УДК 004.4’23

АНАЛИЗ ДОСТОИНСТВ И НЕДОСТАТКОВ КВАНТОВЫХ ЭМУЛЯТОРОВ НА ПРИМЕРЕ ВЗАИМОДЕЙСТВИЯ С ПОЛЬЗОВАТЕЛЕМ

Третьяк М.А., Щекатурин А.Е., Пилипенко И.А., Кравченко В.О., Черкесова Л.В.
ФГБОУ ВПО «Донской государственный технический университет», Ростов-на-Дону, e-mail: reception@donstu.ru

В данной статье были рассмотрены наиболее популярные квантовые эмуляторы, использующиеся для квантовых вычислений и позволяющие эмулировать процесс работы квантового компьютера. Была проведена работа по изучению квантовых эмуляторов, выявлению и описанию их индивидуальных особенностей, составлению рекомендаций для более удобного начала работы с ними, а также описанию их достоинств и недостатков. Задача эмуляторов состоит в том, чтобы предоставлять пользователю возможность отладки и запуска программ или схем, предназначенных для работы на квантовом компьютере. Чтобы решить эту задачу, нужно эмулировать логические квантовые вентили, которые являются базовым элементом квантового компьютера. Они преобразуют входящие состояния кубитов на выходные по определенному закону, и в отличие от многих классических логических вентилей всегда являются обратимыми. В этом принципиальном моменте эмуляторы одинаковы, но есть различия в интерфейсах, полноте документации и т.д. Именно взаимодействию с пользователем будет посвящена эта статья. Ввиду того что работа с реальными квантовыми компьютерами доступна лишь узкому кругу исследователей, эмуляторы просто необходимы для проверки гипотез или алгоритмов.

Ключевые слова: кубит, эмулятор, гейт, квантовый компьютер, квантовое программирование

ANALYSIS OF THE ADVANTAGES AND DISADVANTAGES OF QUANTUM EMULATORS ON THE EXAMPLE OF INTERACTION WITH THE USER

Tretyak M.A., Schekaturin A.E., Pilipenko I.A., Kravchenko V.O., Cherkesova L.V.
Don State Technical University, Rostov-on-Don, e-mail: reception@donstu.ru

This article examined the most popular quantum emulators used for quantum computing and allowing to emulate the process of a quantum computer. Work was done to study quantum emulators, identify and describe their individual characteristics, make recommendations for a more convenient start to work with them, as well as describe their advantages and disadvantages. The task of emulators is to provide the user with the ability to debug and run programs or circuits designed to work on a quantum computer. To solve this problem, it is necessary to emulate logical quantum gates, which are the basic element of a quantum computer. They transform the incoming qubit states to the weekend ones according to a certain law, and unlike many classical logic gates, they are always reversible. At this crucial moment, emulators are the same, but there are differences in interfaces, completeness of documentation, etc. This article will be devoted to user interaction. Due to the fact that working with real quantum computers is available only to a narrow circle of researchers, emulators are simply necessary to test hypotheses or algorithms.

Keywords: qubit, emulator, gate, quantum computer, quantum programming

Современного человека компьютерами уже не удивишь. Они стали чем-то обыденным, давно интегрировались в жизнь, с их помощью можно выполнить широкий ряд задач, реализация которых еще в прошлом веке считалась бы чем-то научно-фантастическим. Рост производительности достигается за счет увеличения количества транзисторов на кристалле процессора, что является достаточно дорогостоящим способом [1]. И несмотря на бурное развитие классических компьютеров, уже сейчас существуют ряд задач, которые они не в состоянии решить [2]. Они обычно требуют экспоненциального роста вычислений относительно входных данных. С рядом подобных задач может справиться квантовый компьютер, чья производительность многократно превосходит возможности классических компьютеров. Это связано с тем, что в процессоре цифрового компьютера все вычисления строятся на основе битов, способных находиться в двух состояниях: 1 либо 0. То есть вся работа сводится к анализу огромного количества данных на предмет соответствия заданным условиям. В основу же квантового компьютера положены кубиты. Кубитом называют квантовые биты, особенностью которых является возможность находиться в состоянии 1 или 0, а также одновременного нахождения в состоянии 1 и 0. Такое состояние называется суперпозицией. Возможности квантового компьютера превосходят возможности классического, так как нет необходимости искать нужный ответ среди множества [3]. В этом случае ответ выбирается из уже имеющихся вариантов с определенной долей вероятности соответствия [4; 5].

Существует такое понятие, как «квантовое превосходство» – это потенциальная способность квантовых вычислительных

устройств решать проблемы, которые классические компьютеры практически не могут, или им потребуется для этого слишком много времени [6]. В той же статье BBC [6] был дан критерий наступления квантового превосходства, по мнению авторов, это – порог вычислительной мощности в 50 кубитов. И хотя квантовое превосходство уже было достигнуто [6], до промышленного производства таких процессоров ещё далеко. Но прикоснуться к будущему IT-сферы возможно уже сегодня, благодаря эмуляторам квантовых компьютеров или облачным платформам, которые взаимодействуют с квантовым компьютером. Различие между эмулятором и облачной платформой заключается в том, что облачная платформа передаёт код программы существующему квантовому компьютеру, после чего передаёт результат обратно пользователю, в то время как эмулятор, пользуясь ресурсами цифрового компьютера, воспроизводит работу идеального квантового компьютера с данной ему программой.

В этой статье мы рассмотрим достоинства и недостатки нескольких эмуляторов, которые, на наш взгляд, лучше всего подходят для обучения квантовому программированию, а именно Quantum Development Kit [7], Quantum Computing Playground [8], jQuantum [9], QuEST [10], Quantum Programming Studio [11], Q-Kit [12]. Однако разнообразие квантовых эмуляторов гораздо больше, и их список представлен на информационном портале Quantiki [13]. В качестве примера работы эмулятора взят алгоритм Шора для числа 15.

1. Quantum Development Kit (QDK) был предложен компанией Microsoft в качестве решения, содержащего огромный инструментарий, необходимый для быстрого изучения основ квантового программирования. Этому способствует собственный язык Q#, разработанный компанией Microsoft специально для реализации квантовых вычислений. Microsoft QDK ориентирован в первую очередь на разработчиков, проявляющих интерес к квантовым вычислениям и желающих изучать квантовое программирование, даже если они не являются специалистами по квантовой физике.

Примечательной особенностью Quantum Development Kit является предоставленная разработчикам возможность вести проект в среде разработки, что весьма удобно для создания больших проектов. QDK интегрирован в среду разработки Microsoft Visual Studio (VS), что станет существенным плюсом для тех, кто уже с ней работал и знаком с процессом построения проекта в ней. На данный момент с помощью QDK можно эмулировать 30 логических кубитов для написания программ на цифровом компьютере. QDK поддерживает написание программ с использованием нескольких языков программирования, таких как Python, Q# или C#. При этом разработка программ может происходить как в VS или VS Code, так и в Notebook Jupyter, используя ядро Q#. Однако QDK не имеет графической оболочки, что может вызвать неудобство для людей, не имеющих опыта программирования. Но взамен графической оболочки и основной документации по QDK Microsoft предлагает сборник учебных пособий Quantum Katas [14], в котором представлены элементы квантовых вычислений, а также руководство по программированию на Q# [15; 16], что делает этот вариант наиболее удобным для новичков.

Подходы к разработке квантовых программ для QDK:

1. Разработка с помощью Python. Данный пакет позволяет использовать функции, написанные на языке квантового программирования Q#, программируя при этом на Python.

2. Разработка с помощью Notebook Jupyter. Jupyter предлагает вести разработку квантовых программ из браузера, создавая так называемые заметки, инструкции и другие составляющие квантовой программы. При этом код находится в записной книжке, состоящей из специальных ячеек, запуск которых компилирует код написанный на Q#, и выводит результат операции. Jupyter предоставляет возможность компиляции и моделирования методов, использующихся в Q#.

3. Разработка в Visual Studio. Visual Studio предлагает разработчику вести проект в уже знакомой среде разработки, предоставляя функцию подсветки синтаксиса, для большего удобства написания приложений, а также расширения для Q# VS Code.

На рис. 1 представлен пример кода, написанный на языке программирования Q# в среде разработки Microsoft Visual Studio.

Результат работы квантовой программы выводится в командную строку. На рис. 2 представлен пример такого вывода.

Достоинства Quantum Development Kit:

– QDK позволяет реализовывать сложные квантовые алгоритмы наравне с написанием программ для классических цифровых компьютеров;

– профилирование;

– открытый исходный код;

– есть возможность оценки и контроля вычислительных ресурсов;

– QDK предоставляет возможность отладки программы, что редко наблюдается в других эмуляторах;

```
function SetBitValue(reg: Int, bit: Int, value: Bool): Int {
 if(value) {return reg ||| (1 <<< bit);}
 else {return reg &&& ~~~(1 <<< bit);}}
operation HelloQ () : Unit {
    mutable c = 0;
 using(qubits = Qubit[7]) {
    H(qubits[0]);
    H(qubits[1]);
    H(qubits[2]);X(qubits[6]);
    CNOT(qubits[0], qubits[5]);
    CNOT(qubits[0], qubits[4]);
    X(qubits[4]);
    CCNOT(qubits[4], qubits[1], qubits[6]);
    CNOT(qubits[6], qubits[4]);
    CCNOT(qubits[5], qubits[1], qubits[3]);
    H(qubits[2]);
    CNOT(qubits[3], qubits[5]);
    Controlled Rz([qubits[1]], (PI() / 2.0, qubits[2]));
    H(qubits[1]);
    Controlled Rz([qubits[0]], (PI() / 4.0, qubits[2]));
    Controlled Rz([qubits[0]], (PI() / 2.0, qubits[1]));
    H(qubits[0]);
    SWAP(qubits[2], qubits[0]);
    set c = SetBitValue(c, 2, ResultAsBool(M(qubits[2])));
    set c = SetBitValue(c, 1, ResultAsBool(M(qubits[1])));
    set c = SetBitValue(c, 0, ResultAsBool(M(qubits[0])));
    ResetAll(qubits);   }
 let answer = IntAsString(c);
 Message(answer);
}
```

Рис. 1. Реализация алгоритма Шора в эмуляторе Microsoft QDK

```
4

C:\Users\jfisto\source\repos\QSharpApplication3
cc 6536) завершил работу с кодом 0.
```

Рис. 2. Результат алгоритма Шора в эмуляторе QDK

– возможность написания модульных тестов для тестирования квантовых программ;

– можно вычислить идентификатор кубита, поместив точку останова в код;

– есть возможность работы в облаке Azure с поддержкой до 40 кубитов [17];

– широкий набор примеров реализации алгоритмов на Q# при помощи QDK;

– большой набор библиотек квантового вычисления с открытым исходным кодом.

Недостатки Quantum Development Kit:

– программа, написанная на Q#, не имеет возможности самоанализа;

– нет инструментов для изображения схемы;

– вычисления в облаке Azure не доступны в России;

– возможны ошибки производительности при работе с Jupyter.

2. Quantum Computing Playground (QCP), созданный WebGL Chrome в экспериментальных целях, имеет свой собственный язык сценариев и обладает удобным графическим веб-интерфейсом IDE. При этом Quantum Computing Playground способен имитировать квантовые регистры до 22 кубитов. Для начала работы с эмулятором QCP разработчики предлагают изучить основы синтаксиса языка программирования QScript, а также встроенные команды и математические функции, которые приведены в образцах программ. При этом эмулятор обладает руководством по написанию кода и работы с некоторыми гейтами, а также описанием алгоритмов Шора и Гровера.

Quantum Computing Playground предоставляет возможность использовать отладчик, который позволяет отслеживать поведение написанной программы, а также возможность сохранения кода по URL-адресу. Данный эмулятор вряд ли подойдет человеку без опыта программирования ввиду недостаточного объема обучающего ма-

териала. На рис. 3 представлена реализация алгоритма Шора в веб-эмуляторе QCP.

В эмуляторе реализован графический способ наблюдения за состояниями кубитов в 2D или 3D, где точки описывают суперпозицию кубитов, а их цвет или высота полос описывают амплитуду и фазу заданной суперпозиции. На рис. 4 представлен результат работы квантового алгоритма Шора.

Достоинства Quantum Computing Palyground:

– открытый исходный код [18];

– поддержка отладки кода программы, написанного на собственном языке программирования QScript;

– кроссплатформенность.

Недостатки Quantum Computing Palyground:

– при наличии большого объема кода или использования большого числа искусственных кубитов замедлит работу шейдеров программы;

– высокое потребление оперативной и графической памяти.

```
proc FindFactors N
  x = 0
  if N < 15
    Print "Invalid number!"
    Breakpoint
  endif
  width = QMath.getWidth(N)
  twidth = 2 * width + 3
  for x; (QMath.gcd(N, x) > 1) || (x < 2); x
    x = Math.floor(Math.random() * 10000) % N
  endfor
  Print "Random seed: " + x
  for i = 0; i < twidth; i++
    Hadamard i
  endfor
  ExpModN x, N, twidth
  for i = 0; i < width; i++
    MeasureBit twidth + i
  endfor
  InvQFT 0, twidth
  for i = 0; i < twidth / 2; i++
    Swap i, twidth - i - 1
  endfor
  for trycnt = 100; trycnt >= 0; trycnt--
    Measure
    c = measured_value
    if c == 0
      Print "Measured zero, try again."
      continue
    endif
    q = 1 << width
    Print "Measured " + c + " (" + c / q + ")"
    tmp = QMath.fracApprox(c, q, width)
    c = tmp[0];
    q = tmp[1];
    Print "fractional approximation is " + c + "/" + q
    if (q % 2 == 1) && (2 * q < (1 << width))
      Print "Odd denominator, trying to expand by 2."
      q *= 2
    endif
    if q % 2 == 1
      Print "Odd period, try again."
      continue
    endif
    Print "Possible period is " + q
    a = QMath.ipow(x, q / 2) + 1 % N
    b = QMath.ipow(x, q / 2) - 1 % N
    a = QMath.gcd(N, a)
    b = QMath.gcd(N, b)
    if a > b
      factor = a
    else
      factor = b
    endif
    if (factor < N) && (factor > 1)
      Display "<h2>Success: " + factor + " " + N / factor
      Breakpoint
    else
      Print "Unable to determine factors, try again."
      continue
    endif
  endfor
endproc
VectorSize 16
FindFactors 15
```

Рис. 3. Реализация алгоритма Шора в Quantum Computing Playground

Рис. 4. Результат алгоритма Шора в эмуляторе Quantum Computing Palyground

3. JQuantum – это специализированная программа, взаимодействующая с виртуальной Java-машиной, предназначенная для моделирования процессов, запускаемых только на квантовом компьютере. Проще говоря – для создания программ и визуализации результатов работы алгоритмов, написанных для квантового компьютера. Основной задачей JQuantum является обеспечение прозрачности работы программного кода квантового компьютера. JQuantum помогает визуализировать код, что очень удобно для людей, которые не знакомы с низкоуровневыми языками программирования квантового компьютера, такими как OpenQASM для IBM и т.п. При этом сам эмулятор может использовать максимум 15 кубитов при создании схемы.

Одной из приятных особенностей в JQuantum является возможность пользователя работать как с графическим эмулятором, так и с вычислительным кодом библиотек квантового программирования java, способных выполнять математические операции, которые необходимы для более точного квантового результата, что делает JQuantum удобным для пользователей с опытом программирования и без.

Для того чтобы работать с квантовыми вычислениями в среде разработки от компании JetBrains Intellij IDEA, необходимо импортировать библиотеку quantum, в которой хранится множество функций и гейтов квантового программирования. Данная особенность будет удобна для программирования квантового компьютера на высокоуровневом языке программирования Java или стремительно набирающем популярность языке Kotlin, который полностью совместим с Java.

На панели управления (представленной на рис. 5) слева расположен инструмент инициализации регистров, в котором нам предлагается указать их начальные состояния, а также их количество. Там же расположены команды контроля выполнения квантовой программы. Можно запускать схему пошагово, производя отладку на каждом этапе, при этом в окне вывода результатов видны промежуточные значения выполнения схемы. Можно вернуться к предыдущему шагу, запустить всю схему целиком с окончательным выводом результата или же инициализировать схему заново, для нового запуска квантовой программы, – очень вариативно.

На панели проектирования схемы представлен набор основных гейтов, использующихся при написании квантовых программ, а также инструменты редактирования схемы. Рабочая область представляет собой набор инициализированных кубитов, на которых располагаются гейты. Окно вывода позволяет пользователю следить за состоянием регистров x и y, выводя в них результаты выполнения квантовой программы в процессе ее работы. На рис. 5 представлен пример квантовой схемы алгоритма Шора в эмуляторе JQuantum.

На рис. 6 представлен результат работы схемы алгоритма Шора в эмуляторе JQuantum. Результат работы схемы выводится в регистре y в двоичной и целочисленной записи числа.

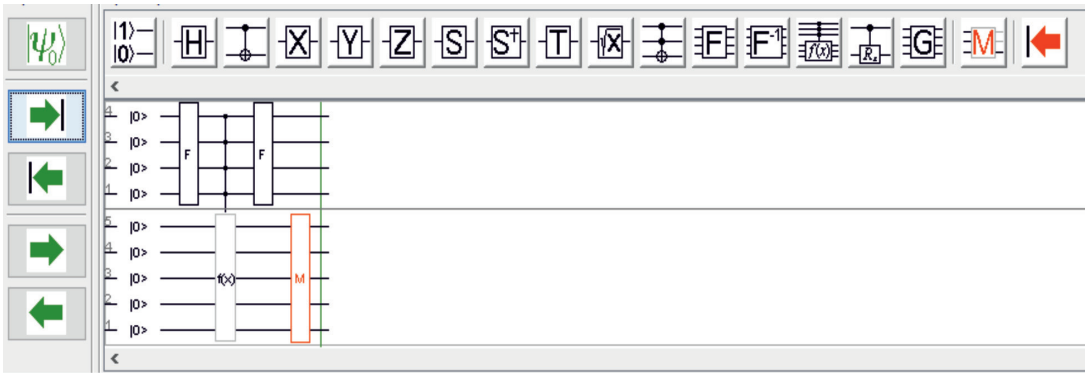

Рис. 5. Реализация схемы алгоритма Шора в эмуляторе JQuantum

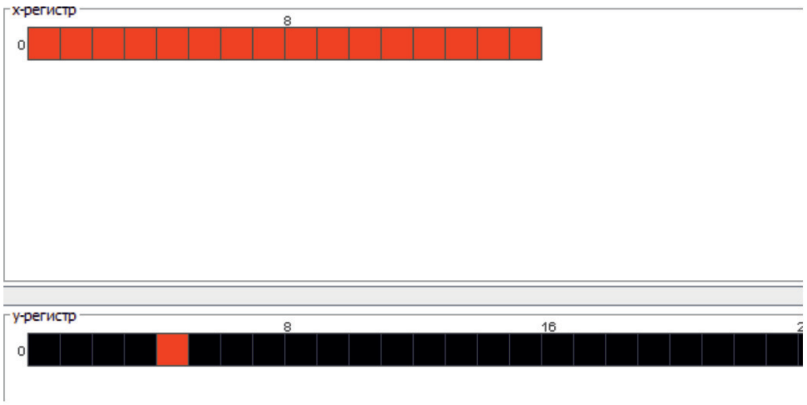

Рис. 6. Результат работы алгоритма Шора в эмуляторе JQuantum

Достоинства JQuantum:

– исходный код [19] находится в открытом доступе;

– кроссплатформенность. JQuantum может стабильно работать с версии Java 5, которая находится в свободном доступе на сайте Oracle;

– эмулятор можно запустить как апплет из браузера или предварительно установив исполняемый файл jar на рабочий стол;

– повторяемость экспериментов;

– чрезвычайная надежность вычислений;

– масштабируемость.

Недостатки JQuantum:

– данная библиотека не может работать в экстремальном масштабе;

– эмулятор не решает проблем, связанных с оптимизацией времени работы программы, сравнимого с теоретическими квантовыми компьютерами;

– для стабильного использования эмулятора необходимо установить Java 5+;

– для изменения схемы необходима полная или частичная инициализация гейтов схемы;

– не полный набор гейтов, требующий создания недостающих из уже имеющихся;

– нет возможности запуска программ на реальном квантовом компьютере.

4. QuEST – это первый GPU-симулятор универсальных квантовых цепей с открытым исходным кодом, гибрид OpenMP и MPI. Созданный как библиотека C, он разработан таким образом, что пользовательский код может быть легко развернут на любой платформе от ноутбука до суперкомпьютера. QuEST способен моделировать общие квантовые схемы общих однокубитных и многокубитных управляемых вентилей в чистых и смешанных состояниях, представленных в виде векторов состояний и матриц плотности, и в присутствии декогеренции [20]. На рис. 7.1 и 7.2 представлена реализация алгоритма Шора, написанного на языке программирования C++.

Достоинства QuEST:

– низкое потребление оперативной памяти на небольшом количестве кубитов [20];

– открытый исходный код;

– кроссплатформенность;

– распределение вычислений на несколько ядер процессора;

– возможность перенести вычисления на графический процессор.

Недостатки QuEST:

– нет инструментов для графического представления схемы;

– вся документация представлена только на английском языке.

5. Quantum Programming Studio (QPS) – это графический пользовательский веб-интерфейс для разработки квантовых алгоритмов и выполнения их на симуляторах или на реальных квантовых компьютерах, разработанный группой Quantastica в 2019 г.

```
7  void swap(Qureg qubits, const int q1, const int q2) {
8      controlledNot(qubits, q1, q2);
9      controlledNot(qubits, q2, q1);
10     controlledNot(qubits, q1, q2);
11 }
12 void ccx(Qureg qubits, const int q1, const int q2, const int q3) {
13     hadamard(qubits, q3);
14     controlledNot(qubits, q2, q3);
15     phaseShift(qubits, q3, -M_PI / 4);
16     controlledNot(qubits, q1, q3);
17     tGate(qubits, q3);
18     controlledNot(qubits, q2, q3);
19     phaseShift(qubits, q3, -M_PI / 4);
20     controlledNot(qubits, q1, q3);
21     tGate(qubits, q2);
22     tGate(qubits, q3);
23     hadamard(qubits, q3);
24     controlledNot(qubits, q1, q2);
25     tGate(qubits, q1);
26     phaseShift(qubits, q2, -M_PI / 4);
27     controlledNot(qubits, q2, q3);
28 }
```

Рис. 7.1. Реализация алгоритма Шора в эмуляторе QuEST

```
29 int main(int argc, char* argv[]) {
30     QuESTEnv env = createQuESTEnv();
31     Qureg qubits = createQureg(7, env);
32     int measured[7];
33     hadamard(qubits, 0);
34     hadamard(qubits, 1);
35     hadamard(qubits, 2);
36     pauliX(qubits, 6);
37     controlledNot(qubits, 0, 5);
38     controlledNot(qubits, 0, 4);
39     pauliX(qubits, 4);
40     ccx(qubits, 4, 1, 6);
41     controlledNot(qubits, 6, 4);
42     ccx(qubits, 5, 1, 3);
43     hadamard(qubits, 2);
44     controlledNot(qubits, 3, 5);
45     controlledPhaseShift(qubits, 1, 2, M_PI / 2);
46     hadamard(qubits, 1);
47     controlledPhaseShift(qubits, 0, 2, M_PI / 4);
48     controlledPhaseShift(qubits, 0, 1, M_PI / 2);
49     hadamard(qubits, 0);
50     swap(qubits, 2, 0);
51     measured[2] = measure(qubits, 2);
52     measured[1] = measure(qubits, 1);
53     measured[0] = measure(qubits, 0);
54     destroyQureg(qubits, env);
55     destroyQuESTEnv(env);
56     return 0;
57 }
```

Рис. 7.2. Реализация алгоритма Шора в эмуляторе QuEST

```
4
--------------------------------
Process exited after 1.586 seconds with return value 0
Для продолжения нажмите любую клавишу . . .
```

Рис. 8. Результат работы алгоритма Шора в эмуляторе QuEST

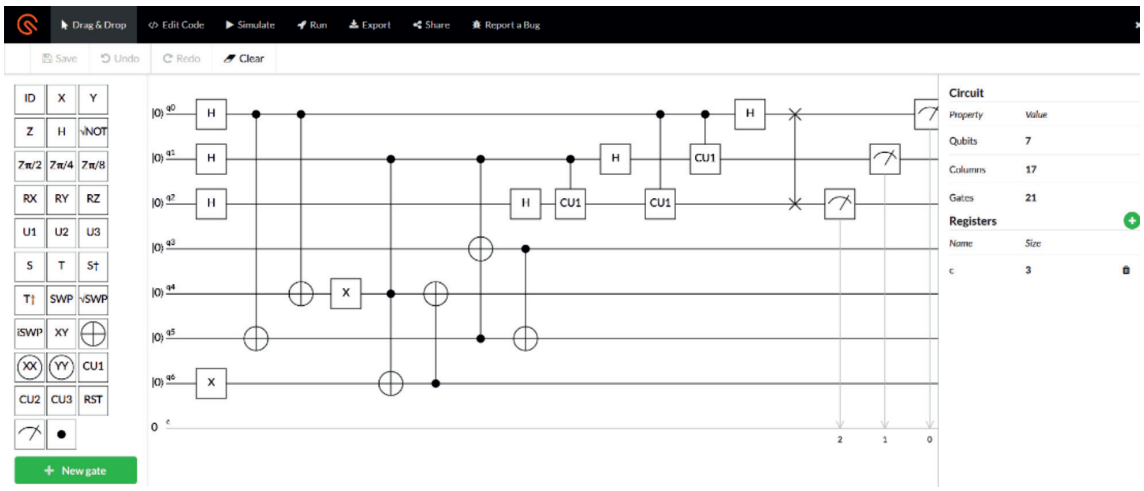

Рис. 9. Реализация схемы алгоритма Шора в эмуляторе QPS

Основной особенностью является кроссплатформенность, заключающаяся в возможности запускать программы в различных эмуляторах или существующих на момент написания статьи квантовых компьютерах. Помимо этого, в QPS существует возможность экспорта кода в наиболее популярные языки квантового программирования.

Данный эмулятор предназначен как для специалистов, так и для людей, которые только начинают постигать азы квантового программирования.

На рис. 9 представлен пример алгоритма Шора в эмуляторе Quantum Programming Studio.

Схема слегка отличается от традиционной в связи с тем, что некоторые необходимые алгоритмы были заранее реализованы. Результат выводится в похожем наглядном виде, представленном на рис. 10.

Достоинства квантового эмулятора QPS:

– QPS является веб-приложением и, как следствие, доступен для любого устройства с выходом в Интернет;

– кроссплатформенность;

– возможность экспорта кода на другие языки;

– наглядное построение схем;

– возможность запустить программу на квантовом компьютере;

– открытый исходный код.

Classical registers

Register	Bin	Hex	Dec
c	100	4h	4

Local state

Qubit	Measured	Probability of 1	θ °deg	φ °deg	Bloch
q0	0	0	0	0	
q1	0	0.5	90	0	
q2	1	0.5	90	0	
q3	1	0.25	60	0	
q4	1	0.75	120	0	
q5	0	0.25	60	0	
q6	1	0.75	120	0	

Рис. 10. Результат схемы алгоритма Шора в эмуляторе QPS

Недостатки квантового эмулятора QPS:

– QPS является веб-приложением, и, как следствие, нет возможности автономной работы;

– запуск схемы с 24 кубитами и больше приведёт к падению браузера;

– сравнительно небольшое количество заранее реализованных алгоритмов.

6. Quantum-Kit (Q-Kit) – это графический симулятор квантовых цепей. Он позволяет строить и проектировать квантовые схемы, визуализируя влияние операций с квантовыми гейтами в виде распределений квантовых состояний или на блоховскую сферу.

Q-Kit – бесценный инструмент в интуитивном понимании квантовых цепей, таких как квантовая телепортация, факторизация Шора или алгоритмы поиска Гровера. Это позволяет экспериментировать с различными квантовыми алгоритмами путем моделирования кубитов для изучения сложных квантовых систем на компьютере, а не в лаборатории. Это также позволяет тестировать квантовое программное обеспечение на эмуляторе перед запуском onquantum-оборудования. Q-Kit имеет приложения не только для исследования, разработки и проверки квантовых алгоритмов, но и для интуитивного понимания квантовых схем, как высокоуровневого предшественника квантового программирования.

Данный эмулятор хорошо подойдёт для тех, кто только начал разбираться с квантовым программированием, ввиду простого интерфейса и наглядности вычислений. На рис. 11 представлена реализация алгоритма Шора в среде эмулятора Q-Kit.

На рис. 12 представлен результат работы схемы алгоритма Шора.

Достоинства квантового эмулятора Q-Kit:

– удобный интерфейс;

– относительная быстрота работы схем [21];

– простое построение схем;

– кроссплатформенность;

– большое количество встроенных алгоритмов и гейтов;

– возможность создавать собственные гейты.

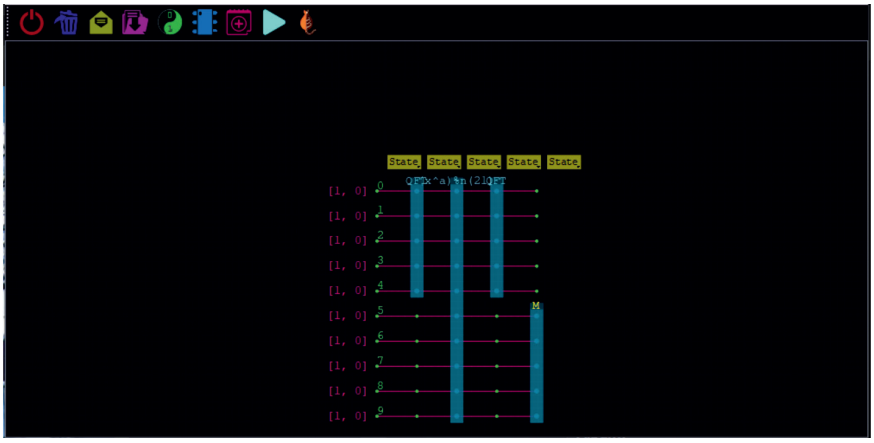

Рис. 11. Реализация схемы алгоритма Шора в эмуляторе Q-Kit

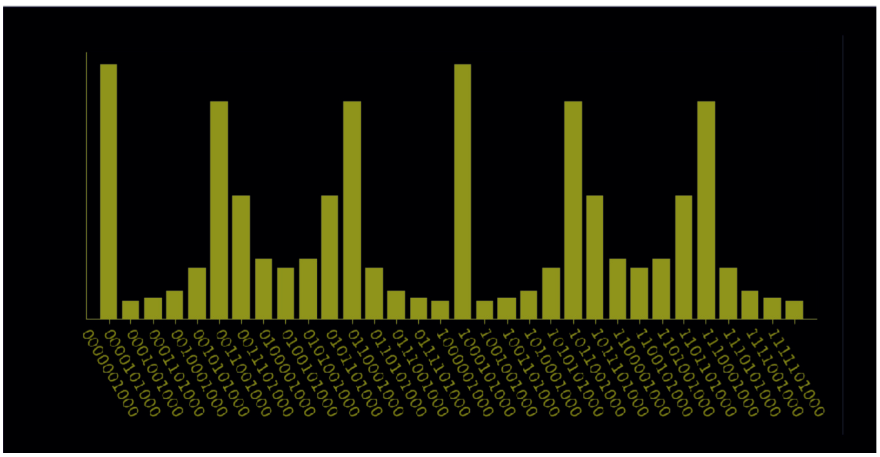

Рис. 12. Результат работы схемы алгоритма Шора в эмуляторе Q-Kit

Сравнительная таблица платформ

	QDK	QCP	JQuantum	QuEST	QPS	Quantum-Kit
Поддерживаемые платформы	Windows, macOS, Linux	Web	Windows	Windows, macOS, Linux	Web	Windows, macOS, Linux
Исходный код	(+)	(+)	(+)	(+)	(+)	(-)
Графическое приложение или библиотека	Библиотека Q#	Web-приложение	Графическое приложение	Библиотека для C++	Web-приложение	Графическое приложение
Количество логических кубитов	32	22	15	29	24	30
Количество реализованных гейтов	17	21	16	29	30	21
Возможность создать свои гейты	(+)	(+)	(-)	(+)	(+)	(+)
Наличие внутреннего языка программирования	(+)	(+)	(+)	(+)	(+)	(-)
Документация	(+)	(-)	(+)	(+)	(+)	(-)

Недостатки квантового эмулятора Q-Kit:
– ограничение в 20 кубит;
– присутствие визуальных багов;
– отсутствует встроенный язык.

На сегодняшний день имеется большое количество разнообразных эмуляторов квантовых компьютеров, которые могут стать хорошим инструментом в руках человека, исследующего квантовые вычисления. На просторах Интернета хранится огромное число методических пособий, руководств и документов, описывающих принципы работы с квантовыми эмуляторами [22].

В таблице приведены обобщенные результаты данного исследования.

Выводы

В данной статье был рассмотрен ряд квантовых эмуляторов, которые хорошо подходят для начала работы с квантовыми вычислениями. При этом их всех объединяет то, что начало работы с данными эмуляторами не вызовет у пользователя каких-либо затруднений. Все рассмотренные эмуляторы обладают удобным интерфейсом, который также способствует повышению продуктивности пользователя при работе с кодом. Для большей части эмуляторов есть актуальная, структурированная и понятная документация, изучение которой поможет освоить квантовое программирование и написать первую квантовую программу. При этом эмуляторы поддерживают возможность отладки программы, что позволяет оценить корректность работы кода. В данной статье были выбраны и проанализированы наиболее подходящие эмуляторы для начала работы с квантовыми вычислениями. Невозможно сказать, что какой-то из них лучше или хуже. Можно лишь рассмотреть их особенности, которые могут быть удобны на тех или иных уровнях квантового программирования. Также возможно разделить эмуляторы на следующие группы для удобства:

– для тех, кто только начинает изучать квантовые вычисления, хорошо подойдут QPS, jQuantum, QCP, с примерами квантовых алгоритмов;

– для тех, кто знаком с программированием на разных языках, QDK, QuEST и jQuantum – отличный выбор для начала и продолжения работы с квантовыми вычислениями;

– для изучения поведения схем отлично подойдут Q-kit, JQuantum, QPS;

– а QPS – хороший инструмент для изучения квантовых языков программирования в общем виде.

Таким образом, квантовые вычисления теперь доступны не только узкой прослойке исследователей, трудящихся в передовых учреждениях с квантовыми компьютерами в их физическом виде. Доступ к вышеперечисленным эмуляторам так прост, что начать заниматься квантовым программированием под силу даже студенту. Осталось только выбрать подходящий эмулятор из списка.

Список литературы

1. Intel. Amazing Performance and Responsiveness. [Electronic resource]. URL: https://www.intel.ru/content/www/ru/ru/homepage.html (date of access: 27.06.2020).

2. Dunne P.E. An annotated list of selected NP-complete problems. Dept. of Computer Science, University of Liverpool. 2008. [Electronic resource]. URL: https://cgi.csc.liv.ac.uk/~ped/teachadmin/COMP202/annotated_np.html (date of access: 27.06.2020).

3. Georgescu M., Ashhab S., Franco Nori. «Quantum SimulationI». Cornell University. 2013. [Electronic resource]. URL: https://arxiv.org/abs/1308.6253 (date of access: 27.06.2020).

4. Душкин Р.В. Квантовые вычисления и функциональное программирование. М.: ДМК-Пресс, 2015, 232 с.

5. Berkeley of University of Califoria, «Quantum Mechanics and Quantum Computation». [Electronic resource]. URL:https://www.edx.org/course/quantum-mechanics-and-quantum-computation#.VOugFrDF9sA (date of access: 27.06.2020).

6. National Academies of Sciences, Engineering, and Medicine 2018. Quantum Computing: Progress and Prospects. Washington, DC: The National Academies Press. DOI: 10.17226/25196.

7. Quantum Development Kit. [Electronic resource]. URL: https://www.microsoft.com/en-us/quantum/development-kit (date of access: 27.06.2020).

8. Quantum Computing Playground. [Electronic resource]. URL: http://www.quantumplayground.net (date of access: 27.06.2020).

9. Andreas de Vries A Quantum Computer Simulator. [Electronic resource]. URL: http://jquantum.sourceforge.net/ (date of access: 27.06.2020).

10. QuEst. Quantum Exact Simulation Toolkit. [Electronic resource]. URL: https://quest.qtechtheory.org/ (date of access: 27.06.2020).

11. Quantum Programming Studio. [Electronic resource]. URL: https://quantum-circuit.com (date of access: 27.06.2020).

12. Quantum-Kit. [Electronic resource]. URL: https://sites.google.com/view/quantum-kit (date of access: 27.06.2020).

13. Quantiki. List of QC simulators. [Electronic resource]. URL: https://quantiki.org/wiki/list-qc-simulators (date of access: 27.06.2020).

14. GitHub. Quantum Ktas. [Electronic resource]. URL: https://github.com/Microsoft/QuantumKatas/ (date of access: 27.06.2020).

15. GitHub. Microsoft Quantum Development Kit Samples. [Electronic resource]. URL: https://github.com/microsoft/Quantum (date of access: 27.06.2020).

16. Microsoft documentation. Microsoft Quantum Development Kit. [Electronic resource]. URL: https://docs.microsoft.com/en-us/quantum (date of access: 27.06.2020).

17. Microsoft Azure Documentation. Azure Quantum. [Electronic resource]. URL: https://azure.microsoft.com/ru-ru/services/quantum/?cdn=disable (date of access: 27.06.2020).

18. GitHub. Quantum-Computing-Playground. [Electronic resource]. URL: https://github.com/gwroblew/Quantum-Computing-Playground (date of access: 27.06.2020).

19. GitHub. JQuantum. [Electronic resource]. URL: https://github.com/chasenorman/JQuantum (date of access: 27.06.2020).

20. Tyson Jones «QuEST and High Performance Simulation of Quantum Computers». 2018. [Electronic resource]. URL: https://arxiv.org/pdf/1802.08032.pdf (date of access: 27.06.2020).

21. Archana Tankasala «Q-Kit: SimulatingShor's Factorization of 24-Bit Number on Desktop». [Electronic resource]. URL: https://arxiv.org/ftp/arxiv/papers/1908/1908.07187.pdf (date of access: 27.06.2020).

22. Quantum Computing Report. Education. [Electronic resource]. URL: https://quantumcomputingreport.com/resources/education/ (date of access: 27.06.2020).