. M. Chow, and J. M. Gambetta, "Exploiting dynamic quantum circuits in a quantum algorithm with superconducting qubits," *Phys. Rev. Lett.*, vol. 127, p. 100501, Aug 2021.

[16] J. M. Pino, J. M. Dreiling, C. Figgatt, J. P. Gaebler, S. A. Moses, M. S. Allman, C. H. Baldwin, M. Foss-Feig, D. Hayes, K. Mayer, C. Ryan-Anderson, and B. Neyenhuis, "Demonstration of the trapped-ion quantum ccd computer architecture," *Nature*, vol. 592, pp. 209–213, Apr 2021.

[17] G. Q. AI, "Suppressing quantum errors by scaling a surface code logical qubit," *Nature*, vol. 614, pp. 676–681, Feb 2023.

[18] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Phys. Rev. A*, vol. 86, p. 032324, Sep 2012.

[19] R. Raussendorf and H. J. Briegel, "A one-way quantum computer," *Phys. Rev. Lett.*, vol. 86, pp. 5188–5191, May 2001.

[20] J. E. Sherwood, T. E. Stephenson, and S. Bernstein, "Stern-gerlach experiment on polarized neutrons," *Phys. Rev.*, vol. 96, pp. 1546–1548, Dec 1954.

[21] H. M. Wiseman and G. J. Milburn, *Quantum measurement and control*. Cambridge university press, 2009.

[22] Y.-S. Kim, J.-C. Lee, O. Kwon, and Y.-H. Kim, "Protecting entanglement from decoherence using weak measurement and quantum measurement reversal," *Nature Physics*, vol. 8, pp. 117–120, Feb 2012.

[23] E. Gutiérrez, S. Romero, M. A. Trenas, and E. L. Zapata, "Quantum computer simulation using the cuda programming model," *Computer Physics Communications*, vol. 181, no. 2, pp. 283–300, 2010.

[24] A. Avila, A. Maron, R. Reiser, M. Pilla, and A. Yamin, "Gpu-aware distributed quantum simulation," in *Proceedings of the 29th Annual ACM Symposium on Applied Computing*,

SAC '14, (New York, NY, USA), p. 860–865, Association for Computing Machinery, 2014.

[25] A. B. de Avila, R. H. S. Reiser, A. C. Yamin, and M. L. Pilla, "Scalable quantum simulation by reductions and decompositions through the id-operator," in *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, SAC '16, (New York, NY, USA), p. 1255–1257, Association for Computing Machinery, 2016.

[26] M. Sawerwain, "Gpu-based parallel algorithms for transformations of quantum states expressed as vectors and density matrices," in *Parallel Processing and Applied Mathematics* (R. Wyrzykowski, J. Dongarra, K. Karczewski, and J. Waśniewski, eds.), (Berlin, Heidelberg), pp. 215–224, Springer Berlin Heidelberg, 2012.

[27] A. Khalid, Z. Zilic, and K. Radecka, "Fpga emulation of quantum circuits," in *IEEE International Conference on Computer Design: VLSI in Computers and Processors, 2004. ICCD 2004. Proceedings.*, pp. 310–315, 2004.

[28] Y. Goto and M. Fujishima, "Efficient quantum computing emulation system with unitary macro-operations," *Japanese Journal of Applied Physics*, vol. 46, p. 2278, apr 2007.

[29] Y. H. Lee, M. Khalil-Hani, and M. N. Marsono, "An fpga-based quantum computing emulation framework based on serial-parallel architecture," *International Journal of Reconfigurable Computing*, vol. 2016, p. 5718124, Apr 2016.

[30] M. Aminian, M. Saeedi, M. S. Zamani, and M. Sedighi, "Fpga-based circuit model emulation of quantum algorithms," in *2008 IEEE Computer Society Annual Symposium on VLSI*, pp. 399–404, 2008.

[31] G. Negovetic, M. Perkowski, M. Lukac, and A. Buller, "Evolving quantum circuits and an FPGA-based quantum computing emulator," 2002.

[32] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca, "Quantum algorithms revisited," *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 454, p. 339–354, Jan. 1998.

[33] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[34] K. Fang, M. Zhang, R. Shi, and Y. Li, "Dynamic quantum circuit compilation," *arXiv e-prints*, pp. arXiv–2310, 2023.

[35] various authors, *Qiskit Textbook*. Github, 2023.

[36] A. Grbic, S. Brown, S. Caranci, R. Grindley, M. Gusat, G. Lemieux, K. Loveless, N. Manjikian, S. Srbljic, M. Stumm, Z. Vranesic, and Z. Zilic, "Design and implementation of the NUMAchine multiprocessor," in *Proceedings 1998 Design and Automation Conference. 35th DAC. (Cat. No.98CH36175)*, pp. 66–69, 1998.