FPGA Based Hardware Abstraction of Quantum Computing System

Madiha Khalid
Bahria University
Najam ul Islam MUHAMMAD (najam@bahria.edu.pk)
Bahria University https://orcid.org/0000-0003-1365-5279
Umar Mujahid Khokhar
Georgia Gwinnett College
Atif Jafri
Bahria University
Hongsik Choi
Georgia Gwinnett College

Research Article

Keywords: FPGA emulation, quantum abstraction, parallelism, probabilistic measurement

Posted Date: August 23rd, 2021

DOI: https://doi.org/10.21203/rs.3.rs-467244/v1

Noname manuscript No.
(will be inserted by the editor)

FPGA Based Hardware Abstraction of Quantum Computing System

Madiha Khalid · Umar Mujahid · Atif Jafri · Hongsik Choi · M. Najam-ul-Islam

Received: date / Accepted: date

Abstract The number of transistors per unit area are increasing every year by the virtue of Moore's law. It is estimated that the current rate of evolution in the field of chip design will reduce the size of transistor to an atomic scale by 2024. At atomic level the quantum mechanical characteristics dominate, effecting the ability of transistors to store information in the form of bits. The quantum computers have been proposed as one way to effectively deal with this predicament. The quantum computing circuits utilize the spinning characteristics of the electron to store information. This paper describes a proposition of resource efficient FPGA based quantum circuit abstraction. A non-programmable embedded system capable of storing, measuring and introducing a phase shift in the *qubit* is implemented. The main objective of the proposed abstraction is to provide the FPGA based platform comprising of fundamental sub-blocks for designing the quantum circuits. A primary quantum key distribution algorithm i.e. BB84 is implemented on the proposed platform as a proof of concept. The distinguishing feature of the proposed design is the flexibility to enhance the quantum circuit emulation accuracy at the cost of computational resources. The proposed emulation exhibits two principal properties of the quantum computing i.e. parallelism and probabilistic measurement.

Madiha Khalid
Department of Electrical Engineering, Bahria University, Islamabad, Pakistan
Tel.: +92-323-5519362
Fax: +92-51-9260885
E-mail: madihazoheb.buic@bahria.edu.pk *Present address:* of F. Author

Umar Mujahid
Department of Information Technology, Georgia Gwinnett College, Lawrenceville, USA

Keywords FPGA emulation · quantum abstraction · parallelism · probabilistic measurement

1 Introduction

In 1965, Gordon Moore observed that the number of transistors per unit area had doubled every two years since 1959. He predicted that based on the pace of technological innovations, this trend would continue for at least a decade [1]. The growth trend defined by the Moore's law persisted 50 years longer than he predicted. Since 1960s, multiple semiconductor technologies have been implemented to increase the transistor density and the switching speed. Some of the eminent technologies are MOSFETs, Piezo-Electric Transistors (PETs), Tunneling Field Effect Transistors (TFETs) and Near-Threshold Voltage (NTV)[2]. The photolithography is recently being introduced to further reduce the transistor fabrication area. At the current rate of improvement, the photolithography system will be able to use $5nm$ technology to create transistor features on the scale of the handful of atoms by 2024 [3].
In atomic-scale transistors, the quantum mechanical characteristics of electrons dominate to cause the tunneling effect. Thus, allowing a current to flow in the reverse biased mode of the transistor. Therefore, the quantum tunneling effect in transistors at nano-scale is assumed as the prime cause to end the Moore's law. The quantum computers have been proposed as one way to effectively deal with this predicament. The quantum processors utilize the spinning characteristics of the electrons as demonstrated in the Stern-Gerlach experiment to store information [4]. A single electron with spin i.e. *qubit* constitutes the smallest building block of a quantum computer . A brief comparison between the

conventional and the quantum computers is given in table 1.

The companies like Google, IBM, Hitachi and D-wave

Table 1: Comparison between the conventional and the quantum computers

Characteristics	Conventional Computer	Quantum Computer
Information Storage	bit	*qubit*
Circuit Bahaviour	Classical Physics	Quantum Mechanics
Building Blocks	Transistors	Superconducting Quantum Interference Devices (SQID)
States	$0, 1$	0, 1, *superposition of* 0 *and* 1
Single Input Gates	*not*	*Pauli* $-X$, *Pauli* $-Y$, *Pauli* $-Z$, *Identity*, H *Gate*
Two Input Gates	and, or, xor	*Controlled not Gate*

are working to develop a fully functional and stable quantum computer. However due to the challenges like decoherence, imperfect isolation from the environment, limitations of the capability to measure the output from the *qubit* and robust representation of the quantum information; a true quantum computer has not been developed yet[5][6]. The technologies that are being used for the quantum computer hardware architecture development are summarized in table 2.

The quantum computers are ideally suited for solving

Table 2: Quantum computing enabling technologies

S no	Technology	*qubit* representing entity
1	Nuclear Magnetic Resonance (NMR)	The vial of liquid filled with sample molecules [7]
2	Ion-trap-based Computer	Single ion constitutes controlled by the laser beams[8]
3	Cavity Quantum Electrodynamics (QED)	The photon[9]
4	Linear Optics Quantum Computer	Optical mode of the photon[10]
5	Quantum-dot-based Computer	Spin of the electron[11]

computationally expensive problems. Fundamental examples proving the quantum computing precedence to the conventional computers are Shor's algorithm and Grover's search algorithm. The Shor's algorithm can factor a 512-bit number in 3.5 hours with $1GHz$ clock rate[12], whereas 8400 years are required by the conventional computer working with the speed of million instructions per second[13]. The Grover's search algorithm can search an element from an unsorted list of N elements in $O(\sqrt{N})$ time[14]. Some of the advanced examples of the quantum computing algorithms from a variety of disciplines are as follows:

1. *Intelligent Control Systems:* The knowledge-based optimizers with quantum computing, incorporated with the structure of intelligent control system has been proven effective for solving multicriteria control problems[15].
2. *Cryptanalysis Techniques*: Fuchs et.al explored the quantum analogous for the classical measure of distinguishably for probability distribution [16]. This concept facilitates the security analysis of the quantum key distribution algorithms.
3. *Cryptography:* The security of quantum cryptography suites is based on the non-cloning property of the *qubits* being transmitted over the wireless channel. The non-cloning property allows information transmission through the non-orthogonal states only. Goldenberg et.al introduced the idea of encryption by using orthogonal states without compromising the security of the transmitted message [17].
4. *Image Processing:* The representation of an image on a quantum computer in the form of normalized states, facilitate the solution of numerous image processing problems. The prominent quantum computing based image processing techniques are Flexible Representation of Quantum Images (FRQI) [18], quantum watermarking [19] [20] and quantum ghost imaging[21].
5. *Artificial Intelligence:* Quantum Inspired Neural Networks (QINN) can be classified in two categories [22]. The class of QINN which explicitly uses the concepts from quantum computing remains at a theoretical level as it requires a functional quantum computer to be implemented. The second category includes models of biological neural networks explaining the exceptional performance of the biological brains by employing concepts from the quantum computing and the quantum mechanics.

The idea of using quantum circuits to perform the computational tasks has been prevailing for over 30 years. The non-availability of a fully functional quantum computer is impeding the implementation of the quantum algorithms on a useful scale. This research gap drives the focus on the concept of quantum computer hardware/software abstractions [5].

The concept of hardware/software abstraction of the quantum circuits was introduced at the beginning of the 20^{th} century. The quantum computing emulations are designed by using Graphics Processing Unit (GPU), multiprocessor systems, super computers and FPGA. The FPGA based system by the leverage of parallelism provides emulation in a time-efficient manner. In this paper, a hardware abstraction of the quantum computing system is implemented on the FPGA. The proposed quantum gate emulation circuit provides a platform to develop and test the algorithms based on the quantum computing principles.

The organization of the paper is as follows: section 2 presents a literature review on the quantum computing principles. In section 3, the architecture of the proposed Single Input Single Output (SISO) quantum abstraction is presented followed by the performance analysis in section 4. Section 5 demonstrates the implementation of a quantum cryptographic algorithm i.e. BB84 based on the cascaded SISO sub-block. The conclusion explores the connection between the *qubit* size and the emulation accuracy in section 6.

2 Literature Review

In the conventional computing, a bit is the smallest unit of information describing a classical system. Whereas, the *qubit* is the basic unit of information in a quantum system. The Stern- Gerlach experiment defines the concept of the *qubit* with the help of an electron spin [23] . The experimental setup consists of shooting a beam of electrons through non-homogeneous magnetic field oriented along the $z-axis$. The field splits the beam into two streams with opposite spins i.e. positive $z-axis$ or the up spin ($|\uparrow\rangle$) and negative $z-axis$ or the down spin ($|\downarrow\rangle$). The mathematical model of the experiment states that an electron exists in the superposition (a combination of ($|\uparrow\rangle$) and ($|\downarrow\rangle$)) and the probability of an electron to follow up or down path is $|\alpha_0|^2$ and $|\alpha_1|^2$ respectively. An electron spin is described as follows:

$$spin(e) = \alpha_0|\uparrow\rangle + \alpha_1|\downarrow\rangle \tag{1}$$

$$|\alpha_0|^2 + |\alpha_1|^2 = 1 \tag{2}$$

In the digital computers, a bit represents two states i.e. 0 and 1. In quantum computing, a 2×1 matrix is used to define a *qubit*. The conventional states analogous are represented as follows:

$$0\,qubit = |0\rangle = \begin{bmatrix}1\\0\end{bmatrix} \tag{3}$$

$$1\,qubit = |1\rangle = \begin{bmatrix}0\\1\end{bmatrix} \tag{4}$$

A *qubit* as defined in the equation (1) exists as a combination of orthonormal states known as the superposition. The mathematical model for superposition is given in equation (5).

$$\begin{aligned} |\psi\rangle &= \begin{bmatrix}C_0\\C_1\end{bmatrix} = C_0\begin{bmatrix}1\\0\end{bmatrix} + C_1\begin{bmatrix}0\\1\end{bmatrix} \\ |C_0|^2 &= probability\,of\,|\psi\rangle\,to\,assume\,|0\rangle\,state \\ |C_1|^2 &= probability\,of\,|\psi\rangle\,to\,assume\,|1\rangle\,state \end{aligned} \tag{5}$$

where C_0 and C_1 are complex numbers.

The graphical representation of the *qubit* is a unit sphere termed as the Bloch sphere. The bloch sphere maps complex numbers C_0 and C_1 on a spherical coordinate system. The *qubit* is first translated into 3-dimensional rectangular space and then on a unit sphere. The mathematical expressions for the transformations are as follows:

Consider a *qubit* $|\psi\rangle$, defined as the superposition of $|0\rangle$ and $|1\rangle$

$$|\psi\rangle = \begin{bmatrix}C_0\\C_1\end{bmatrix} = C_0\begin{bmatrix}1\\0\end{bmatrix} + C_1\begin{bmatrix}0\\1\end{bmatrix} \tag{6}$$

$$|\psi\rangle = \begin{bmatrix}C_0\\C_1\end{bmatrix} = r_{C_0}e^{j\phi C_0}\begin{bmatrix}1\\0\end{bmatrix} + r_{C_1}e^{j\phi C_1}\begin{bmatrix}0\\1\end{bmatrix} \tag{7}$$

The equation (7) represents the complex coefficients of the *qubit* in a polar form. Since the *qubit* does not change if multiplied with a unit magnitude complex number, we take product of the equation (7) with $e^{j\phi c_0}$. Following equation gives the result of multiplication.

$$|\psi\rangle = \begin{bmatrix}C_0\\C_1\end{bmatrix} = r_{C_0}\begin{bmatrix}1\\0\end{bmatrix} + r_{C_1}^{j\phi(C_1-C_0)}\begin{bmatrix}0\\1\end{bmatrix} \tag{8}$$

$$\begin{aligned} |\psi\rangle = \begin{bmatrix}C_0\\C_1\end{bmatrix} &= r_{C_0}\begin{bmatrix}1\\0\end{bmatrix} + r_{C_1}^{j\phi}\begin{bmatrix}0\\1\end{bmatrix} \\ assumption\,is\,\phi &= \phi(C_1 - C_0) \end{aligned} \tag{9}$$

Implementing the Euler's identity to expand $e^{j\phi}$.

$$|\psi\rangle = \begin{bmatrix}C_0\\C_1\end{bmatrix} = r_{C_0}\begin{bmatrix}1\\0\end{bmatrix} + r_{C_1}(cos\phi + jsin\phi)\begin{bmatrix}0\\1\end{bmatrix} \tag{10}$$

Since *qubit* is a normalized vector, for the translation of such a vector onto unit sphere; the unit magnitude condition is applied on the *qubit* under observation as depicted in equation (11). In equation (12), the *qubit* is represented in spherical coordinate system ,$r_{C_0} = cos\theta$ is the projection of vector $|\psi\rangle$ on the $\theta = 0^0; \phi = 0^0$ line and $r_{C_1} = sin\theta$ is the shadow of *qubit* under discussion on $\theta = 90^0$ plane.

$$\begin{aligned} \sqrt{[(r_{C_0})^2 + (r_{C_1})^2]} &= \sqrt{[(cos\theta)^2 + (sin\theta)^2]} = 1 \\ r_{C_0} &= cos\theta \\ r_{C_1} &= sin\theta \end{aligned} \tag{11}$$

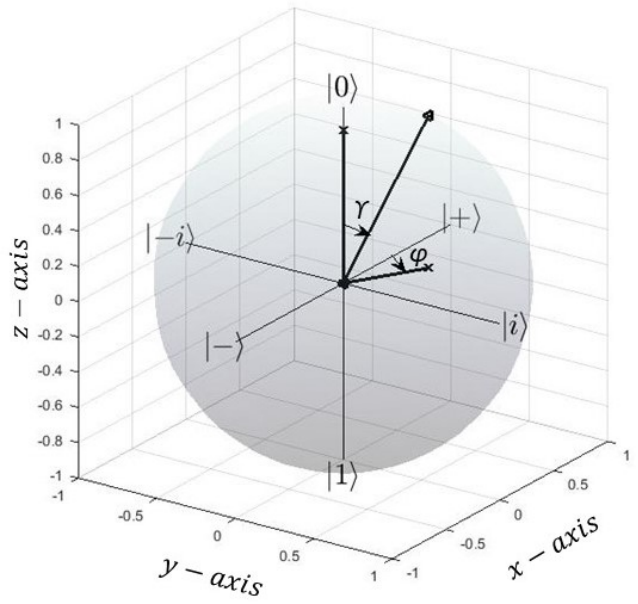


Fig. 1: Bloch sphere transformation

$$|\psi\rangle = \begin{bmatrix} C_0 \\ C_1 \end{bmatrix} = cos\theta \begin{bmatrix} 1 \\ 0 \end{bmatrix} + sin\theta(cos\phi + jsin\phi) \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (12)$$

Equation (13) shows the transformation equations from the *qubit* representation to the rectangular system.

$$x = sin\theta cos\phi;\ y = sin\theta sin\phi;\ z = cos\theta \quad (13)$$

In order to make the $|0\rangle$ and $|1\rangle$ *qubits* 180^0 apart, the θ in equation (12) moves at half speed which can be mathematically represented by multiplying θ with two $(2\theta = \gamma)$. The equation (14) represents the *qubit* on to the bloch sphere.

$$x = sin\gamma cos\phi;\ y = sin\gamma sin\phi;\ z = cos\gamma \quad (14)$$

The graphical depiction of *qubit* representation on the bloch sphere and the rectangular coordinates system is given in figure 1.

In the electron beam splitting experiment, the asymmetric magnetic field is used to define the spin of an electron. Analogous to the Stern- Gerlach experiment, in quantum computing the measurement procedure transforms the *qubit* into a classical bit. From the bloch sphere point of view, the projection of *qubit* on positive $z - axis$ depicts its chances to be measured as a zero bit. Similarly, the projection of the *qubit* vector on the negative $z - axis$ shows its probability to be measured as one bit. The classical Probabilistic Turning Machine (PTM) is used to express the concept of measurement. In the PTM tree, the vertex shows the state and the edges define the probability of the transition occurrence. In figure 2, the level 1 vertex is a state of the *qubit* and the level 2 vertices define the states in which the electron will collapse after transition based on the probabilities defined by the edges.

In classical computing, the logic gates are the fundamental building blocks of the digital integrated circuits. These gates implement boolean functions on the classical bits. The fundamental classical logic gates are and, or, xor and not. In the quantum world, the reversible matrix functions are analogous to the classical logical gates [24]. A quantum gate is formally defined as a unitary function that acts on the *qubits*. The quantum gates can be classified into three categories i.e. Signal Input (SI) operations, Double Input (DI) operations and Multiple Input (MI) operations. The SI operations add a phase shift to a *qubit* on the bloch sphere thus varying the probability of the *qubit* to assume $|0\rangle$ or $|1\rangle$ once measured. The remaining two categories are collectively termed as controlled - U gates. In controlled-U gates, the first input controls the nature of operation on the remaining inputs [25]. The quantum gates and their effects on the *qubits* are given in table 3 [4].

In quantum information theory, a quantum circuit model is used to describe the quantum computations and algorithms as a sequence of quantum gates; which are reversible transformations on the $n - qubit$ register. The universal set of the quantum gates capable of performing any unitary transformation on the $n - qubits$ are $Controlled - NOT$ gate and single-*qubit* gates[26]. The fundamental features of the quantum mechanics demonstrated by the unitary transformations are parallelism, probabilistic measurement and entanglement. The SI quantum gates can perform parallelism and measurement whereas entanglement is demonstrated by the $Controlled - NOT$ gate. In the absence of fully functional quantum computers, the emulation of the quantum gates and the *qubit* registers provide a platform for the design and the development of quantum circuit models. Table 4 elaborate prominent hardware abstrac-

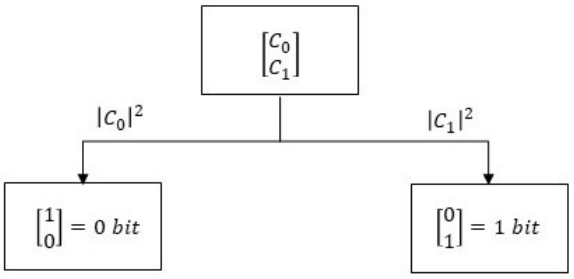


Fig. 2: Probabilistic Turning Machine for *qubit* measurement

Table 3: Fundamental quantum gates

Single Input (SI) gates	
Gates	**Functionality**
measurement	Coverts *qubit* into classical bit
$Pauli - X$	Flip the Bloch sphere 180^0 along x-axis
$Pauli - Y$	Flip the Bloch sphere 180^0 along y-axis
$Pauli - Z$	Flip the Bloch sphere 180^0 along z-axis
$Hadamard$	90^0 rotation along y axis followed by 180^0 rotation about x axis
Double Input (DI) gates-inputs (x, y)	
Gates	**Functionality**
Controlled-NOT gate	$\|x, y\rangle \Rightarrow \|x, (x \oplus y)\rangle$
Multiple Input (MI) gates-inputs (x, y, z)	
Gates	**Functionality**
Toffoli gate	$\|x, y, z\rangle \Rightarrow \|x, y, z \oplus (x \wedge y)\rangle$
Fredkin gate	$\|0, y, z\rangle \Rightarrow \|0, y, z\rangle$; $\|1, y, z\rangle \Rightarrow \|0, z, y\rangle$

tion circuits available in the literature.

The first two emulation approaches defined in table 4 involve the abstraction of quantum algorithms however the third approach is based on the development of quantum circuit building blocks. This approach facilitates the quantum circuit design at the abstraction level by cascading the quantum gates and registers. In this paper, the architecture of the hardware abstraction for the single input quantum system is presented. This design is capable of storing *qubit*, introducing a phase shift in the *qubit* and its measurement. The main objective of the proposed abstraction is to provide an FPGA based platform comprising of the fundamental sub-block for designing quantum circuits. The distinguishing feature of our design is the flexibility to enhance the quantum circuit emulation accuracy at the cost of computational resources.

3 Architecture of Single Input Single Output Quantum System

An ultra-lightweight quantum abstraction can enhance the computational capabilities of the classical passive devices. With the Single Input Single Output (SISO) quantum emulator, the data processing and the security of resource constraint communicating devices can be improved compared to the conventional $8, 16, 32$ or 64 bit processors. This section presents a complete architecture of the quantum emulator. The proposed architecture is implemented on FPGA using ISE and the resource requirement of each block is given in terms of FPGA Look-Up Tables (LUTs) and slice count. A quantum gate-level abstraction primarily consists of five components i.e. the *qubit* flip flop, measurement block, Arithmetic Logic Unit (ALU), Counter and Quantum Finite State Machine (QFSM) [35]. Figure 3 presents the elaborated block diagram of the proposed SISO emulator. The QFSM along with the counter controls the sequence of operations for the *qubit* processing. The *qubit* processing involves fetching of the quantum information, application of unitary transformation on the *qubit* by the ALU, writing the ALU output in the *qubit* flipflop and conversion of the ALU results to the classical bit via measurement block.

The design also provides flexibility to improve the emulation results accuracy by increasing the quantized superposition states of the *qubits*. This is achieved by increasing the number of bits used for the *qubit* representation. For demonstration purposes the abstraction working is defined with respect to 8-bit model for the *qubit* representation. The following subsections elaborate the working principle of each function block in the SISO architecture.

3.1 *Qubit* Flip Flop

The *qubit* flip flop allows the abstraction to store or fetch a *qubit* as per instructions from the Quantum Finite State Machine (QFSM) in the form of read/write signals. The mathematical representation of the *qubit* is given in equation (5). In this equation C_0 and C_1 are complex numbers with the condition $|C_0|^2 + |C_1|^2 = 1$. Graphically all the points on the bloch sphere represents possible values of a *qubit*. Ideally, a *qubit* has infinite number of states as C_0 and C_1 has infinite possible values between absolute values $1 + 0i$ and $-1 + 0i$.

In the abstraction model, we need to store C_0 and C_1 in order to create a memory for a *qubit*. Since coefficients of the *qubit* are complex numbers, two eight-bit fixed point values are used to store each quantity. Consider the coefficient C_0, it is represented as $\alpha_0 + \beta_0 i$ where α_0 and β_0 are represented as $Q(2, 6)$ signed numbers. The reason for dedicating 2 bits for integer part is to include the north and south pole of the bloch sphere in the given range. By dedicating six bits for the fractional part representation, we are limiting the superposition states that can be assumed by the *qubit*. There is a direct relation between accuracy of the *qubit* abstraction and the number of fraction bits dedicated to represent

Table 4: Quantum abstraction comparative analysis

Emulation Approach	Architecture	Description
Emulation of quantum algorithms focused on efficient time and hardware resource usage	Fujishima et al. [27] [28]	Memory efficient architecture of quantum search-based algorithms.
	Franka et al. [29]	Resource efficient design of quantum algorithms
	Goto and Fujishima [30]	Use of unitary macro-operations to facilitate memory efficient simulation of quantum circuits
Behavioral emulation of physical quantum algorithms	Lee et al. [31]	Quantum algorithm emulation closer to natural quantum systems at the cost of increased hardware complexity
Design of basic quantum gates for developing quantum algorithms on classical architecture	Aminian et al. [32]	The proposed approach is focused on development of quantum gates and using them to emulate quantum algorithms.
	Negovetic et al.[33]	The paper proposes a software-hardware solution for quantum circuit emulation
	Khalid et al. [34]	Quantum circuit development encompassing the concept of entanglement and parallelism

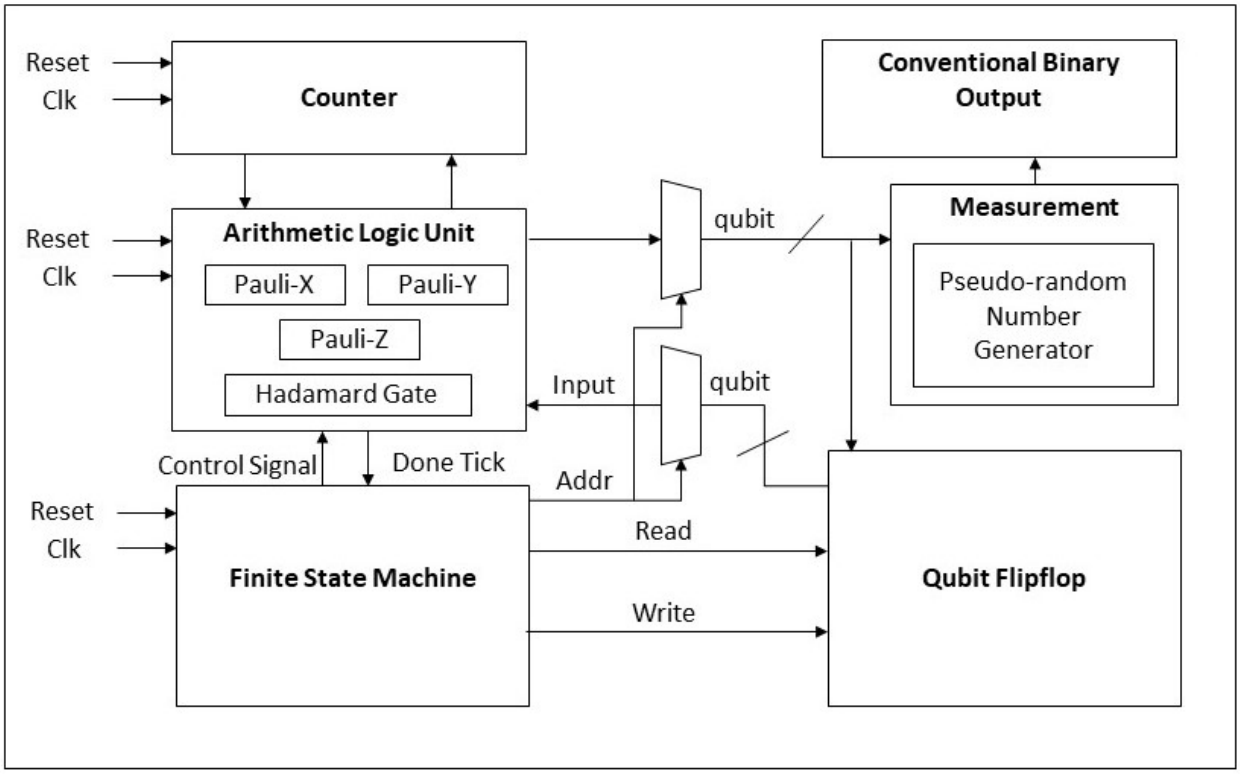


Fig. 3: The SISO Quantum gate emulator

real and imaginary part of the complex *qubit* coefficients.

The accuracy of the abstraction of a *qubit* is also effected when a *qubit* is processed by a quantum gate. Each gate involves addition and multiplication of complex numbers. Since the gate output is stored in $Q(2,6)$ memory, an error is introduced by truncating the fractional part of the gate's output. Figure 4 shows the architecture of single-*qubit* storage.

3.2 Measurement Block

The concept of an electron spin in quantum mechanics allows the *qubit* to be in a coherent superposition of the absolute states i.e. $|0\rangle$ and $|1\rangle$. The north and south pole of the bloch sphere can be considered analogous to the classical 0 and 1 bit respectively. The measurement block transforms the *qubit* coherent superposition to the discrete 0 or 1 bit based on the probabilistic model. The equation (5) represents that the probability of a *qubit* to land as a $|0\rangle$ or 0 bit is $|C_0|^2$ and probability to assume the $|1\rangle$ state is $|C_1|^2$. In the proposed design, the measurement takes place on the principle of Roulette wheel [35]. The following procedure describes the computational steps performed to convert a *qubit* to a classical bit:

1. The probability of measuring 0 ($|C_0|^2$) is transformed from fixed point number ranging between $0-1$ to the integer number $ket0\_p$ within the range $0-255$ via linear mapping.
2. An 8 bit pseudo-random number R_p is generated that is unsigned integer in nature.
3. If the pseudo-random number R_p is less than $ket0\_p$, the *qubit* is transformed to the classical bit 0 else the output is a conventional bit 1.

Figure 5 presents the architecture of the measurement block. The function used for the pseudo-random number generation that facilitates resource efficient measurement of a *qubits* is termed as Permutation based Shuffling (PbS) function. The PbS function is inspired from the primitive used in an ultra-lightweight MAC protocol [36].

The PbS function accepts two 16-bit numbers (m and n) as an input and it produces two 8-bit pseudo-random numbers ($c1$ *and* $c2$). The PbS function re-positions the bits of input m based on the corresponding bit values of the input n to form the output c_{inter} and for the final answer the c_{inter} is *xored* with the input n. The mechanism for shuffling of input m in the PbS function is elaborated in algorithm 1. For the next set of random numbers, the PbS function is executed with inputs i.e. $m = [c1, c2] = c$ and $n = c_{inter}$.The flags p_{zero} and p_{one} are responsible for toggling the bits of m between the lsb and the msb side of c_{inter}.

The architecture of the PbS based PRNG consists of six sub-modules that work sequentially for 16 cycles to create c_{inter} and the final output c is generated by 7^th sub-module that gives the *xored* version of c_{inter} and n. The registers used in the architecture and their purposes are given in table 5.

Step by step description of each stage of PbS function architecture is elaborated in figure 6 and is narrated as follows:

1. As a first step, the inputs m and n are stored in registers m_reg and n_reg respectively and also the most significant bits of m_reg and n_reg are stored in as m_msb and n_msb. For the bit by bit manipulation, both registers are left rotated so that the msb now represents the subsequent bits for the next cycle.

Algorithm 1: Algorithm for the random number generation

```
Result: N/2 bits pseudo-random numbers [c1, c2]
initialization;
i ← 0; j ← (N − 1) ;
p_one ← 0 ; p_zero ← 0;
ptr ← (N − 1);
for a ← (N − 1) to 0 do
    if n[ptr] = 0 then
        if p_zero = 0 then
            c_inter[i] = m[ptr];
            i = i + 1;
            ptr = ptr − 1;
            p_zero = 1;
        else
            c_inter[j] = m[ptr];
            j = j − 1;
            ptr = ptr − 1;
            p_zero = 0;
        end
    else
        if p_one = 0 then
            c_inter[j] = m[ptr];
            j = j − 1;
            ptr = ptr − 1;
            p_one = 1;
        else
            c_inter[i] = m[ptr];
            i = i + 1;
            ptr = ptr − 1;
            p_one = 0;
        end
    end
end
[c1, c2] = c_inter xor n;
return [c1, c2];
```

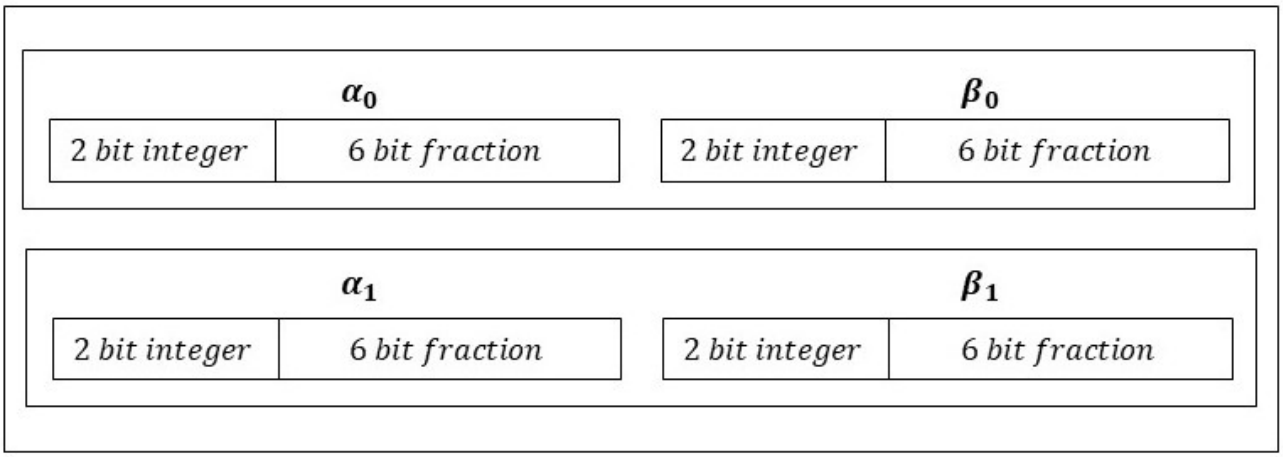


Fig. 4: Register file architecture for the single *qubit* storage

Table 5: Register definition for the PbS function architecture

Register	**Purpose**
m_reg	Stores input m
n_reg	Stores input n
m_msb	Most significant bit (msb) of m_reg
n_msb	Most significant bit (msb) of n_reg
p_one	Flag that determines m_msb position in c_{inter} if $n\_msb = 1$. (Initial value = 0)
p_zero	Flag that determines m_msb position in c_{inter} if $n\_msb = 0$. (Initial value = 0)
$count1$	Pointer that moves along c_inter from msb side. (Initial value = 15)
$count0$	Pointer that moves along c_inter from lsb side. (Initial value = 0)

2. In this step, the sel_p flip flop assumes the value of the flag p_one or p_zero based on the value of n_msb. In terms of implementation, a $2 \times 1\, mux$ is used with the selection line n_msb and the inputs p_one and p_zero. The table 6 presents the output of stage 2.
3. This step is responsible for updating the value of the flags so that the m_msb is placed in c_inter as defined in algorithm 1. The values of p_one and p_zero are evaluated for the next cycle based on the bit value of n_msb. If $n\_msb = 0$, the value of p_zero is toggled otherwise p_one is flipped. This functionality of the flag update is implemented with the help of two 2×1 multiplexers.
4. In step 4, a $4 \times 2\, mux$ is defined to identify the position in c_inter to place m_msb. The output of the mux i.e. $pointerc$ assumes the value of $count1$ or $count0$ based on the value of selection line [n_msb, sel_p]. The table 7 defines output of the mux based on the selection line.

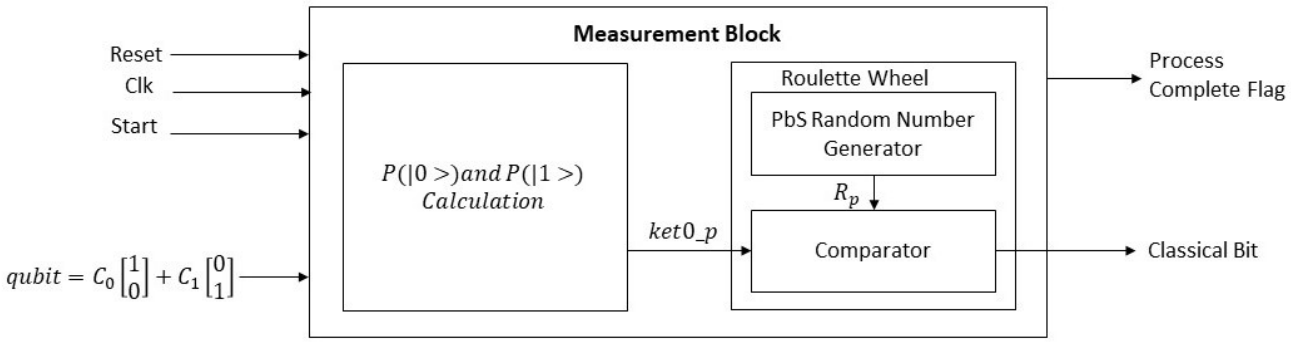


Fig. 5: Architecture of Measurement Block

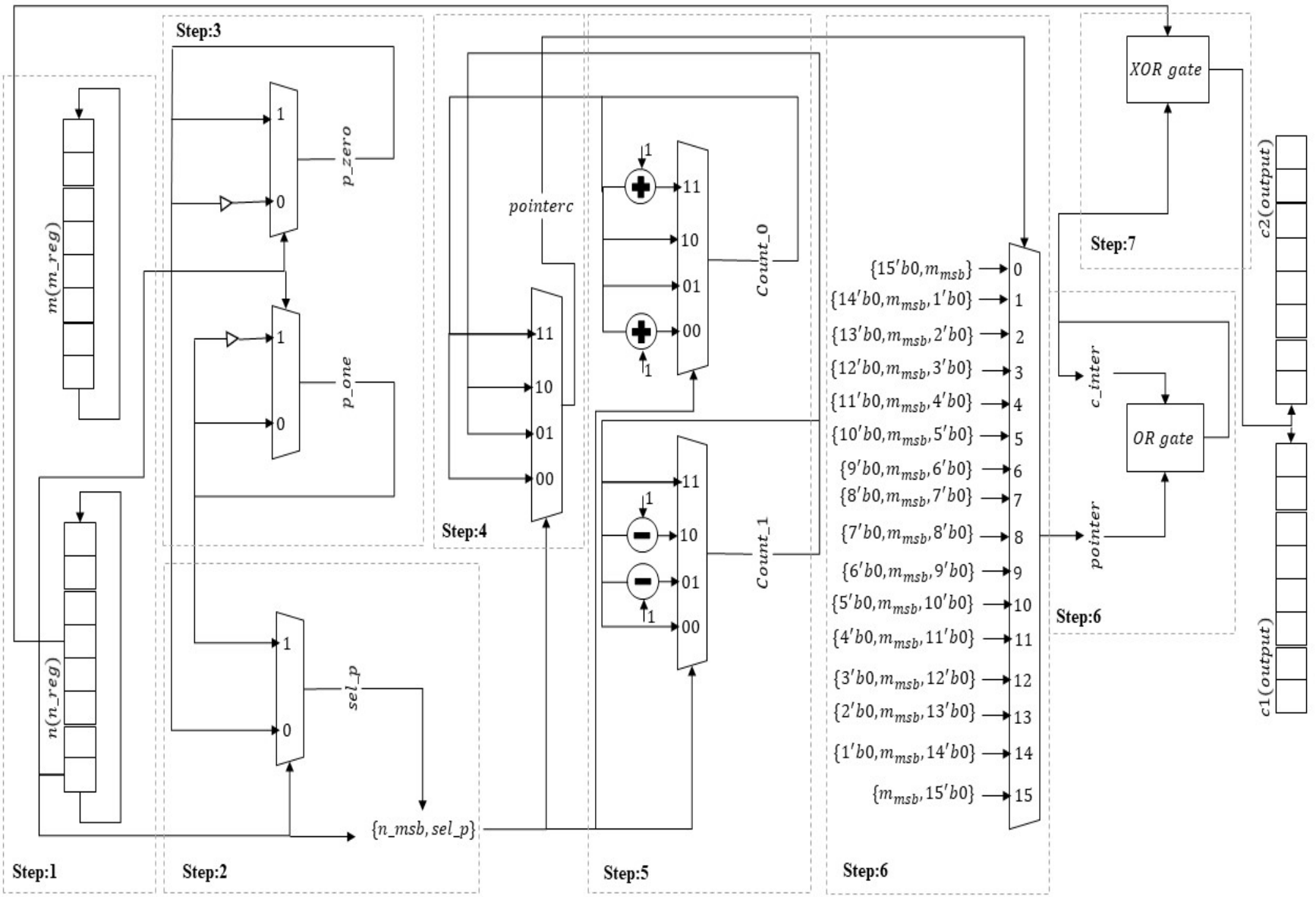


Fig. 6: Architecture of PbS function Based Pseudo Random Number Generator

Table 6: PbS function stage 2 output

Selection line	$2x1$ **mux output**
$n\_msb = 0$	p_zero
$n\_msb = 1$	p_one

Table 7: PbS function stage 4 output

n_msb, sel_p	4×2 **mux output**
00	$count0$
01	$count1$
10	$count1$
11	$count0$

5. The step five updates the values of $count0$ and $count1$ for the next bit placement. This stage is implemented with the help of two multiplexers controlled by selection line n_msb, sel_p. The functionality of the block is summarized in table 8.

Table 8: PbS function Stage 5 output

n_msb, sel_p	$count0(InitialValue = 0x0)$	$count1(InitialValue = 0xF)$
00	$count0 + 1$	$count1$
01	$count0$	$count1 - 1$
10	$count0$	$count1 - 1$
11	$count0 + 1$	$count1$

6. In this step, the m_msb is actually placed at the position defined by $pointerc$ defined in step 4. The above stated six steps are repeated for 16 cycles to completely shuffle the bits of m_reg and form c_inter.
7. As a final step, the output is obtained by taking xor of c_inter and n_reg. This gives a 16 bit output that is divided into two 8 bit PRNs.

The block diagram of the architecture of PbS based pseudo random number generator is presented in figure 6. The statistical test suite developed by National Institute of Standards and Testing (NIST) is an international merit to evaluate various aspects of randomness in a long sequence of bits[37]. In order to verify the efficiency of the proposed random number generator, the output of PbS function is subjected to NIST randomness test suit. Appendix A shows that the proposed PbS based random number generator creates 8-bit random number as per NIST standards as p-value for each test is well within the upper bound defined by randomness test suite [38].

3.3 Arithmetic Logic Unit

The *qubit* signal processing is defined by applying a set of transformations on the quantum state space. These transformations are termed as the quantum gates. In classical computing, the basic gates are and, or, not and xor. Any computation on conventional bits can be simplified into a sequence of basic operations.
In quantum computations, any algorithms can be broken down into a series of the quantum gates. Mathematically a quantum gate can be represented as a unitary matrix and graphically the transformations correspond to a phase shift of *qubit* on the bloch sphere. These gates can be classified into two categories i.e. Single Input (SI) operations and Multiple Input (MI) operations. A brief description of quantum transformation categories are as follows:

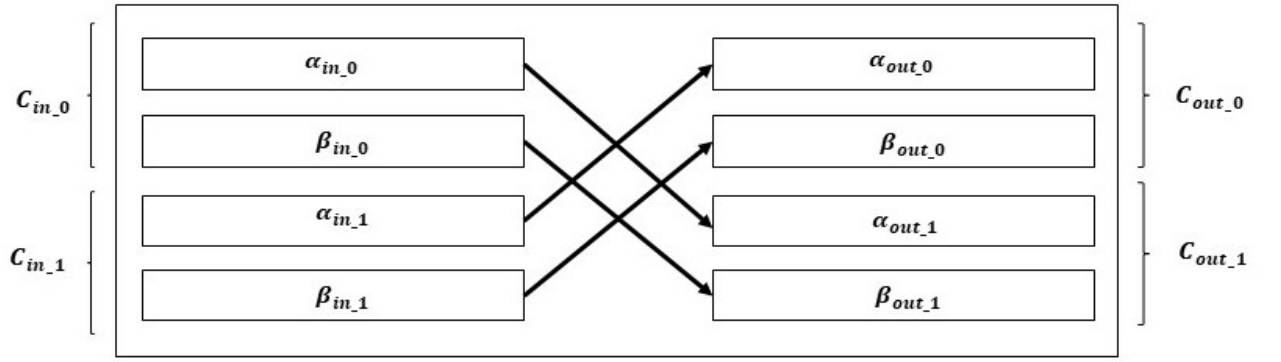
- *Single Input (SI) Operations:* These gates accept single *qubit* as an input and alters the spin of the *qubit*. Common SI gates are $Pauli - X, Pauli - Y, Pauli - Z$ and $Hadamard$ gate.
- *Multiple Input (MI) Operations:* These gates have two kinds of inputs i.e. control *qubit* and data *qubit*. The value of the control *qubit* determines the operation on the data *qubit*. The most common example of the MI operation is $CNOT$ gate.

In this paper, we have presented the hardware abstraction of the SI operations. The implemented operations are $Pauli - X, Pauli - Y, Pauli - Z$ and $Hadamard$ gate. Mathematically each SI gates depicts a matrix multiplication. However, in terms of architecture, the behavioral model of matrix multiplication is implemented to reduce the implementation cost. The architecture of each gate is elaborated in figure 7 and the description is as follows:

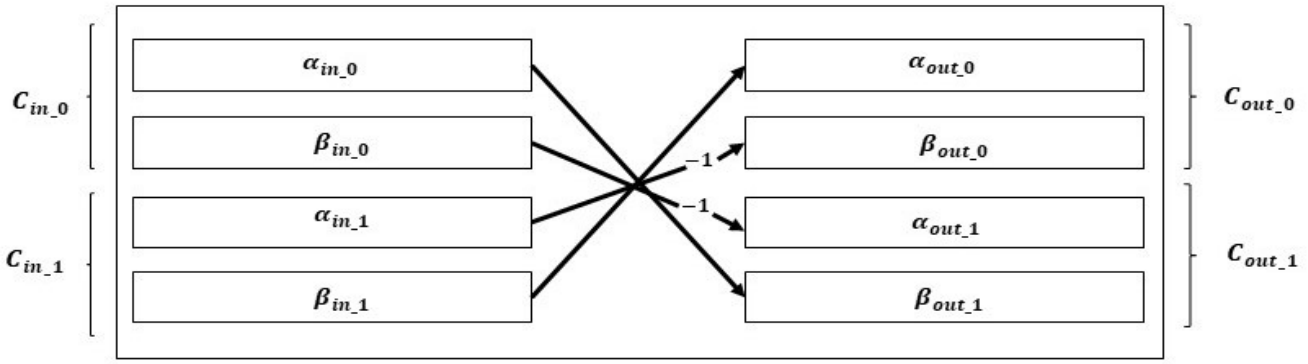
3.3.1 $Pauli - X$ Gate

The $Pauli-X$ gate is analogous to the *not* gate. Graphically, it gives the *qubit* vector 180^0 rotation along $x - axis$. Equation (15) shows the mathematical description of the gate.

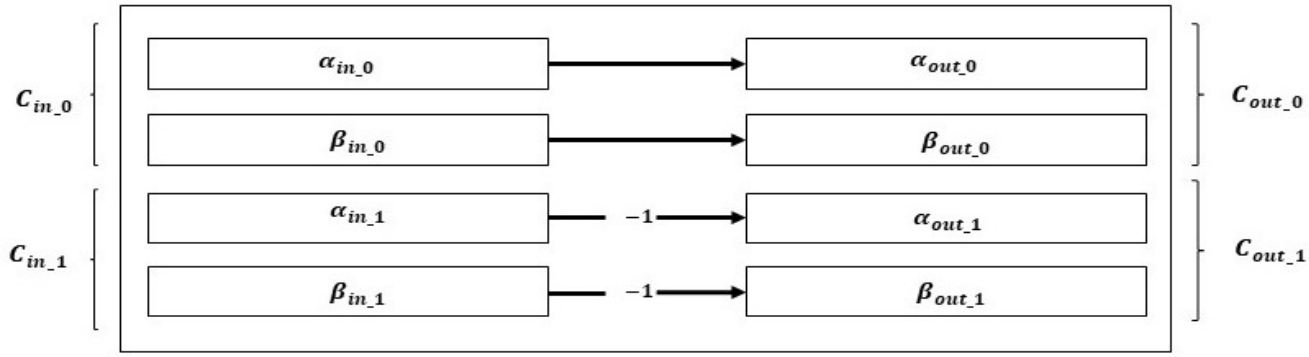
$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} C_{in\_0} \\ C_{in\_1} \end{bmatrix} = \begin{bmatrix} C_{out\_0} \\ C_{out\_1} \end{bmatrix} \tag{15}$$



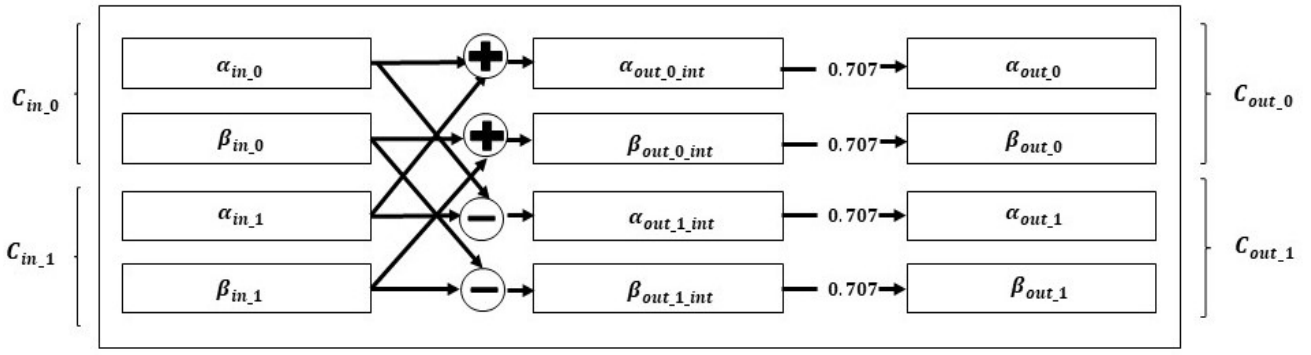
(a) $Pauli - X$ gate equation and architecture



(b) $Pauli - Y$ gate equation and architecture



(c) $Pauli - Z$ gate equation and architecture



(d) Hadamard gate equation and architecture

Fig. 7: Architecture of Single Input Quantum Gates

3.3.2 Pauli − Y Gate

This transformation rotates the bloch sphere and *qubit* vector plotted on the sphere 180^0 along $y-axis$. Equation (16) presents the mathematical model of the proposed gate.

$$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \begin{bmatrix} C_{in\_0} \\ C_{in\_1} \end{bmatrix} = \begin{bmatrix} C_{out\_0} \\ C_{out\_1} \end{bmatrix} \tag{16}$$

3.3.3 Pauli − Z Gate

This transformation gives an anticlockwise flip to the *qubit* along $z-axis$. This results in variation of probabilities to materialize as $|0\rangle$ and $|1\rangle$ *qubit* after measurement. The mathematical description of $Pauli-Z$ gate is given in equation (17).

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} C_{in\_0} \\ C_{in\_1} \end{bmatrix} = \begin{bmatrix} C_{out\_0} \\ C_{out\_1} \end{bmatrix} \tag{17}$$

3.3.4 Hadamard Gate

The transformation gives a sequence of two rotations to a *qubit* on the bloch sphere. The $Hadamard$ gate can be expressed as 90^0 rotation around $y-axis$ followed by 180^0 rotation around $x-axis$. Equation (18) represents its mathematical expression and architecture.

$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} C_{in\_0} \\ C_{in\_1} \end{bmatrix} = \begin{bmatrix} C_{out\_0} \\ C_{out\_1} \end{bmatrix} \tag{18}$$

3.4 The Counter

The counter is connected with the ALU and QFSM to synchronize the *qubit* signal processing operations based on clock cycles. The number of clock cycles required by the quantum operation depicts its execution time.

3.5 Quantum Finite State Machine (QFSM)

The finite state machine is responsible for coordination among all the functional blocks of the quantum processor. Following are the features of the quantum emulator FSM.

1. Take *qubit* as an input.
2. Coordinate with the ALU and counter for SI transformation.
3. Stores the output in *qubit* memory.
4. Transfer the SI output to the measurement block so that the *qubit* can be materialized as classical 0 bit or 1 bit.

4 Performance Analysis

This section defines the accuracy of the emulation model and the computational cost of the abstraction. A trade off exists among the emulation accuracy and the implementation cost mainly because of fixed point nature of the proposed FPGA based architecture. Figure 8 presents the accuracy model of the proposed design. There are four types of errors introduced during the circuit execution i.e.

1. *Bloch sphere quantization:* Initially, when the *qubit* is stored in a register file with 6-bit mantissa, the superposition states of basis are quantized introducing an error in *qubit* representation.
2. *Fixed point quantum gate representation:* The fixed point $(Q(2,6))$ gate representation introduces error because SISO gate matrices such as Hadamard gate and phase shift comprises of the irrational numbers.
3. *Fixed point qubit representation:* The fixed-point representation of *qubit* flip flops also results in the truncation of gate outputs thus introducing calculation errors.
4. *Discrete pseudo-random number generator:* The PbS function generates 8-bit integer values. Therefore, the measurement accuracy of a *qubit* is effected by the discrete nature of pseudo-random number generator.

The gate level hardware abstraction is implemented on *spartan* 3 using Verilog. The resource requirement of each block is illustrated in table 9 as Look-up-tables (LUT) and slice count. The direct application of the proposed design is an ultra-lightweight emulation of quantum key distribution algorithms. The basic quantum algorithms used for the symmetric key distribution are BB84 and BB92 [4]. The quantum circuits for these algorithms exercise the concept of parallelism and probabilistic measurement for the encryption of a private key. The micro blocks in the proposed architecture are used to design non-programmable architectures of the BB84 algorithms in the subsequent section. The total computational cost of the quantum protocol designed by using the proposed micro blocks is the cumulative effect of operations performed on the input.

5 Emulation of Quantum Algorithms

The emulation of SISO quantum gates defined in section 3 can be used to develop abstractions of the quantum algorithms i.e. Shor's algorithm, Grover's search algorithm and quantum key distribution algorithms. By virtue of the NIST post-quantum cryptography call, the probabilistic measurement feature of the *qubit* is

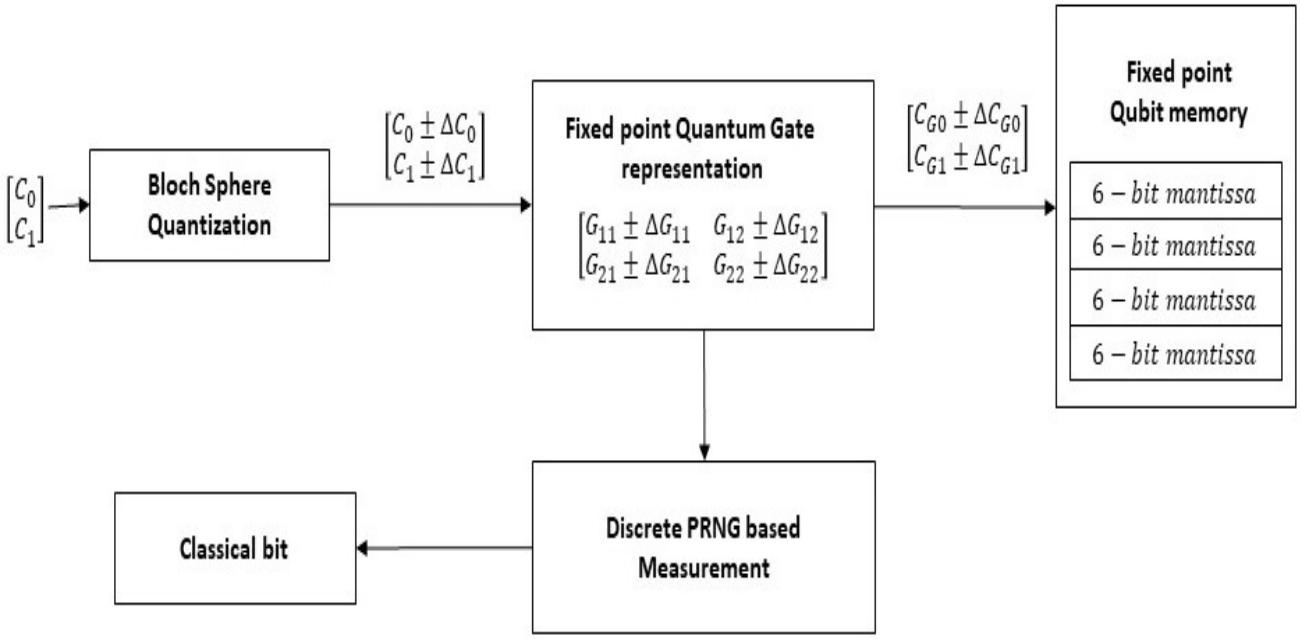


Fig. 8: Accuracy model of the quantum emulation

Table 9: Resource utilization of the quantum abstraction functional blocks

Functional Block	**Slice Count**	**Look-Up-Table**
qubit Memory	8	16
Measurement Block	189	301
$Pauli - X$	208	313
$Pauli - Y$	209	322
$Pauli - Z$	208	313
$Hadamard\,Gate$	256	419

widely being explored for public and private key encryption techniques [39]. The post quantum cryptography aims to develop a cryptographic system that is secure against both quantum and classical computers. Since large-scale quantum computers are not widely accessible, an emulation of quantum systems can facilitate the design, development and testing of the algorithms for the post-quantum era. In this paper, we have implemented the architecture of the quantum key distribution algorithm i.e. BB84 [40]. The protocol is deployed on four different systems i.e 8-bit,16-bit,32-bit and 64-bit. With the increase in the number of bits used to define a *qubit*, the superposition states increase and hence the accuracy of the emulation output enhances. This section explains the working of the BB84 protocol, the abstraction architecture and performance analysis of the algorithm.

5.1 The Bennett & Brassard 84 Algorithm

The Bennett and Brassard (BB84) algorithm is considered as the foundation stone in the field of quantum cryptography. The protocol uses non-cloning and probabilistic measurement feature to securely communicate the private key. The BB84 protocol implementation requires two unbiased orthonormal basis i.e. $[+, \times]$. The polarization states of each basis are given in table 10. The protocol executes in three steps defined as follows:

Table 10: Quantum basis definition for BB84 algorithm

States	*qubits* **in** $+$ **basis**	*qubits* **in** $\times$ **basis**
$\|0\rangle$	$\| \rightarrow \rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$	$\| \nearrow \rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$
$\|1\rangle$	$\| \uparrow \rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$	$\| \nwarrow \rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}$

1. *Key transmission phase:* The transmitter i.e. Alice generates an 8-bit binary key through a pseudo-random number generator. Alice then generates a random sequence of basis among $[+, \times]$, transforms each bit of the key to a *qubit* with the corresponding basis. Finally, an $8 - qubit$ sequence is transmitted to the receiver. Table 11 elaborates the key transmission process.

Table 11: Example of BB84 protocol transmitter

Bit Number	7	6	5	4	3	2	1	0
Key	0	1	0	0	1	0	1	0
Random Base	+	+	x	x	x	+	x	x
Key *Qubit*	$\|\rightarrow\rangle$	$\|\uparrow\rangle$	$\|\nearrow\rangle$	$\|\nearrow\rangle$	$\|\nwarrow\rangle$	$\|\rightarrow\rangle$	$\|\nwarrow\rangle$	$\|\nearrow\rangle$

2. *Key retrieval phase:*The receiver i.e. Bob obtains the *qubit* sequence and measures each one by randomly choosing among $[+, \times]$ measurement blocks. Table 12 explains the key retrieval process at the receiver's side.
3. *Reconciliation phase:* As a final step, the communicating parties use a public channel to compare their basis choices.
   Since Bob randomly chooses the basis for measurement, the probability of selection of the same basis for the polarization and the measurement of the classical bit is 0.5. The remaining 50% of the instances when Bob uses different basis, the measurement of *qubit* will agree with Alice's about 50% of the time by virtue of unbiased orthonormal nature of the selected basis. Therefore, the overall error probability in the received private key is 0.25. In this paper the effect of intruder on the error probability is not considered. The adversary can execute man-in-the -middle attacks to increase the bit error rate of the private key.

In the subsequent subsection, the architecture of the BB84 protocol is discussed. This emulation is designed on the principle of cascading quantum sub-blocks to transmit the private key over the quantum channel.

5.2 BB84 Emulation Architecture

In this sub section, the architecture of the transmitter and the receiver implementing BB84 protocol is presented. The basic design principle of both architectures is to incorporate SISO sub-blocks defined in section 3. The transmitter's basic purpose is to input a private key and encode it into a *qubit* sequence. The orthonormal basis used by the abstraction are $[+, \times]$. The step by step description of the transmitter architecture is as follows:

1. The transmitter is activated by an 8-bit binary key.
2. An 8-bit pseudo-random sequence is generated by the PbS function defined in section 3.2.
3. A $2 \times 4\, mux$ sequentially transforms each bit of the input key to a *qubit* with the basis defined by the corresponding bits of random sequence i.e. $(r[n] = 0 \mathrel{\widehat{=}} +; r[n] = 1 \mathrel{\widehat{=}} \times)$. The working principle of the $2 \times 4\, mux$ is defined in table 13.
4. The sequential processing of the input sequence is governed by a $3 \times 8\, mux$. The selection line of this mux is a 3-bit counter that increments after every clock cycle. Starting from the *lsb* side every positive edge of the clock, inputs subsequent key bit along with the corresponding random sequence bit to the selection line of $2 \times 4\, mux$.
5. The quantum transformation of the key is saved in an 8-bit *qubit* flipflops and also transmitted to the receiver's end.

The architecture of the transmitter's side is depicted in figure 9.

At the receiver's side, the *qubit* key sequence is transformed to a classical 8-bit sequence through the probabilistic measurement. The *qubits* are randomly subjected to either of the basis $[+, \times]$ for the measurement purpose. The step by step description of the receiver's architecture is as follows:

1. A $8 \times 1\, mux$ is responsible for the sequential measurement of the *qubits* with the help of a 3-bit counter as a selection line. The counter increments after every successful measurement of a *qubit*.
2. This step corresponds to the transformation of a *qubit* so that for the basis among $[+, \times]$, a single measurement block can be used for the key retrieval. For the + basis measurement, no transformation is required as the block in section 3 is based on states $|\rightarrow\rangle, |\uparrow\rangle$.
   For measurement with $\times$ base, the measurement block should use $|\nearrow\rangle$ and $|\nwarrow\rangle$ as absolute states. A *Hadamard* gate is used to spin $(|\nearrow\rangle, |\nwarrow\rangle$ states to $(|\rightarrow\rangle, |\uparrow\rangle)$ respectively. This corresponds to translation of $\times$ basis *qubit* to a + basis *qubit* without altering its probability to materialize as a 0 or a 1.
3. A $2 \times 1\, mux$ is used to select the basis for measurement of a *qubit* based on the sequential bits of 8-bit random number generated by PbS function i.e. $randomnumber[n] = 0 \mathrel{\widehat{=}} +$ basis and $randomnumber[n] = 1 \mathrel{\widehat{=}} \times$ basis.
4. As a final step, the measurement block is used to convert a *qubit* to a conventional bit.

The architecture of the receiver's side is presented in figure 10. The architecture is designed for four different systems i.e. 8-bit,16-bit,32-bit and 64-bit. The perfor-

Table 12: Example of BB84 protocol receiver

Bit Number	7	6	5	4	3	2	1	0
Key *Qubit*	$\|\rightarrow\rangle$	$\|\uparrow\rangle$	$\|\nearrow\rangle$	$\|\nearrow\rangle$	$\|\nwarrow\rangle$	$\|\rightarrow\rangle$	$\|\nwarrow\rangle$	$\|\nearrow\rangle$
Random Base	+	x	x	+	x	x	x	+
Key	0	1	0	1	1	0	1	1

Table 13: *qubit* transformation of the classical bits

Random number$r[n]$	**Key**$key[n]$	$2x4$ **mux output**
0	0	$\|\rightarrow\rangle$
0	1	$\|\uparrow\rangle$
1	0	$\|\nearrow\rangle$
1	1	$\|\nwarrow\rangle$

mance analysis of the emulations in comparison to the ideal system is given in the subsequent sub section.

5.3 Performance Analysis of BB84 Emulation

The architecture of the BB84 protocol described in subsection 5.2 is tested on four quantum abstractions. Each abstraction is defined by the size of the *qubit* flipflops. The n-bit system corresponds to four n-bit registers for a single *qubit*. In this section, four emulations i.e. 8-bit, 16-bit, 32-bit and 64-bit are deployed and compared against the ideal quantum system. For the analysis purposes every BB84 abstraction encodes and transmits an 8-bit randomly generated key. Fifty randomly selected events of the key retrieval are recorded for every abstraction variant. Figure 11 represents the frequency of number of bits received for 50 instances by all systems.The specifications of the proposed system are given in table 14.

The average number of bits retrieved for every system gives percentage of success. For an ideal quantum system, 75% of the retrieved key should confirm with the transmitter's input thus justifying the bit error rate

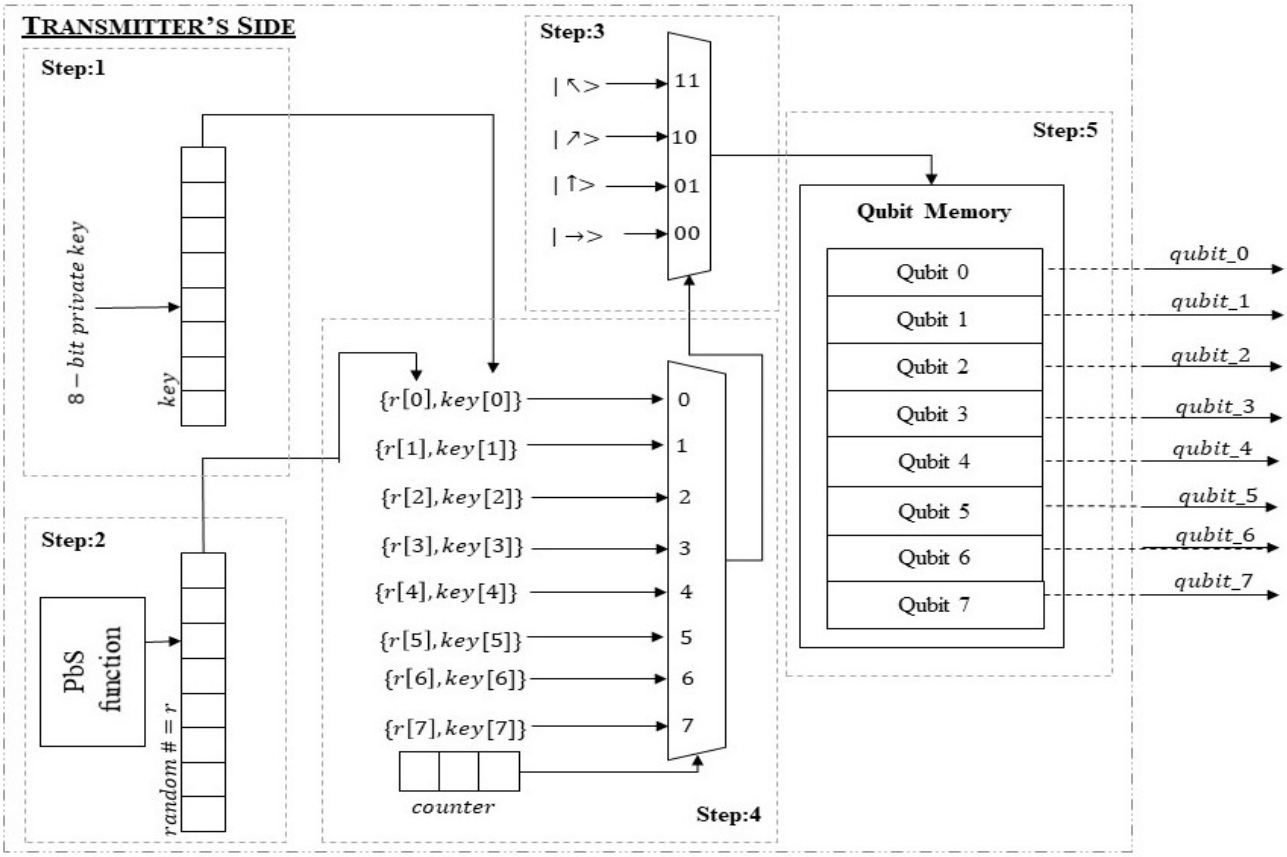


Fig. 9: Transmitter emulation for the BB84 protocol

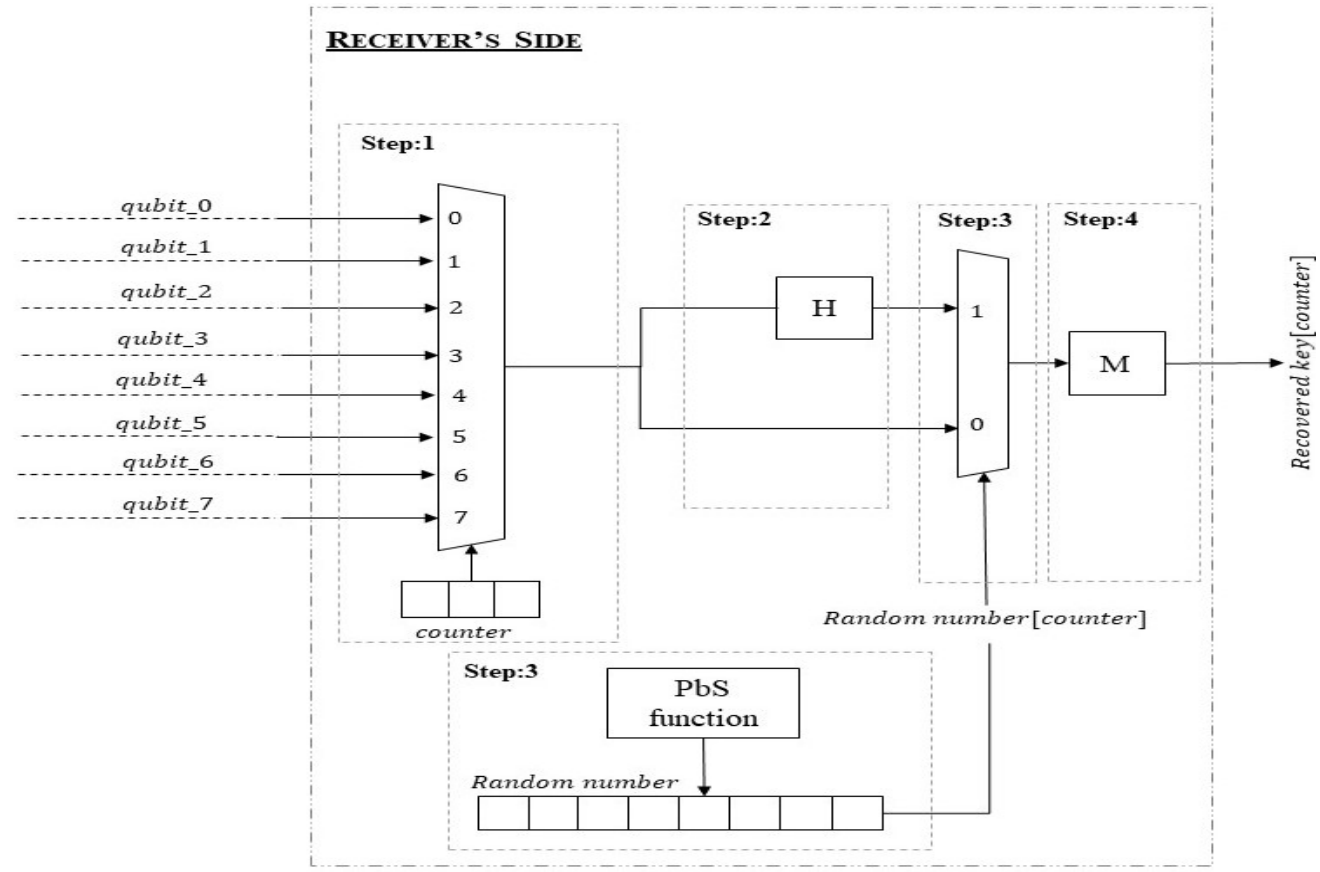


Fig. 10: Receiver's emulation for BB84 protocol

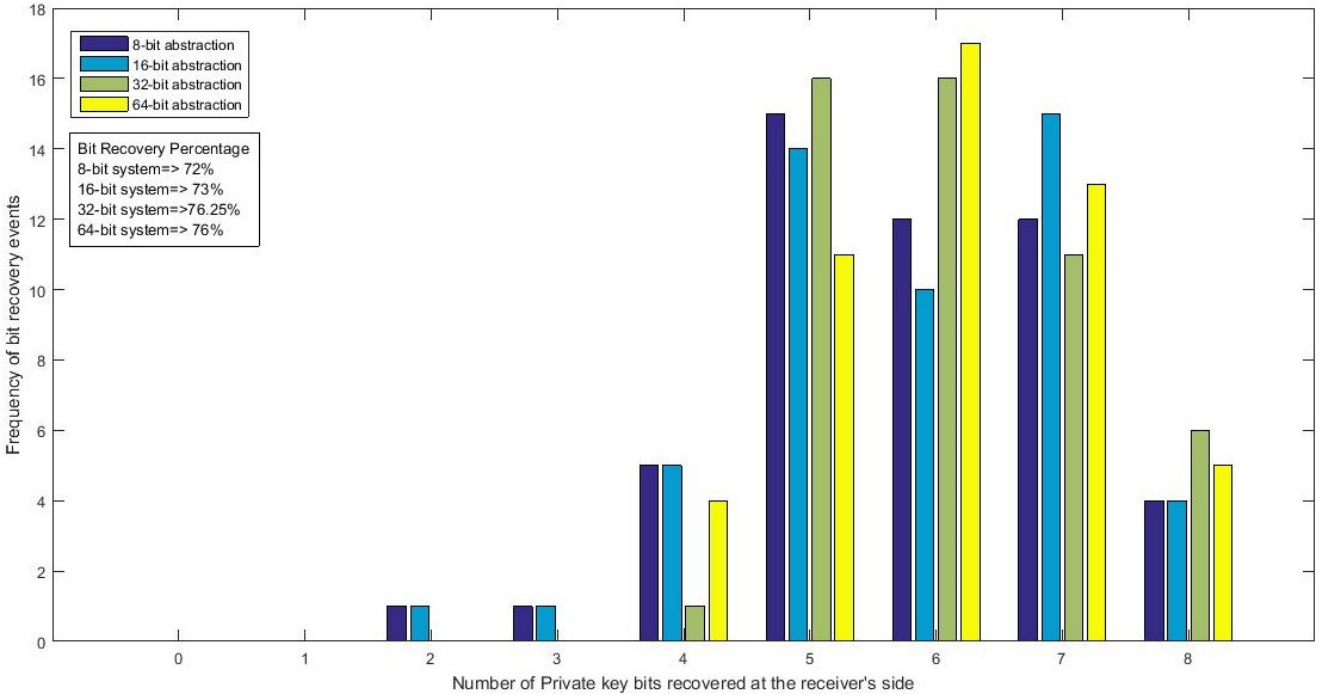


Fig. 11: Experimental data of BB84 emulation

Table 14: System Specifications of BB84 emulation

Parameters	Values
Private Key	0 1 0 0 1 0 1 0
Quantum Key	$\|\rightarrow\rangle \|\uparrow\rangle \|\nearrow\rangle \|\nearrow\rangle \|\nwarrow\rangle \|\rightarrow\rangle\|\nwarrow\rangle \|\nearrow\rangle$
Key Size	8-bit
Number of Trials	50
Abstraction system	8-bit,16-bit,32-bit, 64-bit
Ideal success rate of Key distribution	75%

Table 15: Performance Analysis of BB84 Emulation

8-bit abstraction error	3%	**8-bit abstraction implementation cost**	302LUT
16-bit abstraction error	2%	**16-bit abstraction implementation cost**	1571LUT
32-bit abstraction error	1.5%	**32-bit abstraction implementation cost**	4842LUT
64-bit abstraction error	1%	**64-bit abstraction implementation cost**	14993LUT

of 0.25. The overall performance in terms of emulation error is obtained by comparing the success rate of the abstraction with the ideal system. Table 15 shows linear decrease in the error with the increase of mantissa size for the *qubit* representation. This shows that by increasing the number of bits for the *qubit* representation the quantization and truncation errors are reduced thus making the emulation more reliable.

6 Conclusion

In this paper, an FPGA based emulation of the low-cost quantum circuit is proposed. The architecture of single input gates, memory and measurement block have been implemented. Our proposed solution is a standalone system capable to exhibit parallelism and probabilistic measurement. This paper also presents the gate level abstraction of BB84 protocol on four different system i.e. 8-bit,16-bit,32-bit and 64-bit. The performance analysis shows that with the increase in size of quantum emulation, the accuracy of the model improves linearly. The proposed system can be implemented for the ASIC solution of quantum key distribution for the resource constraint scenarios.

A Appendix

The NIST randomness test results for PbS based random number generator are given in table 16.

Conflict of interest

Not Applicable

References

1. C.P. Williams, *Explorations in quantum computing* (Springer Science & Business Media, 2010)
2. J.M. Shalf, R. Leland, Computer **48**(12), 14 (2015)
3. R. Colwell, in *2013 IEEE Hot Chips 25 Symposium (HCS)* (IEEE Computer Society, 2013), pp. 1–16
4. N.S. Yanofsky, M.A. Mannucci, *Quantum computing for computer scientists* (Cambridge University Press, 2008)
5. Y. Kanamori, S.M. Yoo, W. Pan, F.T. Sheldon, International Journal of Computers and Applications **28**(3), 227 (2006)
6. V. Kendon, arXiv preprint arXiv:2004.00704 (2020)
7. W. Pan, F. Sheldon, International Journal of Computers and Applications **28**(3) (2006)
8. J.I. Cirac, P. Zoller, Physical review letters **74**(20), 4091 (1995)
9. Q.A. Turchette, C.J. Hood, W. Lange, H. Mabuchi, H.J. Kimble, Physical Review Letters **75**(25), 4710 (1995)
10. E. Knill, R. Laflamme, G.J. Milburn, nature **409**(6816), 46 (2001)
11. D. Loss, D.P. DiVincenzo, Physical Review A **57**(1), 120 (1998)
12. M. Oskin, F.T. Chong, I.L. Chuang, Computer **35**(1), 79 (2002)
13. S. Cavallar, B. Dodson, A.K. Lenstra, W. Lioen, P.L. Montgomery, B. Murphy, H. Te Riele, K. Aardal, J. Gilchrist, G. Guillerm, et al., in *International Conference on the Theory and Applications of Cryptographic Techniques* (Springer, 2000), pp. 1–18
14. L.K. Grover, in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing* (1996), pp. 212–219
15. L. Litvintseva, S. Ul'yanov, Journal of Computer and Systems Sciences International **48**(6), 946 (2009)
16. C.A. Fuchs, J. Van De Graaf, IEEE Transactions on Information Theory **45**(4), 1216 (1999)
17. L. Goldenberg, L. Vaidman, Physical Review Letters **75**(7), 1239 (1995)
18. P.Q. Le, F. Dong, K. Hirota, Quantum Information Processing **10**(1), 63 (2011)
19. A.M. Iliyasu, P.Q. Le, F. Dong, K. Hirota, Information Sciences **186**(1), 126 (2012)
20. F. Yan, A.M. Iliyasu, P.Q. Le, International Journal of Quantum Information **15**(03), 1730001 (2017)
21. F. Yan, K. Chen, A.M. Iliyasu, K. Hirota, IEEE Access **8**, 23054 (2020)

Table 16: NIST Randomness Test for PbS Function

S no	**Test**	$p-value^*$
1	Frequency Test ($n = 112$)	1.00
$2a$	Frequency Block test ($M = 20, n = 2208$)	0.498220
$2b$	Frequency Block Test ($M = 32, n = 3504$)	0.152638
3	Runs ($n = 39008$)	0.265440
4	Longest Runs of Ones ($n = 128$)	0.491591
5	Rank Test ($n = 39008$)	0.376508
6	Spectral ($n = 1008$)	0.728709
7	Non-Overlapping Template Matching ($m = 2, n = 112$)	0.742204/0.802245
8	Overlapping Template Matching ($m = 16, n = 1000000$)	0.689811 *
9	Maurer's Universal Test ($n = 390000$)	0.055461
10	Linearity ($M = 500, n = 100016$)	0.050150
11	Serial ($m = 8, n = 1048576$)	0.486466/0.052607
12	Approximate Entropy ($m = 8, n = 16384$)	0.360903
13	Cumulative Sum ($n = 112$)	$F : 0.972584/R : 0.972584$
14	Random Excursions ($n = 1000016$)	0.466659; 0.793406; 0.880520; 0.255360; 0.427173; 0.437131; 0.785004, 0.828347
15	Random Excursions Variant ($n = 1000016$)	0.795474; 0.626100; 0.842141; 0.668670; 0.616389; 0.032382; 0.061898; 0.622928; 0.270399; 0.019393; 0.302252; 0.994041; 0.609178; 0.849891; 0.923787; 0.929878; 0.989680; 0.801730

Acceptable $p-value$ is less than one

22. A. Manju, M.J. Nigam, Artificial Intelligence Review **42**(1), 79 (2014)
23. J. Sherwood, T. Stephenson, S. Bernstein, Physical Review **96**(6), 1546 (1954)
24. N. Raychev, International Journal of Scientific and Engineering Research **6**(6), 1369 (2015)
25. B.Z. Sun, S.M. Fei, N. Jing, X. Li-Jost, Quantum Information Processing **19**(3), 103 (2020)
26. N. Schuch, J. Siewert, Physical Review A **67**(3), 032301 (2003)
27. M. Fujishima, K. Saito, M. Onouchi, H. Hoh, in *Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS'03.*, vol. 4 (IEEE, 2003), vol. 4, pp. IV–IV
28. M. Fujishima, in *Proceedings. 2003 IEEE International Conference on Field-Programmable Technology (FPT)(IEEE Cat. No. 03EX798)* (IEEE, 2003), pp. 21–26
29. M.P. Frank, L. Oniciuc, U.H. Meyer-Baese, I. Chiorescu, in *Quantum Information and Computation VII*, vol. 7342 (International Society for Optics and Photonics, 2009), vol. 7342, p. 734203
30. Y. Goto, M. Fujishima, Japanese journal of applied physics **46**(4S), 2278 (2007)
31. Y.H. Lee, M. Khalil-Hani, M.N. Marsono, International Journal of Reconfigurable Computing **2016** (2016)
32. M. Aminian, M. Saeedi, M.S. Zamani, M. Sedighi, in *2008 IEEE Computer Society Annual Symposium on VLSI* (IEEE, 2008), pp. 399–404
33. G. Negovetic, M. Perkowski, M. Lukac, A. Buller, (2002)
34. A.U. Khalid, Z. Zilic, K. Radecka, in *IEEE International Conference on Computer Design: VLSI in Computers and Processors, 2004. ICCD 2004. Proceedings.* (IEEE, 2004), pp. 310–315
35. J. Pilch, J. Długopolski, Journal of Computational Electronics **18**(1), 329 (2019)
36. R.G. Krishna, G. Sarath, in *2017 International Conference on Technological Advancements in Power and Energy (TAP Energy)* (IEEE, 2017), pp. 1–6
37. S. Chakraborty, A. Garg, M. Suri, IEEE Transactions on Electron Devices **67**(3), 888 (2020)
38. A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, A statistical test suite for random and pseudorandom number generators for cryptographic applications. Tech. rep., Booz-allen and hamilton inc mclean va (2001)
39. G. Alagic, G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, Y.K. Liu, C. Miller, D. Moody, R. Peralta, et al., *Status report on the first round of the NIST post-quantum cryptography standardization process* (US Department of Commerce, National Institute of Standards and Technology, 2019)
40. R. Renner, International Journal of Quantum Information **6**(01), 1 (2008)