САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Математико-механический факультет

Кафедра системного программирования

Коноплев Юрий Михайлович

Разработка симулятора квантового вычислителя для обеспечения учебного и научного процесса по теме квантовых вычислений

Дипломная работа

Допущена к защите.
Зав. кафедрой:
д.ф.-м.н., профессор **Терехов А.Н.**

Научный руководитель:
к.ф.-м.н., доцент **Сысоев С.С.**

Рецензент:
к.ф.-м.н., доцент **Вахитов А.Т.**

Санкт-Петербург

2013

SAINT-PETERSBURG STATE UNIVERSITY

Mathematics&Mechanics Faculty

Software Engineering Chair

Konoplev Iurii

Quantum computation simulator development for study and research in quantum computation topic

Graduation Thesis

Admitted for defence.
Head of the chair:
Professor **A. N. Terekhov**

Scientific supervisor:
PhD **S.S. Sysoev**

Reviewer:
PhD **A.T. Vakhitov**

Saint-Petersburg
2013

Оглавление

Введение....3
Постановка задачи....6
Основные понятия....7
Кубит, система из нескольких кубитов....7
Измерение состояния квантовой системы....7
Эволюция квантовой системы, квантовые вентили....8
Оракул функции....9
Решение....10
Простая схема вычислителя....10
Первый подход (наивный)....10
Второй подход....11
Реализация....14
Технологии....14
Состояние системы....14
Набор квантовых операторов....14
Действие оператора....14
Формат передачи схемы вычислителя....15
Измерение состояния системы....16
Эксперименты....17
Задача Дойча – Джоза....17
Задача Дойча....18
Задача Бернштейна-Вазирани....18
Результаты....20
Литература....21

Введение

Развитие вычислительной техники - это постоянные эксперименты и открытия. Всем известен закон Мура, говорящий об удвоении вычислительных мощностей классических компьютеров каждые полтора года. Вычислительные машины прошли путь от механических устройств до ЭВМ, основанных на полупроводниках и микросхемах. Каждый новый тип устройства был эффективнее предыдущего – естественно было бы ожидать появления новых, еще более эффективных устройств.

Вместе с устройствами эволюционировали и алгоритмы. В 1936 году Алан Тьюринг предложил математическую модель вычислений, названную впоследствии машиной Тьюринга (Deterministic Turing Machine – DTM) [1]. Модель Тьюринга формализовала понятие алгоритма (способа вычисления некоторой функции на физическом устройстве), что позволило ввести понятие вычислимости и, в дальнейшем, сложности вычислений. Тогда же, в 1936 году, Тьюринг показал, что существуют невычислимые (undecidable) функции – такие функции, которые можно формально определить, но для вычисления которых невозможно предложить алгоритм, работающий конечное время.

Понятие сложности вычислений (количества ресурсов, необходимых для выполнения алгоритма) позволило выделить классы задач, названных неразрешимыми (intractable). В отличие от невычислимых функций, для них существуют алгоритмы, работающие за конечное время. Однако ресурсы, необходимые для выполнения этих алгоритмов, растут слишком быстро (например, экспоненциально) с увеличением размера входных данных. Открытым является вопрос, являются ли неразрешимыми задачи из класса NP-complete (знаменитая гипотеза P!=NP).

Одной из неразрешимых в рамках классической теории вычислений задач является эмуляция эволюции квантовой системы произвольного размера. Основываясь на этом факте, в 1981 году нобелевский лауреат по физике

Ричард Фейнман предложил построить на основе квантовой системы вычислительное устройство. Он предполагал, что система, которую сложно рассчитать на классической архитектуре, может сама оказаться эффективным вычислителем (у квантового компьютера не возникает проблем с эмуляцией самого себя).

Уже в 1985 году появилась работа Дэвида Дойча [2], в которой был предложен первый алгоритм, позволяющий использовать преимущества квантовых вычислений. В 1994 году вышла знаменитая статья Питера Шора [3], описывающая полиномиальный алгоритм факторизации чисел.

В 1996 году Гровером [4] был получен важный результат относительно возможности решения на квантовом вычислителе обобщенной модели задачи из класса NP, при полностью неопределенном виде функции (черном ящике).

Несмотря на высокий интерес к квантовым вычислениям, вызванный алгоритмами Шора и Гровера, на текущий момент существует всего несколько квантовых алгоритмов. Одной из возможных причин подобного положения дел автор считает практическую сложность разработки алгоритмов для квантового компьютера и надеется упростить исследователям этот процесс, предложив эффективный общедоступный эмулятор квантовых вычислений.

Целью данной работы была реализация эмулятора квантового вычислителя на классическом компьютере в виде веб-приложения. Подобный эмулятор позволяет “потрогать руками” алгоритмы, что должно способствовать лучшему освоению теории квантовых вычислений студентами, а также разработке новых алгоритмов.

Постановка задачи

Целью данной работы была разработка эмулятора квантового вычислителя в виде веб-приложения.

Были поставлены следующие задачи:

1. Изучение основ квантовых вычислений
2. Исследование общего вида квантовых вентилей в форме матриц
3. Реализация универсального набора квантовых вентилей
4. Реализация пользовательского интерфейса

Основные понятия

Кубит, система из нескольких кубитов

Простейшей квантовой системой является вектор в двумерном Гильбертовом пространстве, называемый кубитом. Его можно представить в виде суперпозиции двух базисных векторов:

$$(1) \quad A * |0\rangle + B * |1\rangle$$

Система из нескольких кубитов описывается, как тензорное произведение систем её образующих:

$$(\alpha_0|0\rangle + \alpha_1|1\rangle) \otimes (\beta_0|0\rangle + \beta_1|1\rangle) \otimes \dots$$

Из этого следует, что размерность пространства системы из n кубит - 2^n. Отсюда вытекает сложность моделирования квантового компьютера на классическом.

Измерение состояния квантовой системы

Классический бит можно легко измерить. С кубитом всё не так просто, он находится не в одном из двух состояний (0 или 1), а в их суперпозиции. Известно, что не существует способов определения коэффициентов в (1). При измерении кубита мы можем получить как 0, так и 1, с некоторой вероятностью. Вероятность каждого исхода определяется, как квадрат модуля коэффициента при базисном векторе:

$|A|^2$ – вероятность при измерении получить 0

$|B|^2$ – вероятность при измерении получить 1

Эволюция квантовой системы, квантовые вентили

Эволюция квантовой системы описывается, как действие на нее некоторого унитарного оператора U. Этот оператор можно разложить в композицию операторов меньшей размерности.

Далее приведены примеры простейших однокубитных операторов:

- Тождественное преобразование:

$$\sigma_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

- Отрицание:

$$\sigma_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

- Фазовый сдвиг:

$$\sigma_3 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

- Преобразование Адамара:

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

Также возможны вентили, имеющие более одного входа. Например, матрица контролируемого отрицания (C-NOT) имеет вид:

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

Набор квантовых вентилей называют универсальным, если любое унитарное преобразование можно аппроксимировать с любой заданной точностью конечной последовательностью вентилей из этого набора.

Например набор, состоящий из отрицания, фазового сдвига, преобразования Адамара, поворота на π/8и C-NOT, является универсальным.

Оракул функции

Так как все преобразования – это применения унитарных операторов, необходимо действие функции на систему представить так же в виде унитарного преобразования.

Такое преобразование принято называть *оракулом функции* U_f. Действие описывается следующей формулой:

$$U_f : |x\rangle|y\rangle \rightarrow |x\rangle|y \oplus f(x)\rangle$$

Решение

Простая схема вычислителя

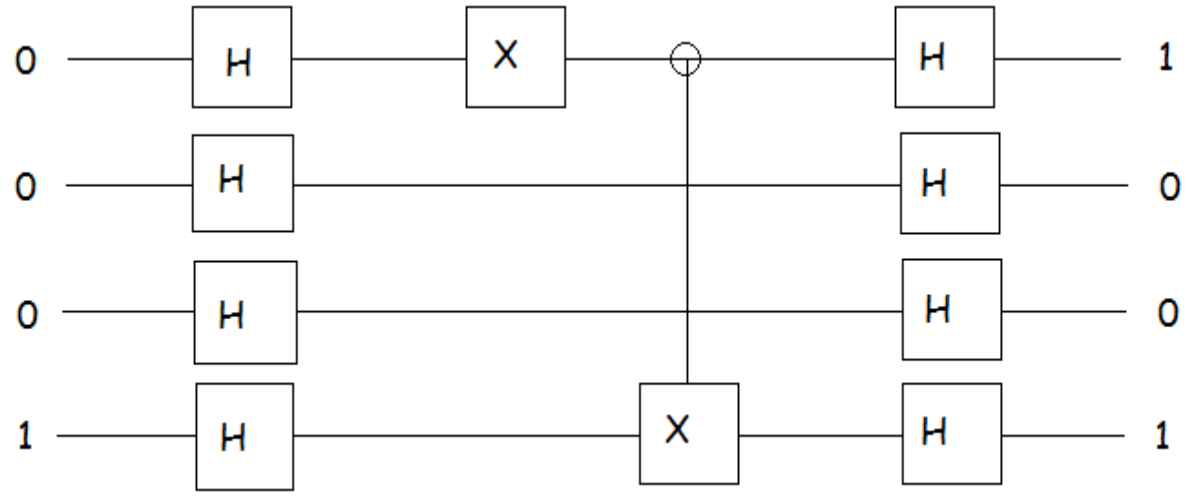

рис.1

На рис.1 представлена схема квантового вычислителя с четыремя разрядами. Для квантового компьютера применение однокубитного вентиля - это просто действие на один кубит системы. При эмуляции на классическом компьютере - это действие матрицы размерностью $2^n \times 2^n$ на вектор, имеющий размерность 2^n, где n - разрядность вычислителя. В данном случае, при n = 4, размерность равна 16. Но, например, при n = 20 размер матрицы становится больше, чем 10^{12}.

Первый подход (наивный)

Первый подход строится на утверждении о том, что состояние квантовой системы из n кубит - тензорное произведение систем (кубит) её образующих.

Утверждение: Матрица оператора, действующего на j-й кубит - это тензорное произведение единичных матриц 2x2 и матрицы самого оператора, входящей в произведение j-м множителем.

Т.е., если оператор должен быть применён к кубиту с номером j , то в

произведении матрица оператора должна находиться на j-м месте:

$$\binom{10}{01} \otimes \binom{10}{01} \otimes \quad \ldots \otimes \underset{j}{H} \otimes \ldots$$

Это утверждение следует из определения понятия состояния квантовой системы, состоящей из нескольких кубит.

Таким образом, чтобы найти состояние системы после применения оператора, нужно просто умножить вектор состояния системы на полученную матрицу оператора.

Минусы подхода:

1. Затраты памяти на хранение матриц размерностью $2^n \times 2^n$.
2. При подсчёте матрицы оператора происходит огромное количество лишних элементарных умножений на 0 и 1.
3. Контролируемые операторы (например, CNOT) не раскладываются в тензорное произведение элементарных вентилей, и для формирования их матрицы нужен другой алгоритм.

Данный подход был реализован. Эмуляция производилась на обычном офисном ПК с объемом оперативной памяти 2Гб. Было выяснено, что система работает при разрядности вычислителя не более 12 кубит. При 13 кубитах выдавалась ошибка памяти.

Мы решили более пристально изучить процесс действия матрицы оператора на систему.

Второй подход

Рассмотрим состояние системы не как вектор размерности 2^n, а как суперпозицию базисных векторов пространства

$$\alpha_0|0\rangle + \alpha_1|1\rangle + \ldots + \alpha_{2^n-1}|2^n-1\rangle \qquad (2)$$

Далее будем рассматривать действие оператора не на всю систему, а только на базисные вектора с соответствующими коэффициентами.

Таким образом, на каждом шаге вычисляется не вся матрица, а только один столбец, соответствующий данному базисному вектору.

Так же стоит заметить, что часто большая часть коэффициентов при базисных векторах равна нулю. Для таких векторов мы вообще не будем производить никаких вычислений, так как на результат они не влияют.

Утверждение: Столбец матрицы оператора имеет следующий вид: у него либо один элемент равен 1, а остальные равны 0, либо два элемента отличаются от 0.

Доказательство:

Рассмотрим (j+1)-й столбец матрицы оператора. Он будет соответствовать вектору, полученному в результате применения оператора к вектору $|j\rangle$.

Пусть оператор применяется к кубиту с номером k (нумерация кубитов начинается с 0).

Допустим, что матрица оператора, действующего на систему из одного кубита имеет вид :

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \quad (3).$$

Рассмотрим два случая.

1-й случай:

Изменяемый кубит имеет значение 1.

Т.е. вектор, соответствующий этому кубиту имеет вид $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$. Результат применения оператора (3) будет равен:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} b \\ d \end{pmatrix},$$

что соответствует системе, состоящей из одного кубита и имеющей состояние

$$b|0\rangle + d|1\rangle$$

Таким образом исходный вектор состояния $|j\rangle$ переходит в композицию :

$$b|j\text{-}2^{n-k+1}\rangle + d|j\rangle$$

Этому состоянию соответствует вектор размерности 2^n, у которого элемент под номером j равен d, а элемент под номером $j-2^{n-k+1}$ равен b.

2-й случай:

Изменяемый кубит имеет значение 0.

Аналогичными рассуждениями получаем вектор размерности 2^n, у которого элемент под номером j равен a, а элемент под номером $j+2^{n-k+1}$ равен c.

Утверждение доказано.

Из доказанного утверждения следует, что можно делать умножение не на весь столбец, а только на два его ненулевых элемента. Результат применения оператора к системе рассчитывается следующим образом:

$\alpha_0 \cdot h_0 + \alpha_1 \cdot h_1 + \alpha_2 \cdot h_2 + \ldots$

Здесь h_j - вычисленные нами столбцы.

Таким образом, мы избежали лишних умножений, ускорив процесс примерно в 2^n раз.

На каждом шаге мы храним всего два вектора размерностью 2^n: исходное состояние системы, результирующий вектор состояния системы.

Реализация описанного подхода на том же макете позволила эффективно эмулировать вычисления для систем до 24-х кубит включительно.

Реализация

Технологии

Так как итоговой целью работы является общедоступное веб-приложение, было решено использовать сервис **Google App Engine.** [5]
Для реализации серверной части, был выбран высокоуровневый язык программирования **Python**. [6]
Пользовательский интерфейс реализован на **JavaScript** с использованием библиотеки **JQuery.**

Состояние системы

Состояние квантовой системы, состоящей из n кубит, можно представить в виде вектора размерностью 2^n. В программе было решено хранить состояние системы в массиве размером 2^n. Элемент массива на месте j задаёт коэффициент в разложении (2) при базисном векторе $|j\rangle$.

Набор квантовых операторов

Как уже было сказано в разделе «Основные понятия», принято выделять так называемый универсальный набор квантовых вентилей. Напомним, набор квантовых вентилей называют универсальным, если любое унитарное преобразование можно аппроксимировать с любой заданной точностью конечной последовательностью вентилей из этого набора.

Для упрощения построения схем вычислителей в приложении, было решено расширить стандартный универсальный набор. Кроме поворота на угол $\pi/8$ реализованы операторы поворота на углы $\pi/2$, $\pi/4$, $\pi/16$, $\pi/32$, $\pi/64$. Повороты на меньшие углы предлагается считать тождественным преобразованием.

Действие оператора

В разделе «Решение» было представлено два подхода к получению результата действия оператора на квантовую систему. Так как второй способ выигрывает и по затратам памяти и по производительности, он и был выбран в качестве основного.

Далее приведён код функции (Python), реализующей второй подход:

```
def  qubitGate(self, c_i, result, i, n, qbit, A, B, C, D):
        base = 2 ** qbit
        if i & base:                # qubit is |1> so B,D column is applied
            result[i] = result[i] + c_i * D
            i2 = i - base
            result[i2] = result[i2] + c_i * B
        else:                       # qubit is |0> so A,C column is applied
            result[i] = result[i] + c_i * A
            i2 = i + base
            result[i2] = result[i2] + c_i * C
        return result
```

Здесь i – номер базисного вектора, для которого считается результат, c_i – коэффициент при нём в разложении (2), qbit – номер кубита, к которому применяется оператор. A, B, C, D – элементы матрицы однокубитного оператора.

Для того, чтобы получить конечное состояние системы после применения оператора, необходимо вызвать эту функцию для каждого базисного вектора, коэффициент при котором не ноль.

Формат передачи схемы вычислителя

После того, как пользователь создал схему вычислителя, она отправляется на сервер. Формат для передачи был выбран довольно простой.

Схема передаётся в виде строки {a,b,c}{I,H,I}. В первых фигурных скобках передаётся исходное состояние системы. Во вторых скобках последовательно указаны применяемые операторы.

Так, например, для схемы на рис.1 строка будет иметь следующий вид:

{0,0,0,1}{H,H,H,H,X,I,I,I,*,I,I,X,H,H,H,H}.

Символ «*» указывает на то, что данный кубит является контрольным.

Измерение состояния системы

Для симуляции квантовых вычислений необходимо реализовать функцию, которая будет производить «измерение» квантовой системы.

Результатом измерения будет один из базисных векторов, который мы получим с вероятностью, равной квадрату модуля коэффициента при нём. Т.е. вектор $|j\rangle$ мы получим с вероятностью $|\alpha_j|^2$.

Для этого реализуем следующую функцию:

```
def measure(self, state):
    for i in range(len(state)):
        state[i] = state[i].real * state[i].real + state[i].imag * state[i].imag
    r = random.random()
    temp = 0
    for i in range(len(state)):
        temp += state[i]
        if temp >= r:
            return bin(i)
    return bin(len(state) - 1)
```

Эксперименты

Для проверки работоспособности симулятора, на нём были реализованы три алгоритма.

Задача Дойча - Джоза

Задача Дойча — Джоза заключается в определении, является ли функция двоичной переменной *f(n)* постоянной (принимает либо значение 0, либо 1 при любых аргументах) или сбалансированной (для половины области определения принимает значение 0, для другой половины 1). При этом считается априорно известным, что функция либо является константой, либо сбалансирована.

На рис.2 приведена схема вычислителя для решения этой задачи. Функция в примере - сбалансированная. В красной рамке выделен оракул функции.

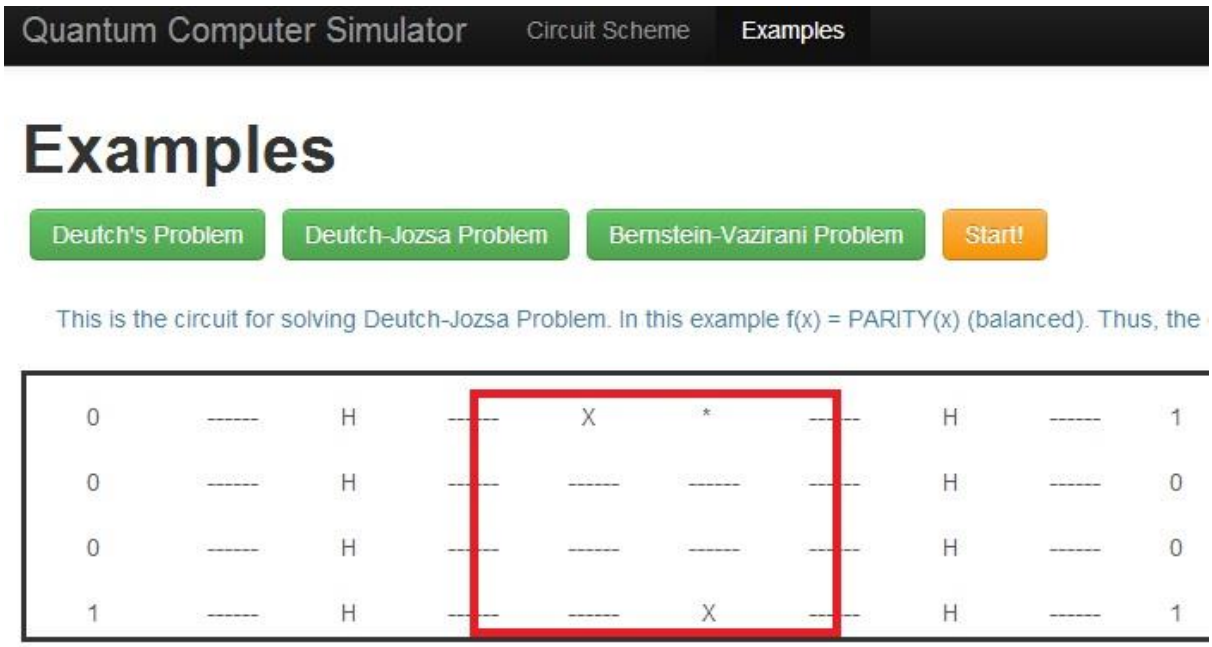

Рис.2 Задача Дойча-Джоза

Задача Дойча

Эта задача является частным случаем предыдущей, только переменная принимает значения 0 и 1. Здесь приведена схема оракула константы (см. рис.3).

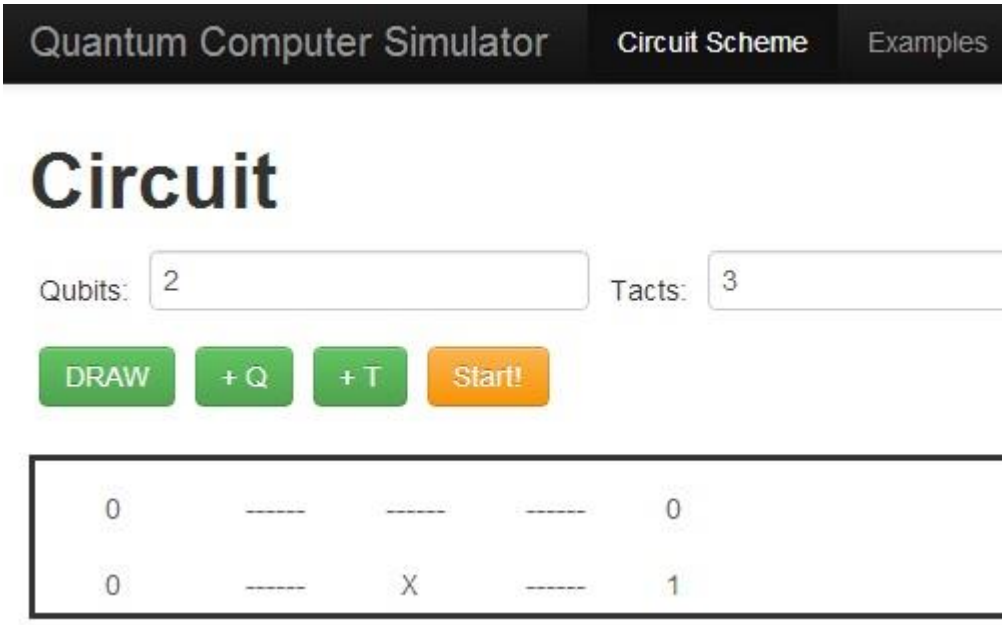

Рис.3 Схема оракула для функции f(x) = 1

Задача Бернштейна-Вазирани

Пусть дана функция $f_a(x) = ax$, где a – n-битовая строка. Задача состоит в нахождении a.

Схема вычислителя для этой задачи отличается от схемы в первой задаче только оракулом функции f. На рис.4 приведена схема оракула для функции $f_a(x) = ax$ где a=163.

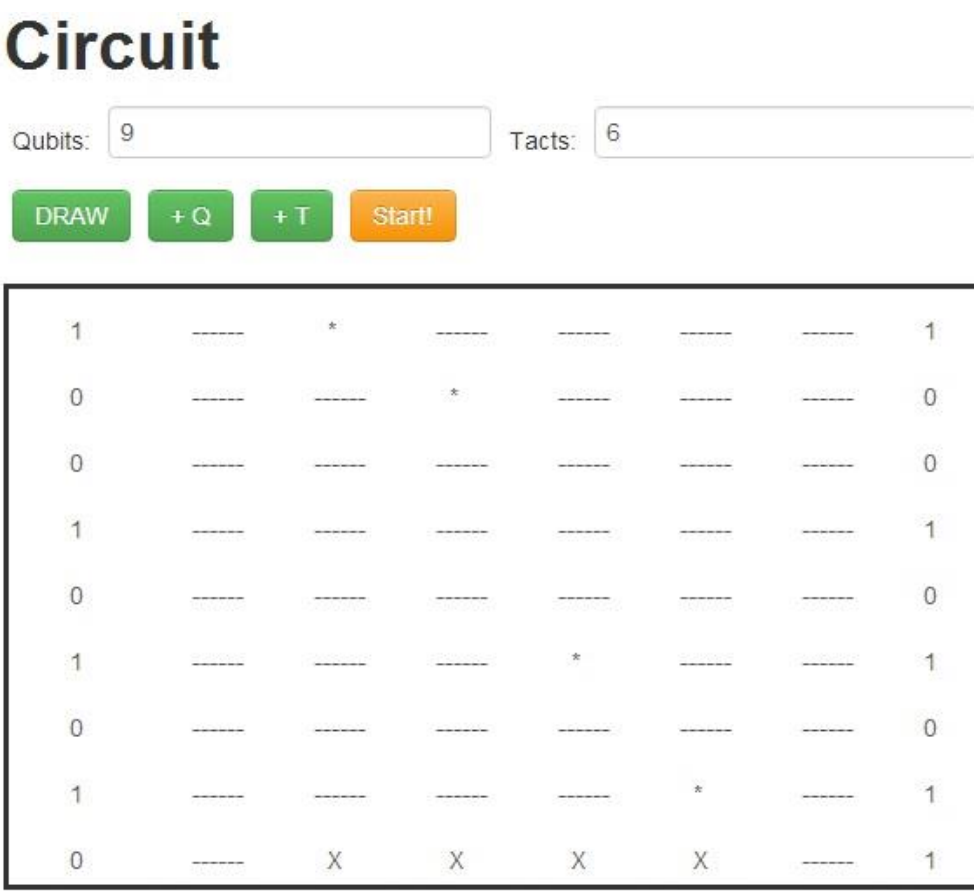

Рис.4 Оракул функции $f_a(x) = ax$ где a=163

Результаты

В процессе работы было рассмотрено два способа представления квантовых вентилей для квантовой системы заданной размерности. Первый способ использует представление этих вентелей, как тензорное произведение матриц из универсального набора.

Второй подход потребовал более тщательного понимания процесса и позволил получить значительную экономию, как по времени, так и по ресурсам памяти.

В результате получен прототип эмулятора квантового вычислителя, доступный по адресу http://qc-sim.appspot.com/ .

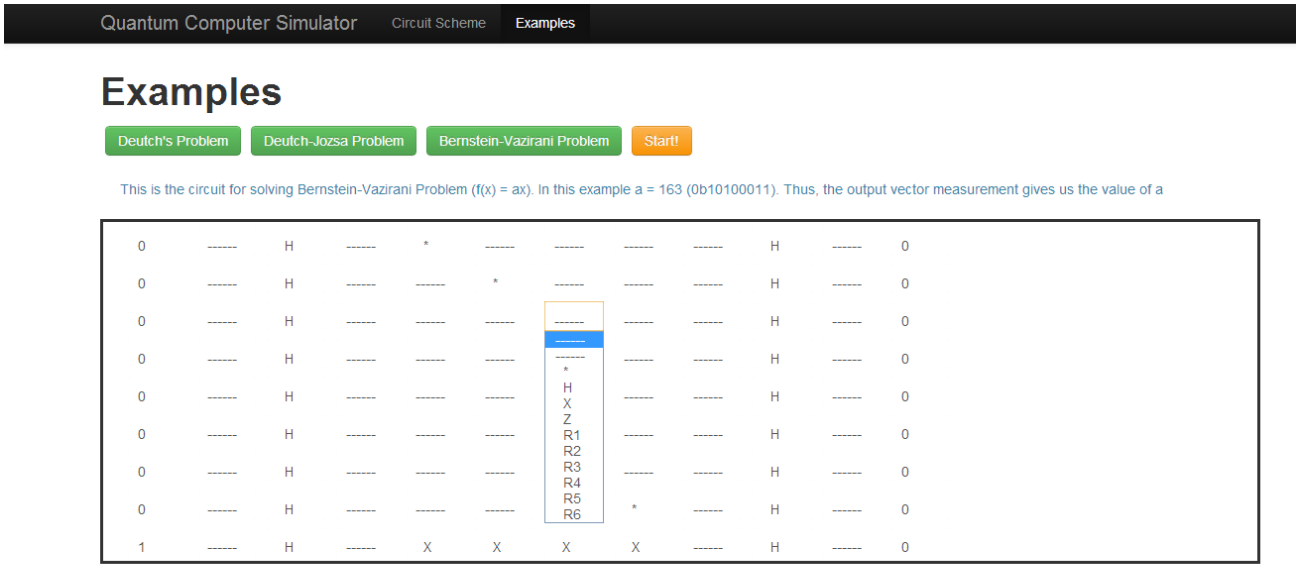

рис.2 Вид симулятора

Так же на сайте в качестве примеров представлены схемы вычислителей для алгоритмов Дойча, Дойча-Джозсы и Бернштейна-Вазирани.

Литература

1 Turing A.M. On Computable Numbers, with an Application to the Entscheidungs Problem // Proceedings of the London Mathematical Society. 1936. Ser. 2, 42, p. 230-265.

2 David Deutsch (1985). "The Church-Turing principle and the universal quantum computer". Proceedings of the Royal Society of London A 400: 97.

3 Peter W. Shor (1994). "Algorithms for quantum computation: discrete logarithms and factoring". Proceedings of the 35th IEEESymposium on Foundations of Computer Science. pp. 124–134.

4 Lov K. Grover (1996). "A fast quantum mechanical algorithm for database search". Proceedings of the Twenty-Eighth Annual ACMSymposium on Theory of Computing. pp. 212–219.

5 https://developers.google.com/appengine/?hl=ru

6 http://python.org