Hello everyone, I’m Chongdan Pan, welcome to my final presentation, my topic is quantum parallelism and the Deutsch’s algorithm.

So what’s quantum parallelism? Let’s start from our familiar computer field.

As we all know, bit is the smallest unit in computer, it’s like a switch, can either be 0 or 1. For a cpu, it can only click the switch once in a clock, hence one cpu can only run one instruction at a time. If we want to run multiple instructions simultaneously, we need more computers work parallelly.

Things are different for quantum computer, qubit is used instead of classic bit, and it’s based on the superposition low in the quantum field. A qubit can either be represented by the different polarization direction of a photo, or an electron spin, as long as they’re orthogonal to each other. Hence, a qubit is like a rotary knob, has infinite states rather than 1 and 0.

For n classic bits, they can only represent one state such as 1-0, but for qubits, they can represent 2 power of n states, each state has its own probability.

Since n qubits can represent 2 power of n’s states, single cpu can solve 2 power of n problems parallelly

So how do we understand and describe qubit in practice?

A qubit is represent by a ket symbol, which is an abbreviation of column vector, each element represent the probability for it to be 1 and 0. Hence a classic bit 1 is 0-1 and 0 is 1-0. When it comes to a superposed qubit, it’s a-b, since the total probability should be one, its should be normalized.

The states of two qubits can also be represented by the multiplication of two independent qubits.

If we measure one of the two qubits, it will collapse into one qubit

To implement qubits, we need logic gates, just like classic computer.

A logic gate is represented a unitary matrix, which can perform transformation on the column vector. Unitary matrix is any matrix whose conjugate transpose U\_dagger is also its inverse. The right side show different gates represented by unitary matrix.

Hadmard Gate is commonly used gate, which can change certain qubit to uncertain states.

After the transformation, each qubit will have same probability to be one or zero. What if we input more qubits together?

Two qubits will have 4 states with same probability

Hence, for n inputs, we’ll have 2 power of n states with same probability. The hadmard gate is like shuffling all inputs.

Controlled NOT Gate is another gate, it usually cope with two inputs, and exchange the probability of two specific states. If we ignore the first qubit and focus on the second one, we will find it’s value depends on the initial value of input.

So the CNOT will perfume an operation like XOR gate on the second qubit, depend on the input value.

No let’s focus on the deutsch’s algorithm.

The input of deutsch problem is a function with input 0 or 1 and similar output. We need a machine called oracle to determine whether it’s constant or balanced. Constant means the function’s output is always the same, balanced means there is same probability for the function to output 1 or 0

In classic method, we need to evaluate the function twice by testing different input.

But through the deutsch algorithm in quantum computer, it only needs one evaluation. First, we input two classic bits 0 and 1 and implement Hamdard gate them. Then we use CNOT Gate to apply XOR operation on the second qubits based on the first qubit.

According to the property of XOR, we can do some simplification, and we only need focus on the first qubit.

After we take out the first qubit, we need to pass it through the H gate again, and get our final result.

According to the equation, we know the function is balance if the qubit is 0 and constant if not.

In this simple case, the Quantum Algorithm just save half time, but what if he problem is more complex?

In this case, the function will take n inputs together, and have same output.

Therefore, in classic solution, we need to evaluate at least more than half of all possible input to determine it’s constant or balance

But it still need once for Deutsch Algorithm. We just need to scale the 0 classic bits to n-1

And again we pass them through the H-gate, and apply f(x) and XOR function on the last 1 bit. Recall the input of f(x) should base on the value of first n-1 qubits.

In the same way, we will evaluate the output of first n-1 output. If they’re all 0, then it’s balance and constant otherwise. Since we already assure that the f(x) is either balance or constant, actually we only need to evaluate once.

Besides the algorithm, I want to talk about some frontiers about quantum parallelism.

Google has claimed that they have already realize quantum supremacy through a quantum CPU with 53 qubits.

Quantum supremacy means the quantum computer can solve a problem much more faster than the most advanced classic computer. And google succeeds in solving a problem which requires a classic computer for 10000years, and the quantum computer only solve it in 200 seconds.

A company called D-wave also has developed quantum computer with 512 Qubits. As I mentioned, adding one qubit to a quantum computer means it’s two times more powerful. So it’s very extraordinary.

However, there are many problems for Quantum Computer. First, the material for the computer is very fragile, and it’s hard for the computer to work in our home and office. Second, since quantum is about uncertainty, the result from the computer is not always true. Thirdly, the we’re in need of a lot of new algorithms like Deustch for quantum computer, for now, even though the computer is very powerful, they can only solve specific problems, and some problems even don’t need the feature of quantum