HHL Analysis and Simulation Verification Based on Origin Quantum Platform

Xiaonan Liu¹, Lina Jing²*, Lin Han², and Jie Gao¹

¹State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, 450001, China
²Henan Supercomputing Center, Zhengzhou University, Zhengzhou, 450001, China
*Corresponding author’s e-mail: liyinxinllx@gs.zzu.edu.cn

Abstract Solving large-scale linear equations is of great significance in many engineering fields, such as weather forecasting and bioengineering. The classical computer solves the linear equations, no matter adopting the elimination method or Kramer’s rule, the time required for solving is in a polynomial relationship with the scale of the equation system. With the advent of the era of big data, the integration of transistors is getting higher and higher. When the size of transistors is close to the order of electron diameter, quantum tunneling will occur, and Moore’s Law will not be followed. Therefore, the traditional computing model will not be able to meet the demand. In this paper, through an in-depth study of the classic HHL algorithm, a small-scale quantum circuit model is proposed to solve a 2×2 linear equations, and the circuit diagram and programming are used to simulate and verify on the Origin Quantum Platform. The fidelity under different parameter values reaches more than 90%. For the case where the matrix to be solved is a sparse matrix, the quantum algorithm has an exponential speed improvement over the best known classical algorithm.

1. Introduction
The quantum era has arrived. Linear equations, as one of the most important content in linear algebra, play an important role in many fields, so it is very important to solve large-scale linear equations. However, regardless of whether Gaussian elimination or Kramer’s rule is used to solve linear equations on a traditional classical computer, the solution time is in a polynomial relationship with the dimension of the equation. With the advent of the era of big data, the growth rate of data has reached an unprecedented level[1], and traditional computers have certain difficulties in processing large-scale data. Therefore, the current situation puts forward higher requirements for us.

In response to this situation, quantum computing bears the brunt, it is currently one of the hottest and best solutions. As we all know, quantum computing is naturally capable of parallel computing due to its superposition, entanglement and reversibility. Compared with the parallel computing capability of classical computers, it has incomparable advantages in multi-machine parallel, multi-thread parallel and instruction level parallel. An N-bit classic register can only store a certain number of bits from 0 to 2N-1, while in a quantum computer, the number of N-bit quantum registers in the super imposed state is all the numbers from 0 to 2N-1, which exist at the same time with certain probability[2]. Therefore, one N-bit quantum register can simultaneously store 2N N-bit binaries. When individuals operate on N qubits, in principle, it means that operate on 2N states simultaneously[2]. The linear increase in the number of bits in the quantum register causes the storage space to increase exponentially. This is the
basic feature of quantum computing memory cells and the premise that quantum computing speed can greatly exceed the classic computing speed[3].

According to the erasure theorem, in an irreversible process, the heat is closely related to the scale of the irreversible operation: The higher the integration, the more heat is generated per unit area[4].

The quantum computer based on the basic principles of quantum mechanics, and the information processing process is unitary transformation, so the reversibility of unitary transformation makes quantum information processing process does not bring heat. Therefore, quantum computing can theoretically solve the problem of high energy consumption, another key technology of modern information processing.

The structure of this paper is as follows: The first part introduces the background and significance of quantum computing. In the second part, the HHL quantum algorithm for solving linear equations is discussed in detail, and its solution idea, main algorithm framework and complexity advantage over classical algorithm are analyzed. In the third part, combining the practical characteristics of the Origin Quantum Platform, the specific case in this HHL algorithm is simulated by means of circuit diagram and programming. The last part is the summary of this paper and the prospect of the follow-up work.

2. HHL quantum algorithm principle
Aram W. Harrow, Avinatan Hassidim and Seth Lloyd explicitly proposed a quantum algorithm for solving linear equations[5], which solves such a problem: In \(Ax = b\), if an Hermite \(N \times N\) matrix \(A\) and a unit vector \(b\) are given, one solve for \(x\) satisfying the above equation. Equivalently, one can convert the problem to solve \(x = A^{-1}b\). Since matrix \(A\) is an invertible matrix, this equation must have a unique set of solutions.

First, the \(N \times N\) matrix \(A\) has to be an Hermitian matrix (the conjugate transposed matrix of \(A\) is equal to itself). Secondly, the input vector \(b\) is a normalized quantum state. When the matrix \(A\) is not an Hermitian matrix, individuals convert the matrix \(A\) into an Hermitian matrix as follows:

\[
\hat{A} = \begin{bmatrix} 0 & A^T \\ A^* & 0 \end{bmatrix}
\]

At this time, the constructed matrix \(\hat{A}\) will be changed from a general matrix to an Hermitian matrix, and subsequent paper will only consider the case where matrix \(A\) is an Hermitian matrix. Correspondingly, when \(\hat{A}\langle x \rangle = \langle b \rangle\) is solved, \(\langle x \rangle = (0 \ x)^T\) will be found, and \(x\) at this time is the solution of the linear equations. Since the quantum state used to store the linear equation vector information must be normalized, the input information of the quantum linear equations algorithm is not a strict vector \(b\), but a normalized quantum state \(|\psi\rangle\). When the matrix \(A\) is not an Hermitian matrix, individuals convert the matrix \(A\) into an Hermitian matrix by construction as follows:

\[
\hat{A} = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}
\]

It is worth noted that the HHL quantum algorithm finally solved a quantum state containing \(x\) information. That is, the HHL quantum algorithm could not directly solve the size of \(x\), but could only obtain a global characteristic about \(x\). Therefore, the result \(\langle x \rangle\) differs from the solution of the original problem by a normalized constant. However, for many real-world applications, individuals don’t need to get the exact value of \(x\) directly, just the overall properties. In addition, the normalized constant can also be obtained by means of amplitude estimation in the case that exact value is required. Therefore, HHL quantum algorithm is worth considered when it is necessary to solve large-scale linear equations without requiring the exact value of output directly.
In order to make the problem more clear, the process of HHL quantum algorithm is first analyzed in detail. For the Hermitian matrix $A$, one express it as the spectral decomposition of the $N \times N$ Hermitian matrix:

$$
A = \sum_j \lambda_j |\mu_j\rangle \langle \mu_j|
$$

(1)

Where $|\mu_j\rangle$ is the eigenvector of matrix $A$, and $\lambda_j$ is the eigenvalue corresponding to matrix $A$. From $A|\mu_j\rangle = \lambda_j |\mu_j\rangle$ theorem one can get such an equation:

$$
A^{-1} |\mu_j\rangle = \frac{1}{\lambda_j} |\mu_j\rangle
$$

(2)

For the input quantum state $|b\rangle$, if expand it on the eigenvector $|\mu_j\rangle$ of the matrix $A$, one can get:

$$
|b\rangle = \sum_j ^N \beta_j |\mu_j\rangle
$$

(3)

The equation of $|x\rangle$ can be obtained from the above equations:

$$
|x\rangle = A^{-1}|b\rangle = A^{-1} \sum_j ^N \beta_j |\mu_j\rangle = \sum_j ^N \frac{\beta_j}{\lambda_j} |\mu_j\rangle
$$

(4)

It can be found that the required output vector $|x\rangle$ is actually the input vector $|b\rangle$ with a corresponding reciprocal of the eigenvalue of the matrix $A$ as a coefficient before each of its eigenvectors. Therefore, the purpose of solving linear equations can be transformed into how to obtain the information of these eigenvalues and add them to the corresponding eigenvectors[6]. The specific implementation of the algorithm can be divided into the following three subroutines: Phase Estimation(PE), Controlled Rotation(CR), and Inverse Phase Estimation(PE$^{-1}$)[7]. The quantum circuit of this algorithm is shown in Fig.1, with the first, second and third registers from top to bottom.

![Quantum circuit of HHL algorithm](image)

The quantum algorithm includes three registers: an auxiliary qubit register with the initial state of $|0\rangle$, a register containing n qubits with the initial state of $|0\rangle$, and a register with the initial state of $|b\rangle$.

The first subroutine, Phase Estimation, is analyzed. Step 1 is the preparation of the initial state. The first register is the auxiliary register and its initial state is $|0\rangle$. The second register is a working register, and its initial state is $|0\rangle$ as well, and it contains n qubits. Here, n is the number of unknowns in linear equations. The third register is initialized to $|b\rangle$. Then, the coefficient matrix $A$ of linear equations is prepared into unitary matrix by $e^{-iAt_0}$. At this point, the joint initial state of the entire system is
(excluding the auxiliary register) \( |0\rangle^\otimes n |b\rangle \). Step 2 is the Phase Estimation operation. The purpose of this operation is to extract the eigenvalues of the coefficient matrix \( A \). One expresses the eigenvector of the coefficient matrix \( A \) as \( |\mu\rangle \), then \( \lambda \) represents the eigenvalue corresponding to the matrix \( A \). Firstly, the eigenvalues of matrix \( A \) are extracted into the probability amplitude of the quantum state of the second register by H gate and controlled unitary matrix. And then the phases of the probability amplitude are extracted by the inverse Quantum Fourier Transform (QFT\(^{-1}\)) and placed into the ground state of the quantum state of the second register. At this time, the state of the second register is \( |\lambda_i\rangle \), where \( |\lambda_i\rangle \) is binary storage. The expansion of \( |\beta\rangle \) in the third register is \( \sum \beta_j |\mu_j\rangle \) based on the characteristic vector \( |\mu_j\rangle \), and the joint initial state of the whole system is \( \sum_j \beta_j |\lambda_i\rangle |\mu_j\rangle \). This step is the acceleration of the whole algorithm, because when calculating the eigenvalues of the coefficient matrix \( A \), it is processed for all the eigenvalues simultaneously, that is, parallel computing.

Although the information of the eigenvalues into the second register in the Phase Estimation has been extracted, the eigenvalues here are stored in the quantum state. According to Formula (4), the eigenvalue \( \lambda_i \) must be stored in the form of coefficient and in the form of reciprocal. The results of the second register do not currently exist in the form individuals expect. Therefore, these can be converted through Controlled Rotation.

Controlled Rotation is mainly implemented in the first register and the second register, which is divided into three steps. First, one change the state of the second register from \( |\lambda_i\rangle \) to \( \frac{1}{\lambda_i} \) temporarily. Then, through the auxiliary qubit introduced in the first register, under the operation of the controlled gate, the state of the first register is mapped from \( |0\rangle \) to the superposition state of \( |0\rangle \) and \( |1\rangle \) to get \( \sqrt{1-c^2} |0\rangle + \frac{c}{\lambda_i} |1\rangle \) (where \( c \) is constant). Finally, the state of the second register is restored to \( |\lambda_i\rangle \). At this time, the state of the whole system is \( \sum \left( \sqrt{1-c^2} |0\rangle + \frac{c}{\lambda_i} |1\rangle \right) \beta_j |\lambda_i\rangle |\mu_j\rangle \) (including the first register). In short, the function of Controlled Rotation is to extract the reciprocal of the ground state value proportionally to the corresponding probability amplitude of the ground state by an auxiliary qubit.

When the Controlled Rotation is completed, the state of the entire system is close to the final output. The output of the third register is useful. At this time, the second register and the third register are in an entangled state, which will affect the result. Therefore, the operation purpose of the Reverse Phase Estimation is to de-entangle the second register and the third register, and restore the result of the second register to the initial state. Only the entanglement result of the first register and the third register is left. The entire Inverse Phase Estimation is actually the inverse process of the previous Phase Estimation. After execution, the second register untangles and returns to the initial state. Meanwhile, the result of the entire system is \( \sum \left( \sqrt{1-c^2} |0\rangle + \frac{c}{\lambda_i} |1\rangle \right) \beta_j |\mu_j\rangle \). Because the first register is in the superimposed state, and what individuals need is when the state of the first register is observed as \( |1\rangle \), then the result of the whole system becomes \( \frac{c}{\lambda_i} \beta_j |\mu_j\rangle |1\rangle \). This result is proportional to the solution of the linear equations. Therefore, individuals only need to perform the measurement operation on the first register, and discard all the measurement results of \( |0\rangle \). The remaining results at this time is the solution.

The complexity analysis of this algorithm mainly comes from Phase Estimation and Controlled Rotation. We define \( s \) as the sparsity of the matrix, that is, there are no more than \( s \) non-zero elements in each row. And is the conditional number, that is, the absolute value of the ratio between the
maximum eigenvalue of matrix A and the minimum eigenvalue. In the third register, the resource required when the coefficient matrix A of the linear equations is prepared into a unitary matrix by $e^{-iAt_0}$ is $O(\log(N)s^2t)$. When performing the Phase Estimation, let $t = O(\kappa/\varepsilon)$, where $\varepsilon$ is the acceptable error rate of the output vector. Finally, in order to measure the result of $|1\rangle$ in the first register, one also need $O(\kappa)$ iterations. Therefore, for a sparse N×N matrix A, the complexity achieved by the entire HHL quantum algorithm is $O(\log(N)s^2\kappa^2/\varepsilon)$[6][7]. At present, for solving the problem of linear equations, the fastest classic algorithm is based on the conjugate gradient descent method, and its complexity is $O(Ns\kappa/\varepsilon^2)$. It can be seen that in addition to the lack of accuracy, the HHL quantum algorithm has a huge advantage over the classical algorithm in other respects. Under certain conditions, it has an exponential time acceleration.

3. Simulation implementation

In order to verify the feasibility of HHL quantum algorithm, 2×2 linear equations will be simulated and verified. In this paper, the Origin Quantum Platform is selected as the experimental platform.

Founded on September 11, 2017, the Origin Quantum is the first domestic start-up company whose main business is the development and application of quantum computer. The Origin Quantum Platform provides quantum-virtual machines for users to choose, in which 32-bit quantum-virtual machine is free to use and 64-bit application is required. Virtual machine adopts visual programming learning mode: legend + quantum language. Users can easily drag and place legend for quantum simulation operation, and can convert the designed operation into quantum language mode for in-depth learning. QPanda SDK is a C++ host language toolkit for writing quantum programs and applications, which enables users to easily connect and execute quantum programs. For quantum learners, it greatly reduces the difficulty of learning quantum computation.

3.1. Circuit implementation

In order to solve the 2×2 linear equations, four qubits are needed to achieve the system. One qubit as the input $|b\rangle$, one as an auxiliary qubit for Controlled Rotation and measurement, and two qubits for Phase Estimation. For 2×2 linear equations, its coefficient matrix A is selected as:

$$A = \frac{1}{2} \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix}$$

Through calculation, individuals can get the eigenvector of matrix A as follows:

$$|\mu_1\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \quad |\mu_2\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

In this demonstration experiment, for input $|b\rangle$ of the third register, the paper selects two sets of data as a demonstration:

$$|b_1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |b_2\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

The circuit for solving linear equations is shown in Fig.2. Assuming $|b_1\rangle^2 + |b_2\rangle^2 = 1$, the vector $b$ can be encoded into this state: $|b\rangle = b_1|0\rangle + b_2|1\rangle$. The eigenvalues of matrix A are $\lambda_1=1$ and $\lambda_2=2$, and the corresponding eigenvectors are $|\mu_1\rangle$ and $|\mu_2\rangle$. At this time, $\lambda_1$ and $\lambda_2$ can be accurately encoded by $|q, q2\rangle = |01\rangle$ and $|q, q2\rangle = |10\rangle$, respectively. Therefore, after the Phase Estimation, the state $|q, q2, q3\rangle$ of the 3-qubit system will be $\beta_1|01\rangle|\mu_1\rangle + \beta_2|10\rangle|\mu_2\rangle$. 

5
After the Phase Estimation, in order to temporarily change the state of the second register from \( |\lambda_i\rangle \) to \( |\frac{1}{\lambda_i}\rangle \), this section uses a special method that does not require any auxiliary qubits. An essential step is to apply the SWAP gate between \( q_1 \) and \( q_2 \), so the 3-qubit system is converted to \( \beta|0\rangle|\mu_i\rangle + \beta|01\rangle|\mu_2\rangle \) state. This can interpret \( |q_1q_2\rangle = |01\rangle \) as the state where the reciprocal of the coded eigenvalue \( 2\lambda_i^2 = 2 \), and interpret \( |q_1q_3\rangle = |01\rangle \) as the reciprocal of the coded eigenvalue \( 2\lambda_2 = 1 \) similarly. In other words, after the SWAP gate, the system state becomes

\[
|q_1q_2q_3\rangle = \sum_{j\neq i}^{2} \beta_j |2\lambda_j^{-1}\rangle|\mu_j\rangle.
\]

When performing a Controlled Rotation, in general, the mapping from \( \sum_{j} \beta_j |\lambda_j\rangle|\mu_j\rangle \) to \( \sum_{j} \beta_j |\lambda_j\rangle |\lambda_j^-\rangle|\mu_j\rangle \propto |x\rangle \) should in principle use a controlled \( R \) operation. The original target operation

\[
R = R(0) \left| \frac{1}{\lambda_j} \right> = \left[ 1 - \left( \frac{1}{\lambda_j} \right)^2 \right] |0\rangle \left| \frac{1}{\lambda_j} \right> + \left( \frac{1}{\lambda_j} \right) |\lambda_j\rangle |0\rangle \left| \frac{1}{\lambda_j} \right>.
\]

In this case, \( R_Y \) is used to approximate \( R \). \( R_Y \) uses a revolving door (Pauli Y) around the Y axis of the Bloch ball, acting on \( |0\rangle \) to obtain

\[
R_Y |0\rangle \left| \frac{1}{\lambda_j} \right> = \cos(\frac{1}{\lambda_j}) |0\rangle \left| \frac{1}{\lambda_j} \right> + \sin(\frac{1}{\lambda_j}) |\lambda_j\rangle \left| \frac{1}{\lambda_j} \right>.
\]

At this time, the system status is

\[
|q_0q_1q_2q_3\rangle = \sum_{j=1}^{2} (\cos(\frac{1}{\lambda_j})|0\rangle + \sin(\frac{1}{\lambda_j})|1\rangle) \beta_j |2\lambda_j^{-1}\rangle|\mu_j\rangle.
\]

When the result of measuring the first quantum register \( |q_0\rangle = 1 \), the ideal \( |q_3\rangle \) should be \( \beta_1 |\mu_1\rangle + \beta_2 |\mu_2\rangle \), but the actual \( |q_3\rangle \) is \( \sin(\frac{1}{\lambda_1})\beta_1 |\mu_1\rangle + \sin(\frac{1}{\lambda_2})\beta_2 |\mu_2\rangle \). Therefore, when \( \frac{1}{\lambda_1} \) is approximately \( \sin \left( \frac{1}{\lambda_1} \right) \), that is, the value of \( \sin \left( \frac{1}{\lambda_1} \right) \) is very small, the probability of outputting \( |x\rangle \) will be very low, but high fidelity of output can be guaranteed.
This paper choose to draw and simulate through the quantum circuit diagram mode on the Origin Quantum Platform. Quantum gates supported by quantum circuit design[8] just including H gate, NOT gate, phase $\pi$ gate, Pauli X gate, Pauli Y gate, Pauli Z gate, revolving gate, CNOT gate, SWAP gate, Toffoli gate, and CR gate. For this reason, in the quantum circuit of Fig.2, some operations need to be converted. The conversion scheme is shown in Fig.3, which are two unitary matrices and a controlled rotating Y gate in Phase Estimation. The final quantum circuit diagram implemented on the Origin Quantum Platform is shown in Fig.4. The \textit{init} operation is the initialization of $|q_i\rangle$, that is, the assignment of vector $b$. For $|b_A\rangle$, it is equivalent to not execute any operation. For $|b_B\rangle$ is equivalent to NOT gate + H gate.

By analyzing the quantum circuit diagram, it is found that $|q_i\rangle$ is used for the input vector $b$ and also used to store the output $|x\rangle$. Therefore, the required $|x\rangle$ is the magnitude of the probability amplitude when $|q_i\rangle$ is $|0\rangle$ and $|1\rangle$ respectively, if the measurement result of the additional qubit $|q_0\rangle$ is 1. Special emphasis is placed on the need to include signs[9][10]. That is, what ultimately need to extract is the probability amplitude of $|0001\rangle$ and $|1001\rangle$. The theoretical value of $|x\rangle$ is calculated through Python’s numpy package. For different input $|b\rangle$, the corresponding theoretical values of $|x\rangle$ are:

$$|b_A\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix} \rightarrow |x_{A\text{-theory}}\rangle = \begin{pmatrix} 0.75 \\ -0.25 \end{pmatrix}$$

$$|b_B\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -1 \end{pmatrix} \rightarrow |x_{B\text{-theory}}\rangle = \begin{pmatrix} 0.7071 \\ -0.7071 \end{pmatrix}$$

Fig. 3 Operation conversion

Fig. 4 Quantum circuit diagram on the cloud platform
In the completed quantum circuit, the different input $|\beta\rangle$ were measured in turn. Each test iteration is 8192 times (the optional number range is 1-8192 times), and the bar chart of the run results is obtained. Fig. 56 shows the run results.

![Fig. 5 The operation results with value $|\beta_A\rangle$](image)

![Fig. 6 The operation results with value $|\beta_B\rangle$](image)

Through the above analysis, it can be clearly known that the useful value is the probability amplitude of $|0001\rangle$ and $|1001\rangle$. The result marks are respectively $|x_{A,\text{test}}\rangle$ and $|x_{B,\text{test}}\rangle$, then

$$
|x_{A,\text{test}}\rangle = \begin{pmatrix}
\sqrt{0.30407714} \\
-\sqrt{0.00036621}
\end{pmatrix}
= \begin{pmatrix}
0.551432 \\
-0.019137
\end{pmatrix}
$$

$$
|x_{B,\text{test}}\rangle = \begin{pmatrix}
\sqrt{0.05212402} \\
-\sqrt{0.00732422}
\end{pmatrix}
= \begin{pmatrix}
0.228307 \\
-0.085582
\end{pmatrix}
$$

By normalizing the theoretical value and the measured value, the inner product of the two value is calculated and the fidelity is approximately 0.959081 and 0.910324, respectively.

### 3.2. Programming implementation

QPanda2 SDK is a fully functional and efficient quantum software development toolkit. It is an open source quantum computing framework developed by the Origin Quantum Platform. It supports mainstream quantum logic gate operations, and can be optimized for quantum programs on different platforms, and can be adapted to a variety of quantum chips. QPanda2 uses C++ as the classic host language and supports quantum languages written in QRunes and QASM. This section provides a quantum programming based on QPanda2 development environment to verify the $2\times2$ linear equations mentioned in the previous section.

Taking the input $|\beta_i\rangle$ as an example to show the code, the code is as follows:
```cpp
#include <QPanda.h>

USING QPANDA

Q Circuit PE(Qubit* qubit1, Qubit* qubit2, Qubit* qubit3)
{
  QCircuit cir;
  cir << H(qubit1) << H(qubit2) << X(qubit3) << RX(qubit3, -PI / 2) << CNOT(qubit2, qubit3)
    << RZ(qubit3, 3 * PI / 4) << X(qubit3) << RZ(qubit3, 3 * PI / 4) << X(qubit3)
    << CNOT(qubit1, qubit3) << SWAP(qubit1, qubit2) << H(qubit2)
    << CR(qubit1, qubit2, -PI / 2) << H(qubit1);
  return cir;
}

Q Circuit PE_(Qubit* qubit1, Qubit* qubit2, Qubit* qubit3)
{
}

Q Circuit cir;
  cir << H(qubit1) << CR(qubit1, qubit2, PI / 2) << H(qubit2) << SWAP(qubit1, qubit2)
    << CNOT(qubit1, qubit3) << X(qubit3) << RZ(qubit3, -3 * PI / 4) << X(qubit3)
    << RZ(qubit3, -3 * PI / 4) << CNOT(qubit2, qubit3) << RX(qubit3, -PI / 2) << X(qubit3)
    << H(qubit2) << H(qubit1);
  return cir;
}

Q Circuit CR(Qubit* qubit1, Qubit* qubit2, Qubit* qubit3)
{
}

Q Circuit cir;
  cir << SWAP(qubit2, qubit3) << RY(qubit1, PI / 16).control(qubit2)
    << RY(qubit1, PI / 32).control(qubit3) << SWAP(qubit2, qubit3);
  return cir;
}

int main(void)
{
  init();
  auto qubits = qAllocMany(4);
  auto cbits = cAllocMany(4);
  auto prog = CreateEmptyQProg();
  auto circuit = CreateEmptyCircuit();
  Q Circuit PE(Qubit* qubit1, Qubit* qubit2, Qubit* qubit3);
  Q Circuit PE_(Qubit* qubit1, Qubit* qubit2, Qubit* qubit3);
  Q Circuit CR(Qubit* qubit1, Qubit* qubit2, Qubit* qubit3);
  circuit << PE(qubits[1], qubits[2], qubits[3])
    << CR(qubits[0], qubits[1], qubits[2])
    << PE_(qubits[1], qubits[2], qubits[3]);
}```
44 prog << circuit << Measure(qubits[0], cbits[0]) << Measure(qubits[1], cbits[1])
45 << Measure(qubits[2], cbits[2]) << Measure(qubits[3], cbits[3]);
46 auto result = runWithConfiguration(prog, cbits, 10000);
47 for (auto &val : result)
48 {
49    std::cout << val.first << “,” << val.second << std::endl;
50 }
51 finalize();
52 system(“pause”);
53 return 0;
54 }

By calling runWithConfiguration() to get the measurement result of the quantum program, the test iteration is set to 10000 times. For input $b_{\text{Ab}}$, the times of getting $|0001\rangle$ and $|1001\rangle$ are 9920 and 34 respectively. The times were converted into probability and normalized, and the inner product was carried out with the normalized theoretical value to obtain the fidelity of 0.965542. Similarly, for input $b_{\text{Bb}}$, the fidelity is about 0.933564.

4. Conclusion
Regardless of whether it is implemented by the circuit diagram or programming, for different input $b$, the output of this HHL quantum circuit is maintained above 90% accuracy, basically meeting the requirements of solving linear equations. As far as the error is concerned, the error of this quantum circuit is mainly caused by the inherent error caused by the Phase Estimation and the choice of the rotation angle when the $R_Y$ approximates the controlled $R$ operation in the Controlled Rotation. In addition, there is random error caused by repeated measurements. The accuracy has reached the required standard, but the success rate is not high. The overall probability of $|0001\rangle$ and $|1001\rangle$ is low, and further research is needed. However, this paper has carried out a complete study of the HHL algorithm from theory to simulation implementation, and the decomposition of advanced logic gates to basic logic gates, which provides a clear reference for the understanding of the algorithm itself and the realization of quantum computing for the next step. It has laid a good foundation for the application research of the broader Shor algorithm and Grover algorithm[11][12][13].

References
[1] Cai X.D. Optical Quantum Computing and Algorithm Implementation[D]. University of Science and Technology of China, 2015.
[2] Guo G.C. Ten Questions of Quantum [J]. Physics, 2019, 48 (3): 189-192.
[3] Wang K.N. The Parallelism of Quantum Computing and Its Philosophical Significance [C]. Shanxi University 2008 National Doctoral Academic Forum (Philosophy of Science and Technology), 2008:9.
[4] Han Y.J, Li C.F, Guo G.C. Principles and Research Progress of Quantum Computing [J]. Science and Technology Review, 2017, 35(23): 70-75.
[5] Harrow A W, Hassidim A, Lloyd S. Quantum Algorithm for Linear Systems of Equations[J]. Physical Review Letters, 2009, 103(15):150502.1-150502.4.
[6] Cai X.D, Weedbrook, C, Su Z.E, et al. Experimental Quantum Computing to Solve Systems of Linear Equations[J]. Physical Review Letters, 110(23):230501.
[7] Pan J, Cao Y, Yao X, et al. Experimental realization of quantum algorithm for solving linear systems of equations[J]. Physical Review A, 2014, 89(2):022313.
[8] Deng X, Deng Y.X. Qsimulation: A Quantum Computing Simulator Tool[J]. Computer Engineering and Science, 2019, 41(05): 843-850.
[9] Huang H.Y, Bharti K, Rebentrost P. Near-term Quantum Algorithms for Linear Systems of Equations[J]. 2019.
[10] Wei J, Ni M, Zhou M, et al. Quantum Algorithm Research Based on IBM Q platform [J]. Computer Engineering, 2018 (12): 6-12.
[11] Long G.L. Research and Development Progress and Future Prospects of Quantum Computers[J]. People's Forum: Academic Frontiers, 2021(07):44-56.
[12] Lu Y, Feng G.R, Li Y.S, et al. Experimental Digital Quantum Simulation of Interaction Fermion System Dynamics in Real Space[J]. Science Bulletin, 2015, 60(2):241-248.
[13] Harrow A W. Review of Quantum Algorithms for Systems of Linear Equations[J]. Computer Science, 2014.