CMPT 376W Project 2

Jiangpei Chen *3*01326516 Chenyu Ru *3*01323578
Nontawat Janpongsri *3*01311427 Luowen Zhu *3*01326420
Zhixin Huang *3*01326521

Abstract

As the summary of the principle reference *You are what you document*, it summarized the documents into three different types. Written documentation, Code documentation, Community documentation. Each type of documentation solves a different problem, we will be following the structure of this document to explain quantum computing, especially virtual quantum computing machine.

1 Written documentation

As the summarize of our project 1, written documentation is the most straight forward document that we feel when we mention "documents". There are different types under the category of written documents. I will use the guide (tutorials) and reference documents as examples to explain the quantum computing, quantum algorithm and more over on virtual quantum computing machine.

In order to have a sense of virtual quantum computing machine, first we need to understand basic quantum terminology and some related definition. After the base is built, then we can start using the documentation to illustrate a virtual quantum computing machine.

1.1 Guides and tutorials

First of all, I found some examples that can explain quantum computing and quantum algorithms.

Quantum Computing is the use of quantum-mechanical phenomena such as superposition and entanglement to perform computation. Wikipedia describes and summarizes quantum computing with the words above, then it mentioned the potential implementation of the quantum computing by using Qubits, and then it gives some introduction of the quantum algorithms.

https://en.wikipedia.org/wiki/Quantum_computing

Here are some other examples that gives a brief introduction of what quantum computing is.

https://docs.microsoft.com/en-us/quantum/overview/what-is-quantum-computing?view=qsharp-preview

What is quantum computing?

10/22/2019 • 5 minutes to read •

There are some problems so difficult, so incredibly vast, that even if every supercomputer in the world worked on the problem, it would still take longer than the lifetime of the universe to solve.

Quantum computers hold the promise to solve some of our planet's biggest challenges - in environment, agriculture, health, energy, climate, materials science, and problems we've not yet even imagined. The impact of quantum computers will be far-reaching and have as great an impact as the creation of the transistor in 1947, which paved the way for today's digital economy.

Quantum computing harnesses the unique behavior of quantum physics to provide a new and powerful model of computing. The theory of quantum physics posits that matter, at a quantum level can be in a superposition of multiple classical states. And those many states interfere with each other like waves in a tide pool. The state of matter after a measurement "collapses" into one of the classical states.

Thereafter, repeating the same measurement will produce the same classical result. Quantum entanglement occurs when particles interact in ways such that the quantum state of each cannot be described independently of the others, even if the particles are physically far apart.

Quantum computing stores information in quantum states of matter and uses its quantum nature of superposition and entanglement to realize quantum operations that compute on that information, thereby harnessing and learning to program quantum interference.

Quantum computing might sound daunting, but with the right resources you can start building quantum applications today.

Figure 1: Microsoft document that explains quantum computing

Another example of explaining quantum computing
https://quantum.country/qcvc

1.2 Reference Documents

With the build of quantum computing and quantum algorithms, then we can go further with the discussion of virtual quantum computing.

Here is one tutorial of the introduction of virtual quantum computing machine.
http://swarm.cs.pub.ro/~agheorghiu/licenta/Quantum%20Computing%20Virtual%20Machine.pdf

We used IBM's Q Experience as an example of the virtual quantum computing machine. Here are some reference documents that have specific and detailed information of the Q Experience quantum computing machine.

https://quantum-computing.ibm.com/docs/ (IBM's Q Experience)

Learn Quantum Computation using Qiskit

Greetings from the Qiskit Community team! We initiated this open-source textbook in collaboration with IBM Research as a university quantum algorithms/computation course supplement based on Qiskit.

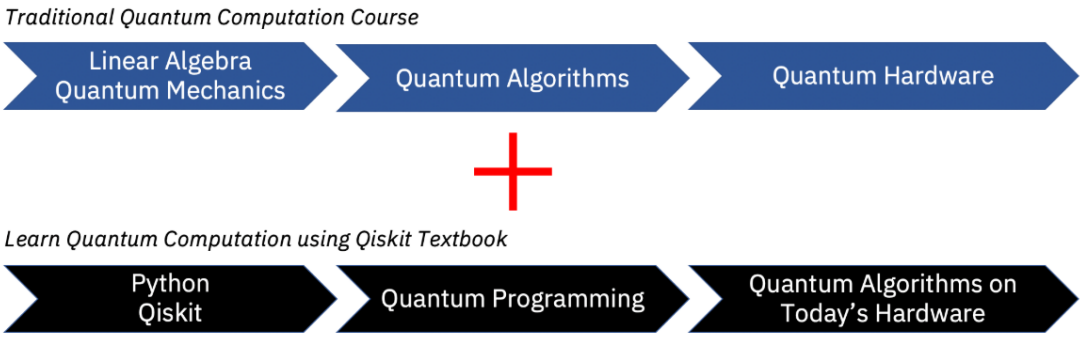

Figure 2: https://qiskit.org/textbook/preface.html (Quantum Computation using Qiskit)

Also, there are some other virtual quantum computing machines on other platforms. Such as MicrosoftQDK.

Get started with the Quantum Development Kit (QDK)

10/23/2019 • 4 minutes to read • +2

Welcome to the Microsoft Quantum Development Kit!
The QDK contains all the tools you'll need to build your own quantum programs and experiments with Q#, a programming language designed specifically for quantum application development.

To jump right in, you can head over to the QDK installation guide. You'll then be guided through installing the Quantum Development Kit on Windows, Linux, or MacOS machines so that you can write your own quantum programs.

If you don't feel quite ready to start coding, but want to learn more about quantum computing and Q#, we encourage you to still read this article to get a feel for the resources at your disposal. In the five questions about quantum computing section, you'll find links to precisely what you're looking for!

Get started programming

The Quantum Development Kit provides many ways to learn how to develop a quantum program with Q#. To get up and running with the power of quantum, you can try out our *quickstarts*:

- The quantum random number generator is a "Q# Hello World" style application, providing a brief introduction to quantum concepts while letting you build and run a quantum application in minutes.
- Quantum basics with Q# guides you on writing a Q# program that demonstrates some of the foundational concepts of quantum programming. If you are not ready to start coding, you can still follow along without installing the QDK and get an overview of the Q# programming language and the first concepts of quantum computing.
- Grover's search algorithm offers an example of a Q# program to get an idea of the power of Q# for expressing the quantum algorithm in a way that abstracts the low-level quantum operations. This quickstart guides you through developing the program in a variety programming environments (with a Python host or with .NET language host and with Visual Studio or Visual Studio Code).

Figure 3: https://docs.microsoft.com/en-us/quantum/welcome?view=qsharp-preview (Microsoft QDK)

As well as the basic operation of the virtual quantum computing machine, like how to install, how to create quantum circuit....
https://quantum-computing.ibm.com/docs/qis-tut/ (Qiskit tutorials)

2 Code documentation

Quantum programming is the process of assembling sequences of instructions, called quantum programs, that are capable of running on a quantum computer.

In quantum computing, a qubit or quantum bit (sometimes qbit) is the basic unit of quantum information. So it determines quantum computer is different from normal computer, it can contain not only value 0 or value 1, but also contain some values which are in superposition state.

2.1 Quantum Code

Quantum programming is based on Q# language, here is an example of Q language from Microsoft.

Symbols

The name of a symbol bound or assigned to a value of type `'T` is an expression of type `'T`. For instance, if the symbol `count` is bound to the integer value `5`, then `count` is an integer expression.

Numeric Expressions

Numeric expressions are expressions of type `Int`, `BigInt`, or `Double`. That is, they are either integer or floating-point numbers.

`Int` literals in Q# are identical to integer literals in C#, except that no trailing "l" or "L" is required (or allowed). Hexadecimal and binary integers are supported with a "0x" and "0b" prefix respectively.

`BigInt` literals in Q# are identical to big integer strings in .NET, with a trailing "l" or "L". Hexadecimal big integers are supported with a "0x" prefix. Thus, the following are all valid uses of `BigInt` literals:

Q# Copy

```
let bigZero = 0L;
let bigHex = 0x123456789abcdef123456789abcdefL;
let bigOne = bigZero + 1L;
```

`Double` literals in Q# are identical to double literals in C#, except that no trailing "d" or "D" is required (or allowed).

Given an array expression of any element type, an `Int` expression may be formed using the `Length` built-in function, with the array expression enclosed in parentheses, `(` and `)`. For instance, if `a` is bound to an array, then `Length(a)` is an integer expression. If `b` is an array of arrays of integers, `Int[][]`, then `Length(b)` is the number of sub-arrays in `b`, and `Length(b[1])` is the number of integers in the second sub-array in `b`.

Given two numeric expressions of the same type, the binary operators `+`, `-`, `*`, and `/` may be used to form a new numeric expression. The type of the new expression will be the same as the types of the constituent expressions.

Figure 4: https://docs.microsoft.com/en-us/quantum/language/expressions

Boolean Expressions

The two `Bool` literal values are `true` and `false`.

Given any two expressions of the same primitive type, the `==` and `!=` binary operators may be used to construct a `Bool` expression. The expression will be true if the two expressions are equal, and false if not.

Values of user-defined types may not be compared, only their unwrapped values can be compared. For example, using the "unwrap" operator `!` (explained in the Q# type model page),

Q# Copy

```
newtype WrappedInt = Int;     // Yes, this is a contrived example
let x = WrappedInt(1);
let y = WrappedInt(2);
let z = x! == y!;             // This will compile and yield z = false.
let t = x == y;               // This will cause a compiler error.
```

Equality comparison for `Qubit` values is identity equality; that is, whether the two expressions identify the same qubit. The state of the two qubits are not compared, accessed, measured, or modified by this comparison.

Equality comparison for `Double` values may be misleading due to rounding effects. For instance, `49.0 * (1.0/49.0) != 1.0`.

Given any two numeric expressions, the binary operators `>`, `<`, `>=`, and `<=` may be used to construct a new Boolean expression that is true if the first expression is respectively greater than, less than, greater than or equal to, or less than or equal to the second expression.

Given any two Boolean expressions, the `and` and `or` binary operators may be used to construct a new Boolean expression that is true if both of (resp. either or both of) the two expressions are true.

Given any Boolean expression, the `not` unary operator may be used to construct a new Boolean expression that is true if the constituent expression is false.

Figure 5: https://docs.microsoft.com/en-us/quantum/language/expressions

2.2 Quantum Algorithm

Just like modern computers, quantum computer also have to be based on quantum algorithm to solve problems. Here are some examples of quantum algorithm:

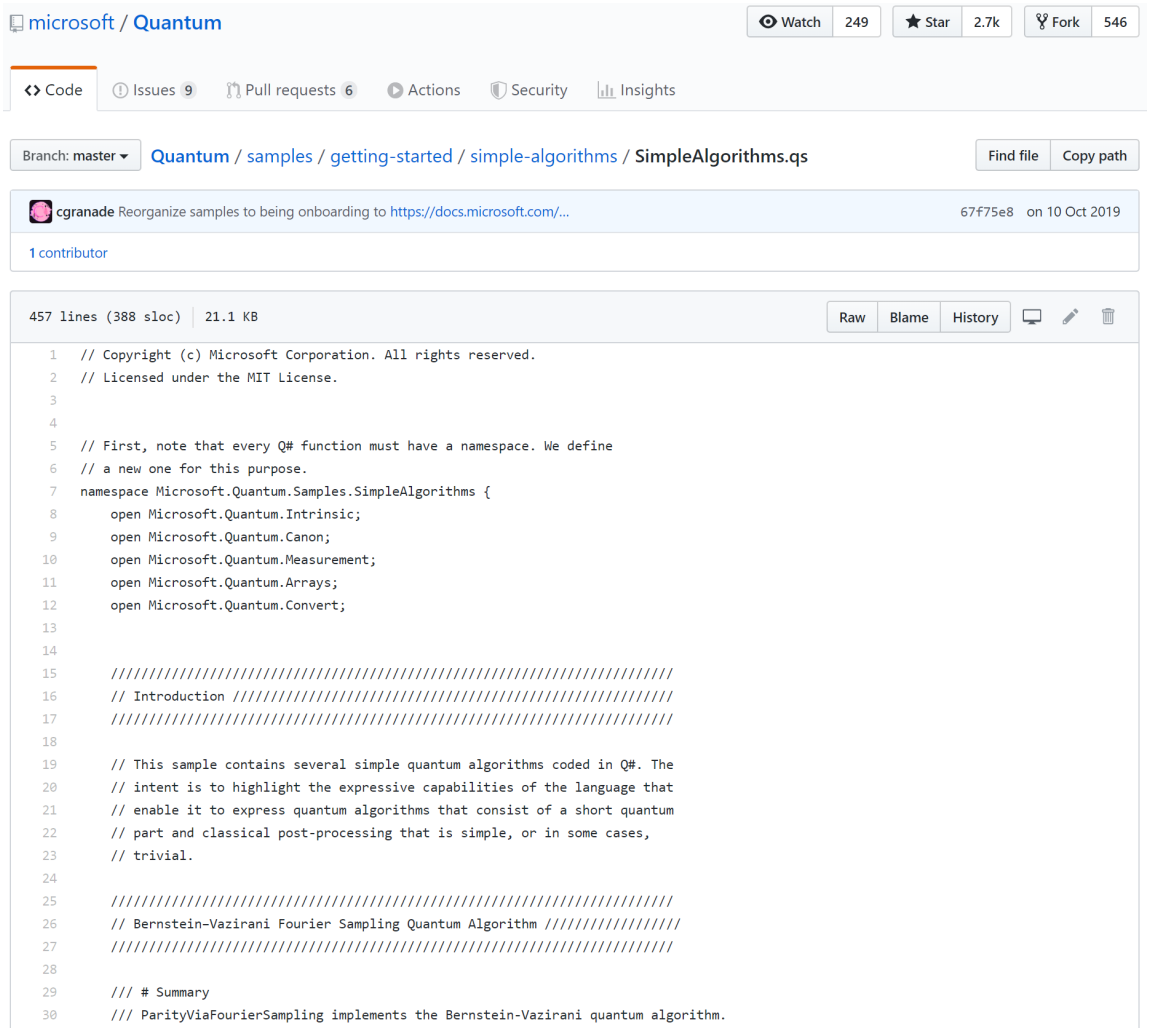

Figure 6: https://github.com/microsoft/Quantum/blob/master/samples/getting-started/simple-algorithms/SimpleAlgorithms.qs

And the classical quantum computing algorithm for integer factorization,Shor's algorithm that can factorize big integers in polynomial-time.

Proceed as follows:

1. Initialize the registers to

$$\frac{1}{\sqrt{Q}}\sum_{x=0}^{Q-1}|x\rangle = \left(\frac{1}{\sqrt{2}}\sum_{x_1=0}^{1}|x_1\rangle\right)\otimes\cdots\otimes\left(\frac{1}{\sqrt{2}}\sum_{x_q=0}^{1}|x_q\rangle\right).$$

   where $\otimes$ denotes the tensor product.

   This initial state is a superposition of Q states, and is easily obtained by generating q independent qubits, each a superposition of 0 and 1 states. We can accomplish this by initializing the qubits to the zero-position, and then applying the hadamard gate in parallel to each of the q qubits, where $2^q = Q$.

2. Construct $f(x)$ as a quantum function and apply it to the above state, to obtain

$$U_f|x,0^q\rangle = |x,f(x)\rangle$$

$$U_f\frac{1}{\sqrt{Q}}\sum_{x=0}^{Q-1}|x,0^q\rangle = \frac{1}{\sqrt{Q}}\sum_{x=0}^{Q-1}|x,f(x)\rangle$$

   This is still a superposition of Q states. However, the $q+n$ qubits, i.e, the q input qubits and n ($>\log_2(N)$) output qubits, are now entangled or not separable, as the state cannot be written as a tensor product of states of individual qubits.

3. Apply the inverse quantum Fourier transform to the input register. This transform (operating on a superposition of $Q = 2^q$ states) uses a Q-th root of unity such as $\omega = e^{\frac{2\pi i}{Q}}$ to distribute the amplitude of any given $|x\rangle$ state equally among all Q of the $|y\rangle$ states, and to do so in a different way for each different x. We thus obtain

Figure 7: https://en.wikipedia.org/wiki/Shor%27s_algorithm

2.3 Quantum circuits

In quantum computing there is a special programing which is quantum circuits.

A quantum circuit is a model for quantum computation in which a computation is a sequence of quantum gates, which are reversible transformations on a quantum mechanical analog of an n-bit register.(https://en.wikipedia.org/wiki/Quantum_circuit)

Here is an example of quantum circuits based on IBM circuits composer experience:

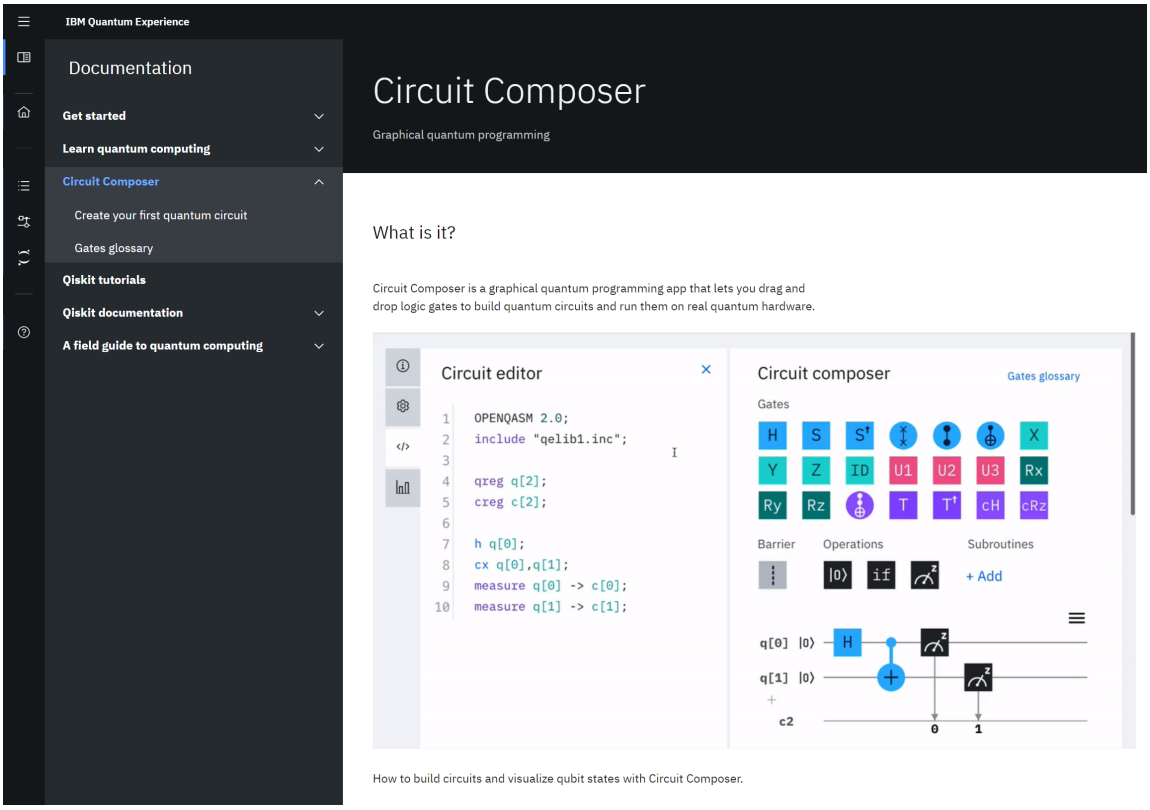

Figure 8: https://quantum-computing.ibm.com/docs/circ-comp/

3 Community documentation

Community documentation is an open source document. There are many forms and uses for the community documentation. One of the forms is QA. This form allows software developers to seek help from another software developer. Another form of community documentation is blog posts. This form is an open source documentation that is provided by the community to share their knowledge about the topic.

3.1 QA

As a new software developer who just started researching on quantum computing, reading the written documentation to understand the very basic terminology might be difficult. Thus, posting your question about quantum computing online can help you get a more understandable explanation to the problem you do not understand. Below is an example of QA form, explaining the basic of quantum computing.

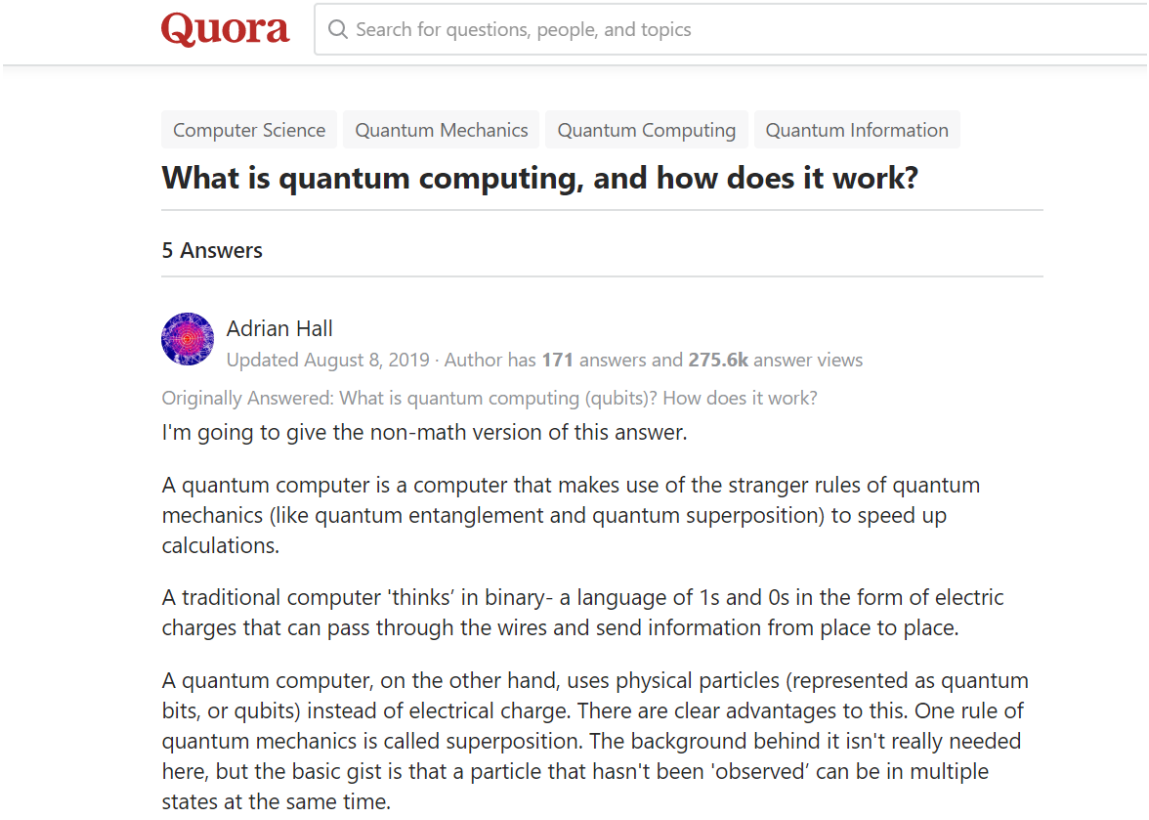

Figure 9: https://www.quora.com/What-is-quantum-computing-and-how-does-it-work

Therefore, to get the general idea of quantum computing as a coprocessor and not just a stand alone computer. We can simply post our question and get the explanation from a QA form. Below

is an example of an explanation received from a question about why quantum computing is not a stand alone computer.

Quora Search for questions, people, and topics

Scott Alexander, PhD in CS and have worked as a sysadmin, application programmer, and researcher

Answered Oct 10, 2017 · Author has **243** answers and **1m** answer views

Quantum "computers" (or more precisely coprocessors) can solve certain problems more quickly than classical computers. They do this by using some of the strange properties of quantum mechanics. (Famously, Einstein referred to entanglement, a prime element of quantum computing, as "spooky action at a distance." He said this because he didn't believe that the universe could work this way. There is evidence to show that he was wrong.)

So quantum coprocessors are not just fast classical computers. Instead, they use a different way of performing the computation to allow them to answer certain questions quickly. Two of the best known / most useful quantum algorithms are Shor's and Grover's.

Shor's algorithm factors numbers more quickly than any known classical algorithm. In particular, encryption algorithms like RSA rely on the idea that we cannot factor numbers quickly. If practical quantum computing becomes a thing, that will no longer be true and it will be trivial to decrypt information encrypted with such an algorithm. By contrast, using classical computers with the algorithms that we currently know, a faster classical computer would have to be ridiculously (perhaps impossibly) faster than current computers.

Grover's algorithm finds an element of an unsorted set of data in sublinear time. For the classical algorithm, of course, you end up looking at every piece of data until you find the one you want. On average, you have to look at half the data before finding what you want. Grover's finds the data without having to look at that much of the data. I can't even imagine how I would approach trying to do that classically.

Figure 10: https://www.quora.com/Is-a-quantum-computer-just-a-fast-computer

3.2 Blog post

Since quantum computing is still in a developing stage, being able to get immediate update information can be crucial for software developers. Thus, with the help of a blog post from a company that works on quantum computing such as IBM. Software developers can now easily get the latest information about quantum computing. Below is a screenshot of IBM blog post page for quantum computing

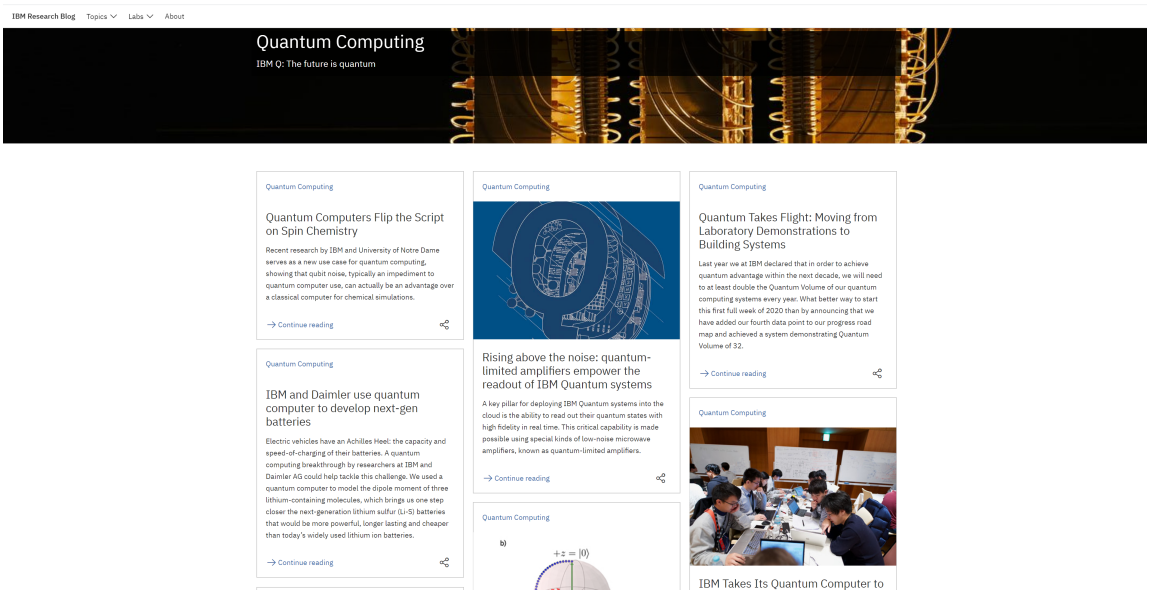

Figure 11: https://www.ibm.com/blogs/research/category/quantcomp/

4 Conclusion

In conclusion, there are quite a lot of advantages of quantum computer and quantum algorithms. Quantum computer uses qubits instead of bits compared with a normal computer. The normal bits can only store information as binary 0 and 1 states, while one qubit can store 2 states at the same time, which means if a quantum computer has 10 qubits, it can run 1024 states at the same time. That is equal to a normal computer use 1024 bits to run. If the quantum computer has large enough qubits like about 50, its efficiency is equal to thousands of normal computers run at the same time.

However, the quantum computer can not run without the quantum algorithm. Because like the normal computer, that we need to store the true and false value into register in order to perform calculation, qubits also need the quantum algorithm to run. For example the Shor's algorithm, it can use qubits to solve the discrete logarithm problem and the integer factorization problem in polynomial time. Also, the qubits use photons or some other particles as a physical medium and they need to have a very strict environment to keep their excited state.

Quantum computers can not run by itself. A reasonable quantum computer, should be implemented with a suitable quantum algorithm, and an environment that keeps photons or particles in excited state and so on. Therefore, it can be thought of as a co-processor.