The Implementation of a QC

The idea is to give insights about what the Hardware and Logic Andrew and Pranav have been talking about in their respective part can accomplish

Implementation mean in Object Oriented Programming (OOP) that a class provide the methods specified in the interface. However, implementing is not specific to OOP and concerns more generally the realisation of algorithms; algorithms which once carefully programmed should output a result that fits a given problem.

To study the quantum implementation, it’s therefore important to work out the logic and algorithms that drive these machines and make them so powerful. We are used to multi-purpose computers that are easily reprogrammable, we need to go back to lower level of details to higher order programming language, this is the key in the development of QC and their implementation.

# The Quantum Algorithms

Algorithm exists from the dawn of computing. They are a sequence of steps that should lead to a logical, generally immutable, and meaningful outcome. We find computers better than humans to follow most algorithm as their capacity to calculate and treat data exceed us by far. However, even Gordon Moore (Moore, 1965) could not predict that we would go so small in the transistors that we would face the limits of the atom. As a consequence, the computing power on regular CPU is not likely to continue increasing as it has now. One of the flaw of those chips, especially in an era like ours where the Big Data is becoming so crucial, the analysis of such flow of information is going to be impossible. Another would be the analysis of multiple factor that impact a phenomenon requires the analysis of each one of them varying separately (weather predictions, deciphering communication, etc.). The algorithms for QC take advantage of the fact QC can explore multiple solutions at once and therefore more capable of tackling today’s computing limitation.

## Algorithms on amplitude amplification

“Quantum mechanics can speed up a range of search applications over unsorted data.”, (Groover, 1997). Groover studied in his paper the problem of accessing an unsorted database of *N* states. On average you hope to only need to scan half of the database before finding the required entry, but in the worst case, you will scan all *N* entries of the database (*O(N)*). The Quantum algorithm uses properties of the amplitude of qubits. The algorithm can be described as follow:

Only one item (*I*) in the database satisfies the property *P(I)=1*, for all the others states *S* we have *P(S)=0*.

Initialise the system so each state has the same amplitude (superposition ).

Apply the transformation and if a state reaches *P(S)* =1, rotate the phase by 180 degrees.

Note that the transformation affects the amplitude of the states in such a way the desired one will stand out. The complexity of the loop and of the initialisation makes it a , which is a gain of polynomial order (divided by two)compared to the classic algorithm.

## Fourier’s transform

The Discrete Fourier’s transform (DFT) represents the frequency domain of the given input. It can be applied to quantum algorithms and implemented very efficiently, particularly useful for finding eigenvalue of unitary operators and probably its most famous application is the use in Shor’s algorithm.

### Shor’s Algorithm

(Shor, 1997)

Shor’s algorithm has become very famous over the years as its realisation, if it should come, will affect radically the domain of communication and cryptography that protects everyone’s privacy. Decrypting an RSA transaction to a bank to retrieve the credit cards info could be done in polynomial time –in terms of log n- instead of exponential time leaving many systems at risk.

The factorisation algorithm of a number *N* consists like many Quantum algorithms of a classic routine run on a regular computer, that routine triggers at some point a quantum routine, but the process is originally controlled by a regular computer. The main advantage of this method is that it allows to save resources on both machines as they are used to what they are the most efficient for. The Quantum routine is very specialised and only serves as finding the order of an element; its efficiency results in the ability to search a remaining square root of different from 1 and -1.

We are sure it exists, as stated by the Chinese remainder theorem. Finding such a number and taking then will be a proper factor of *N* (different from 0 and N).

Indeed is proper because:

* If , then by Bézout’s theorem,

Leading by a multiplication by on both sides that which contradicts how we obtained (S is not congruent to -1 mod N)

* If which also contradicts how we obtained S (S is not congruent to 1 mod N).
* Therefore we only have and d is a proper factor for N

The algorithm to find such an S is as follows:

1. Pick a random number
2. If return a
3. Else find the order of in the group and if or goto step 1
4. Else we’ve found our S!

Researchers have for now mostly limited themselves to factorising the number 15 (probably the smallest non-trivial test for Shor’s algorithm), probably the 256-bits keys of the Advanced Encryption Standard. But it’s probably one of the biggest success for QC which demonstrate their capacity on one the most complex Quantum algorithm that exists until these days. If Quantum Computer comes out, smaller keys may even be factorable from personal machines.

# The needs of a general-purpose programming language

For a computer to be truly useful and game changer, it is important to be able to program it. From the first “Hello, Quantum World”, to Shor’s algorithm, programmers need to be able to express their ideas to the machine. Powerful and diverse programming languages could inspire generation of programmers to accomplish great things with their ‘PQ’ (Personal Quomputer?).

## A Pseudo-code

The first approach to programming may be a precise pseudo-code. It should be clear and intuitive but have a well-defined shared logic that don’t leave room for interpretation.

The Convention for a Quantum Pseudocode (Knill) written in 1996 stands as a monolith in the field. It reuses classic and expected patterns from classic computing (like if evaluates the ket 1 as true, or the for loops behaves as expected).

It also achieves the simplification of recurrent operations like the Hadamard operation on a Qubit a or the phase shift using letter and classic function application like *H(a)* for the application of the Hadamard matrix on the qubit *a*.

Measurements use simple inverted arrows like *a <-- a* and function are just name.

It is therefore possible to write down pseudo code without a powerful text editor.

## Programming Languages

Now that code can be expressed formally, programming languages can have precise logical specification. As it’s possible to expect, there exists two types of Quantum Programming Languages (QPL), imperative and functional. There exists already plenty of different QPL even though the need for the moment would be the actual quantum computer. Many QPLs are based from pre-existing programming languages.

For example Q is an extension of C++, it can therefore use the multiple libraries and makes it easy to reuse chunks of code that nay already exist today. Also, there is no need to start a compiler from scratch, unlike some trials that existed before like Omer’s QCL. However they happen to share some features, the most interesting one could be the representation of the data. The language expect a Quantum heap on to which the *Quantum registers* would hold a reference to the different qubit that would to stored on it.

Haskell, so loved at Imperial College, also happen to be Quantum ready! There is a Haskell package called QIO that draw the bases of what an implementation of QHaskell would look like. For the most interested, follow this link for an implementation of Shor’s algorithm using QIO package given in the example of the package. More recently, The QPL Quipper developed in the recent years and provides all the tools to expect from a QPL but wrapped up in Haskell’s solid type system and suitable for monadic instructions, making it a very powerful tool for tomorrow’s programmers.