# Derivation of Quantum Guarded Command Language Program for Average

Cherry Viona Telap[1], Benny Pinontoan[2], Julia Titaley[3]

[1] **Program Studi Matematika, FMIPA, UNSRAT Manado, cherrytelap@yahoo.com**
[2] **Program Studi Matematika, FMIPA, UNSRAT Manado, bpinonto@yahoo.com**
[3] **Program Studi Matematika, FMIPA, UNSRAT Manado, july_titaley@yahoo.com**

**Abstract**

Has conducted research to determine the derivation of quantum guarded command language (qGCL) program for average. Initially calculation of average value was made in guaded command language (GCL) which is then implemented on a digital computer into the Pascal programming language. Furthermore GCL to calculate the average value was analyzed again to be made in the quantum guarded command language (qGCL). qGCL implementation is on a quantum computer is a future computer could perform calculations very quickly because it uses a superposition state is referred to as quantum bits (qubits).

**Keywords :** GCL, qGCL, Quantum Computer

# *Derivation of Quantum Guarded Command Language Program for Average*

***Abstrak***

*Telah dilakukan penelitian untuk mengetahui derivation of quantum guarded command language (qGCL) program for average. Awalnya perhitungan nilai rata – rata dibuat dalam guaded command language (GCL) yang kemudian diimplementasikan pada computer digital menjadi bahasa pemrograman pascal. Selanjutnya GCL untuk menghitung nilai rata – rata tersebut dianalisa lagi untuk dibuat dalam quantum guarded command language (qGCL). Pengimplementasian qGCL adalah pada komputer kuantum yang merupakan komputer masa depan yang dapat melakukan perhitungan dengan sangat cepat karena memanfaatkan keadaan superposisi yang disebut sebagai quantum bit (qubit).*

***Kata kunci** : GCL, qGCL, Komputer Kuantum*

## 1. Introduction

Mathematics and information technology system have a lot of relationships. The development of information technology is also not far from the role of mathematics, for example, with the development of technology in space there are very advanced, due to advances in the field of physics in which advances in the field of physics will not be achieved without mathematics. Mathematics is science base to train ability to think logically and systematically, therefore mathematics teaching for critical thinking how that technology continues evolve in line with the development of mathematics. Civilization development of mathematics has been widely issued thoughts and ideas towards the implementation of modern equipment, such as computers and communication systems. Processing numbers in mathematics forming a programming formula that is used in the development of computers science is the science which contains the theory, methodology, design and implementation, related to computing, computer, and the algorithm in the perspective of software or hardware. That is why the computer technology more days to experience growth and progress. But apparently the sophistication of existing computer still can not satisfy human desires this dream to make a supercomputer with super speed. Computers that will be appropriately referred to as Super Computer is the Quantum Computer. The theory of quantum

computers was first coined by physicists at the Argonne National Laboratory about 20 years ago that Paul Benioff. Paul Benioff is the first to apply the theory of quantum physics to the world of computers in 1981 [1]. A quantum computer will transform the information technology and human lifestyle in the future because the computer utilizes atoms and molecules as an information carrier. Atom is a quantum object that works based on quantum bits or known as qubits. In addition, a quantum computer can perform mathematical calculations in a matter of minutes in which the calculation usually takes months, years, even centuries. Therefore, in this study the author are interested in doing research on quantum computing which in this case I will solve the derivation of Quantum Guarded-Command Language program for average.

## 2. Qubit vs Bit

Digital computers are computer that we use everyday. Digital computers are very different from the quantum computer as a digital computer works with the help of a microprocessor which is usually known as the Central Processing Unit (CPU). The microprocessor is "computer's heart" in the form of small chips are composed of many transistors. Digital computer system memory using binary system (0 and 1), known as Binary Digit (BIT). Conversion of decimal numbers used are as follows: 0 = 0 ; 1 = 1 ; 2 = 10 ; 3 = 11 ; 4 = 100 ; 5 = 101 ; 6 = 110 ; 7 = 111 ; 8 = 1000 ; 9 = 1001 ; 10 = 1010 ; 11 = 1011 ; 12 = 1100 ; 13 = 1101 ; 14 = 1110 ; 15 = 1111 ; 16 = 10000 ; 17 = 10001 ; etc. If you want to calculate what the numbers represented by 101101 the following way (using system $2^n$) :

$(1 \times 2^5) + (0 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) = 32 + 0 + 8 + 4 + 0 + 1 = 45.$

A sample calculation using the mathematical sum of the binary system :

| | | | |
|---|---|---|---|
| 25 | | | 11001 |
| 17 | + | ⟶ | 10001 + |
| 42 | | | 101010 |

This system which we use when we process information using a digital computer.

Quantum Computer utilizing the phenomenon of 'strange' that is referred to as the superposition of quantum mechanics, the superposition of a particle can be in two states at once. In a quantum computer, in addition to 0 and 1 also known as a superposition of both. This means that the state can be 0 and 1, not only 0 or 1 as in ordinary digital computers. Quantum computers do not use BITS but QUBITS (Quantum Bits). Because of its ability to be in a variety of circumstances (multiple state), quantum computers have the potential to perform a variety of calculations simultaneously so much faster than digital computers. Quantum computer using a particle can be in two states at once, for example atoms at the same time in a state of excited and unexcited, or photons (particles of light) which are in two different places at the same time. That is, the atoms have a spin configuration. Atomic spin could be *up,* also could be *down.* For example, when the atomic spin is pointing up we give the symbol 1, while the spin down is 0 (like binary system in a digital computer) [1].

This quantum circumstances can be modeled mathematically in a linear combination that is $|\Psi\rangle=\psi_1|\uparrow\rangle + \psi_2|\downarrow\rangle$ where $\psi_1$ and $\psi_2$ is the probability amplitude or wave function of each spin state and in general complex-valued. The model is derived from statistical theory the chances of one of the two events between A and B is $p(A \vee B) = p(A) + p(B)$. The atoms that are in a state of superposition means still have $spin\ up$ and $spin\ down$ simultaneously until measurement. Atomic force measurement action to 'choose' one of the two possibilities. This means that after we do the measurement, the atoms no longer be in a superposition state. Atoms are already experiencing the measurement has a fixed spin : $up$ or $down$. When this concept is applied in a quantum computer, a superposition state occurs during the calculation process is underway. Systems on a quantum computer calculation is different from the digital computer. Digital computers perform calculations in a linear manner, while the quantum computer do all the calculations simultaneously (because there are multiple states all calculations can take place simultaneously in all states). This means there are many possible results of calculations, and to know the result of the calculations we have to measure the QUBIT. This QUBIT measurement action to stop the process of calculation and force the system to 'choose' one of all possible

answers. With this calculation system parallelism, we can imagine how fast quantum computers [2].

Currently, the development of technology has resulted in up to 7-qubit quantum computer, but according to the research and analysis that is, in the next few years a quantum computer technology can reach 100-qubit. That is, how quickly the computer a future time which all the calculations that usually takes months, years, even centuries in the and can be implemented in a matter of minutes if we use a quantum computer. Quantum computers no longer requires a computer chip are becoming increasingly congested as more and doubling the number of transistors needed to improve the performance of the computer, but the computer will be met by organic liquids as it's heart. This organic liquid containing atoms / particles can be in a superposition state. This means, we really take advantage of natural organic substances to be a sophisticated calculator because apparently of natural organic liquids have the talent to count [1].

## 3. Quantum Computation

Consider the following two examples binary: Binary 011 and 111. The first is 3 and Second binary is 7. In general, the three-digit number written by $2^3$ = 8 in different configurations that represent an integer of 0 to 7 . However, the three-digit numbers stored in only capable of storing one digit at a time situation. QUBIT in quantum system written boolean with numbers 0 and 1 are represented by a determination of normal and orthogonal quantum mutual expressed by $\{|0\rangle,|1\rangle\}$ [7]. Both forms are forming a computational basis and the other is written as a superposition $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix}\alpha\\ \beta\end{pmatrix}$ where $\alpha$ and $\beta$ in case it is $|\alpha|^2 + |\beta|^2 = 1$ by Euler's Formula, can be rewritten as $|\psi\rangle = e^{i\gamma}\left(cos\frac{\theta}{2}|0\rangle + e^{i\varphi}sin\frac{\theta}{2}|1\rangle\right)$ [3].

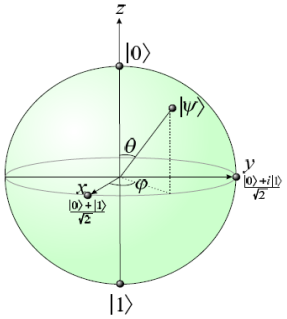


Figure 1. The Bloch Sphere: a geometric representation of a qubit

QUBIT is typical microscopic systems, for example : atomic, nuclear spin and photon polarization. Set of n QUBIT is called a quantum register of size *n*. We assume that the information is stored in form binary registers. For example, the number 6 is represented by $|1\rangle \otimes |1\rangle \otimes |0\rangle$. In the form of a neat notation, $|a\rangle$ is the tensor product $|\alpha_{n-1}\rangle \otimes |\alpha_{n-2}\rangle \ldots |\alpha_1\rangle \otimes |\alpha_0\rangle$ where $\alpha_i \in \{0,1\}$ and represents a quantum registers with the value of $\alpha = 2^0\alpha_0 + 2^1\alpha_1 + \ldots 2^{n-1}\,\alpha_{n-1}$. There are $2^n$ kinds of circumstances, which represents all of the binary length *n* or numbers from 0 to $2^{n-1}$, and all of which form the basis of good computational. In the following example $\alpha_i \in \{0,1\}^n$ ($a$ is a binary string with length *n*) states that $|a\rangle$ included in the computational basis. So quantum register number 3 can store the number 3 or 7 separate, $|0\rangle \otimes |1\rangle \otimes |1\rangle \equiv |011\rangle \equiv |3\rangle$ and $|1\rangle \otimes |1\rangle \otimes |1\rangle \equiv |111\rangle \equiv |7\rangle$. However, it could also save both. What if we take the first qubit, and rather than put it into the $|0\rangle$ or $|1\rangle$, we prepare a superposition $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, then obtained : $\frac{1}{\sqrt{2}}[(|0\rangle + |1\rangle) \otimes |1\rangle \otimes |1\rangle)] \equiv \frac{1}{\sqrt{2}}(|011\rangle + |111\rangle) \equiv \frac{1}{\sqrt{2}}(|3\rangle + |7\rangle)$. In fact, we can prepare these registers into a superposition with into a superposition with all eight figures is enough to put each qubit in a superposition $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \ni \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, which can also be written into the binary follow: (normalization constant $2^{-3/2}$ ignored), $|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + |110\rangle + |111\rangle$. Or in decimal notation such as: $|0\rangle + |1\rangle + |2\rangle + |3\rangle + |4\rangle + |5\rangle + |6\rangle + |7\rangle$. Or as simple as : $\sum_{x=0}^{7}|x\rangle$ [4].

## 4. Guarded Command Language

According Dijkstra [5] Guarded-Command Language program is a sequence of assignment and skip manipulated by the standard constructors of sequential composition, Conditional selection, repetition and nondeterministic choice . Assignment is in the form $x := e$, where $x$ is a vector of program variables and $e$ is a vector of expressions whose evaluations always terminate with a single value. Guarded-Command Language as follows:

```
<guarded command> ::= <guard> → <guard list>
<guard> ::= <Boolean expression>
<guarded list> ::= <statement> {; <statement> }
<guarded command set> ::= <guarded command> {; <guarded command> }
<alternative construct> ::= if <guarded command set> fi
<repetitive construct> ::= do <guarded command set> od
<statement> ::= <alternative construct> | <repetitive construct> |
"other statement"
```

## 5. Quantum Guarded-Command Language (qGCL)

A quantum program is a pGCL program invoking quantum procedures and the resulting language is called qGCL [6]. Quantum procedures be of three kinds: *Initialization* (or state preparation) followed by *Evolution* and finally by *Finalisation* (or observation). Defined the type $\mathbb{B} \mathrel{\widehat{=}} \{0,1\}$, which will treat as Booleans or bits, depending on convenience. A classical register of size $n{:}\mathbb{N}$ is a vector of $n$ Booleans. The type of all registers of size $n$ is then defined to be the set of Boolean-valued functions on $\{0,1,\dots,n-1\} : \mathbb{B}^n \mathrel{\widehat{=}} \{0,1,\dots,n-1\} \to \mathbb{B}$. The quantum analogue of $\mathbb{B}^n$ is given by transform $q$ and it is the set of complex-valued functions on $\mathbb{B}^n$ whose squared modulus sum to 1 : $q(\mathbb{B}^n) \mathrel{\widehat{=}} \{\mathcal{X}: \mathbb{B}^n \to \mathbb{C} \mid \{\sum_{\mathcal{X}:\mathbb{B}^n} |\mathcal{X}(x)|^2 = 1\}$. An element of $q(\mathbb{B})$ is called a *qubit (quantum bit)* and that of $q(\mathbb{B}^n)$ a *qureg (quantum register)*. Classical state is embedded in its quantum analogue by the Dirac delta function : $\delta: \mathbb{B}^n \to q(\mathbb{B}^n)$, $\delta_x(y) \mathrel{\widehat{=}} (y = x)$. The range of $\delta$, $\{\delta_x | x: \mathbb{B}^n\}$ forms a *basis* (called the *standard basis*) for quantum states, that is : $\forall \mathcal{X} : q(\mathbb{B}^n) \bullet \mathcal{X} = \sum_{x:\mathbb{B}^n} \mathcal{X}(x)\delta_x$.

*Initialization* is a procedure which simply assigns to its qureg state the uniform square-convex combination of all standard states $\forall \mathcal{X} : q(\mathbb{B}^n) \bullet \mathbf{In}\ (\mathcal{X}) \mathrel{\widehat{=}} (\mathcal{X} := \frac{1}{\sqrt{2^n}} \sum_{x:\mathbb{B}^n} \delta_x)$.

*Evolution* models the evolution of quantum systems and consists of iteration of unitary transformations on quantum state. A unitary operator U is invertible and preserves scalar products or, equivalently : $\forall \mathcal{X}, \psi : q(\mathbb{B}^n) \bullet \langle U(\mathcal{X}), U(\psi)\rangle = \langle \mathcal{X}, \psi\rangle$. In qGCL evolution is modeled via assignment: for example, $\mathcal{X} = U(\mathcal{X})$ models the evolution of qureg $\mathcal{X}$ by means of unitary transform $U$. The content of a qureg can be read (measured) through quantum procedure *Finalisation* and suitable *observables.* Let $\mathcal{O}$ be an observable defined by the family of pair-wise orthogonal subspaces $\{S_j | 0 \le j < m\}$. In our notation we write $\mathbf{Fin}[\mathcal{O}]\ (i, \mathcal{X})$ for the measurement of $\mathcal{O}$ on a quantum system described by state $\mathcal{X} : q(\mathbb{B}^n)$. After the measurement, variable $i$ stores the index of the subspace to which the state is reduced and $\mathcal{X}$ stores that state.

*Finalisation* is entirely defined using the probabilistic combinator of pGCL : $\mathbf{Fin}[\mathcal{O}]\ (i, \mathcal{X}) \mathrel{\widehat{=}} \oplus \left[\left(i, \mathcal{X} := j, \frac{P_{S_j(\mathcal{X})}}{\left\|P_{S_j(\mathcal{X})}\right\|}\right) @ \langle \mathcal{X}, P_{S_j}(\mathcal{X})\rangle \mid 0 \le j < m\right]$ where $P_{S_j}$ is the projector onto subspace $S_j$. In the case of the one-dimensional space $\mathbb{C}\psi \mathrel{\widehat{=}} \{\alpha\psi \bullet \alpha : \mathbb{C}\}$ spanned by $\psi : q(\mathbb{B}^n)$ the projector is defined by: $\forall \mathcal{X} : q(\mathbb{B}^n) \bullet P_\psi(\mathcal{X}) \mathrel{\widehat{=}} \langle \psi, \mathcal{X}\rangle\psi$. We also note that if $\left\|P_{S_j}(\mathcal{X})\right\| = 0$ then $\langle \mathcal{X}, P_{S_j}(\mathcal{X})\rangle = 0$ and therefore by laws *P*-1 and *P*-2 the definition holds in that case, too. The definition of **Fin** remains also valid when an observable $\mathcal{O}$ is defined by a self-adjoin operator $\mathcal{O}$. In that case the projector for the $j$-th eigen-space of $\mathcal{O}$ is written $P_0^j$ [7].

## 6. Valid Quantum Guarded-Command Language Program

Quantum Guarded-Command Language (qGCL) have the formal syntax as follows [8] :

```
          <qprogram> ::= {<qproc declaration> | <proc declaration>;}
                         <qstatement> {;<qstatement>}
       <qstatement> ::= χ ::= <unitary op>(χ) |
                         Fin(<identifier>,[<identifier> | <identifier,
                         <identifier>]) |
                         In(<identifier>) | <statement>
                  χ ::= <identifier>
<qproc declaration> ::= qproc <identifier> (<formal parameter list>)
                         ≙ <qproc body>
        <qproc body> ::= var • <qproc statement> {;<qproc statement>} rav
  <qproc statement> ::= skip | x := e | <qloop> |
                         <qconditional>
            <qloop> ::= while <cond> do <qproc statement> od
     <qconditional> ::= <qproc statement> ◁ <cond> ▷ <qproc statement>
```

where <unitary op>(χ) is just some mathematical expression involving qureg χ; such expression should of course denote a unitary operator. As we see from the syntax, qGCL extends pGCL with the following :

- **qproc**'s are used to run standard GCL code on the quantum computer;
- to model quantum evolution, assignments outside a **qproc** may take the form :
  χ := U(χ)
  where χ is a qureg and *U* an unitary transformation over quregs. Assignments inside a **qproc** can only be standard;

outside **qproc**'s we may also use *Finalisation* and *Initialisation* on quregs, and of course calls to **qproc**'s.

**Definition** Let Z be the (standard/quantum) procedure defined as :
**proc/qproc** Z(**value** *p1*, **result** *p2*, **value result** *p3*) ≙ Zbody

Consider the following **qproc** with involve the definition :

**qproc** Dummy (**value** $a: \mathbb{B}^n$) ≙ DummyBody

a call to *Dummy* would then be :

**var** χ:q($\mathbb{B}^n$) •
  Dummy(χ)
**rav.**

## 7. Research Method

The steps study conducted among others, as follows :

1. Analysis of Guarded-Command Language to perform the calculations the average value
2. Perform verification on the guarded command language to determine the level of truth calculations with step-by-step derivation of the program.
3. Implement guarded command language to calculate the average value into the existing programming languages on a digital computer. And in this study, the author take Pascal programming language because the program is the most basic programming language.
4. Analysis to obtain a form of Quantum Guarded Command Language but before GCL modified first shape to form pGCL the remainder of the form pGCL made into the form of qGCL.
5. Compare the two programming languages, namely Guarded-Command Language with Quantum Guarded-Command Language in calculating the average value.

## 8. Results and Discussion

### 8.1. Natural Calculation About Average

In this section will be calculated the average value of N pieces of integer data is read from the input device. The average value is the sum of all values divided by the number of values. For example, N = 5 and the number of data are read row is 12, 10, 6, 2, 4, then the average value is : $\frac{(12+10+6+2+4)}{5} = \frac{34}{5}$ = 6.8 or in other words the average value formulated with $\frac{\sum_{i=0}^{n} i}{n}$ where *i* is data and *n* is the number of data.

Calculation of the average value of these when poured in the program will include programming languages that follow the algorithm so that a computer program can run these calculations. Digital computer programs that we use and we know a programming language to perform these calculations include the programming language Pascal, Delphi, C ++, etc.

In this study the author will perform the analysis of the calculation of the average value in the guarded command language that was first coined by Djikstra [2.3]. Guarded command language is the beginning of all programming languages on a digital computer, because the algorithm is a computer programming language on digital formed guarded-command of the language after it is run by a computer program. However, after analyzing the guarded command language does not answer the purpose of the author's study because the author wants the calculation of the average value instead of the digital computer, but in a quantum computer. So in this study, the author will make a quantum guarded command language (qGCL) with a syntax that has been made by the Zuliani [2.5] to perform the calculation of the average value. Quantum guarded command of the language is the beginning of a quantum computer programming language in which we know that quantum computer is a future computer could perform calculations very quickly.

### 8.2. Calculation Using Guarded Command Language

The first step is to design a programming language to calculate the average value of using a Guarded-Command Language.

By using the repetition rule, the draft guarded command language in the calculation of the average value are as follows :

```
|[
k, N, sum : Int;
average : real;
sum:=0;
k:=1;
do k ≤ N ∧ N > 0 → |[ sum:=sum+k; k:=k+1 ]|
od
average := sum/N
]|
```

#### 8.2.1. Derivation of Program

After designing guarded command language to perform the calculations the average value, then in this section will be the origin of the calculation of the program as a proof of the truth of the language of the program before it is implemented on a computer program. Calculation is as follows:

- ***Pre condition*** : $N > 0$
- ***Post condition*** : $R : S = (\underline{S}i: 0 \leq i \leq N : i)$
- ***Invariant*** : $P : S = (\sum i: 0 \leq i \leq k : i)$
- ***Ending*** : $k=N \wedge S = (\sum i: 0 \leq i \leq k : i) \rightarrow R$
- ***Initialization*** : $k:=0$

  $P_0^k : S = (\sum i: 0 \leq i \leq 0 : i) \rightarrow S = 0$ {singleton domain}

- ***Step*** **:** k:=k+1
  $P^{k}_{k+1}$ : S = (∑i: 0≤ i ≤ k+1 :i) →
  S = (∑i: 0 ≤ i ≤ k :i) + (k+1) → S
  +(k+1)

Thus obtained is the way of calculation as follows :

```
S:=0;
k:=0;
do k ≠ N →
      S:=S+(k+1)
      k:=k+1
od
average := S/N
```

### 8.2.2. Implementation of Program (Calculation Using Pascal Programing)

Furthermore, in this section will be the implementation of a guarded command language to calculate the average value of which has been designed in the previous section. The implementation form programming languages that will be executed by a computer program on a digital computer. Language programming on a digital computer in question, among others, which we know is a program Pascal, Delphi, C ++, etc. Which in this case, the author take the Pascal programming language as an example of a guarded command language implementations where Pascal programming language is a programming language is the most basic, simple, and that is not commonly known. Pascal program to calculate the average value is as follows :

```
Program average_calculation;
var
k, N, sum : integer;
average : real;
begin
write('Enter the number of terms the series : ');
readln(N); {N > 0}
sum := 0;
k := 0;
while k <= N do
begin
sum := sum + k;
k := k + 1;
end;
average := sum/N;
writeln(average);
end.
```

## 8.3. Calculation Using Quantum Guarded Command Language

This section will answer the author's study when the author will design a guarded command language in calculating average values using the rules of quantum computing or called quantum guarded command language (qGCL). qGCL to perform the calculation of the average value is as follows by using rules **qloop** and **qproc** as defined in the definition 2.6.1, but prior to qGCL first be designed pGCL:

```
proc average (value k:𝔹^k ; result χ: 𝔹^n) ≙
var ●
χ:=0;
k:=0;
while k ≤ N do
χ,k:= χ+k,k+1
od
average:= χ/N
rav
```

Next will be designed qGCL because as explained in 2.5 that a quantum program is a pGCL (Probabilistic Guarded-Command Language) program procedures and the quantum invoking the resulting language is called qGCL. pGCL denotes the Guarded-Command Language extended with the binary constructor $p\oplus$ for $p$:[0,1], in order to deal with probabilism.

The simple assignments constituting the body of average are readily compiled by means of the quantum primitives for mathematical-logic operations. In our case we need a sum operator Sum(χ,ψ), and division operator Div(χ,ψ). Initialization $\chi$ := 0 and k:=1 is readily modeled via the Dirac $\delta$ map. So quantum guarded command language to calculate the average value is as follows :

```
qproc qaverage (value result k:δ(𝔹^k) ; value result χ: δ(𝔹^n)) ≙
var ●
χ:= δ_0
k:= δ_0
while (k ≤ N) do
χ,k := Sum(χ,k), Sum(k)
od
qaverage:= Div(χ,N)
rav
```

Therefore when we look at the programming language qGCL was very clear that by combining quantum computation logic with guarded command language, then the program can perform calculations quickly because the qGCL utilize Dirac $\delta$ where on quantum programming Dirac $\delta$ is acting as a superposition. Because the Dirac $\delta$ act as a superposition then it is clear that in any programming language qGCL has qubit. Superposition is a state in mathematics is described in a linear combination $|\Psi\rangle = \psi_1|\uparrow\rangle + \psi_2|\downarrow\rangle$ with the provision of $|\psi_1|^2 + |\psi_2|^2 = 1$. By utilizing this superposition state, it is clear that all mathematical calculations in this study the average value calculation if using quantum guarded command language so the program can perform calculations quickly.

### 8.4. Comparison

After seeing the design of the programming language of GCL and qGCL to calculate the average value, then this section will discuss the comparison.

Compared with Quantum Guarded - Command Language (qGCL) in the calculation of the average value, it appears that Guarded Command-Language (GCL) is a simple programming language. GCL and qGCL is the beginning of a programming language on a computer that has guaranteed the truth, in this case the GCL on digital computers and qGCL on quantum computers where the previous chapter has studied the differences between digital computers and quantum computers. Implementation of GCL one of which is the Pascal programming while qGCL can not be implemented because until now there has been no quantum computer. Although qGCL yet to be implemented and visible language complicated but from qGCL can be seen how a program can calculate very quickly. We need not doubt the truth of qGCL level, because just like with GCL that the program can be guaranteed truth when implemented.

Pascal is an example of implementation of GCL seen that the language is quite different from the origin is GCL but taken his arithmetic logic. Neither is the case with qGCL, which must

programming language on quantum computer as the implementation of qGCL would be quite different form with origins that qGCL. Although different, but his arithmetic logic was to be taken so that the design qGCL then we can imagine a quantum computer arithmetic logic of the language in the future.

## 9. Conclusion

From this study, it could be concluded that Quantum-Guarded Command Language (qGCL) is a programming language designed to be implemented on a quantum computer as well as the implementation GCL is Pascal program on a digital computer. This study is an example of a shadow of a language program that when executed or executed by a computer program on a quantum computer, can perform calculations very quickly that in this study the author take the average value calculation.

Quantum computers can calculate quickly because it uses a system of quantum bits (qubits) in contrast to a digital computer which uses only bits. Qubit means to exploit superposition state that is a situation that can be in the state 0 or 1, or both at once (0 and 1).

To get a form of programming language qGCL in calculating the average value requires strong analysis of the GCL and quantum computing before qGCL shaped, the program analyzed prior to form pGCL. Quantum logic coupled with GCL produce qGCL that utilize Dirac δ in the language. The Dirac δ act as a superposition indicate there's qubit in the program.

## 10. References

[1] Surya, Y. 2014. Komputer Kuantum Komputer Masa Depan. www.yohanessurya.com/download/penulis/Bermimpi_07.pdf [Desember 2014].

[2] Hardhienata, H. 2014. Tutorial Mekanika Kuantum. Theoretical Physics Division. Bogor Agricultural University, Jl. Meranti S, Darmaga, Indonesia.

[3] Grattage, J.J. 2006. A Functional Quantum Programming Language [Thesis]. University of Nottingham, England.

[4] Saputra, H. 2009. Kajian Tentang Komputer Kuantum Sebagai Pengganti Komputer Konvensional di Masa Depan. Politeknik Negeri Sriwijaya. *Jurnal Generic*. **4(2):** 1907-4093.

[5] Dijkstra, E.W. 1975. Guarded Commands, Nondeterminacy and the Formal Derivation of Programs. *Proceeding Programming Language; Netherlands, August 1975. Communication of the Association for Computing Machinary*. pp 453-457.

[6] Morgan, C., A. Mclver, and K. Seidel. 1996. Probabilistic predicate transformers. *ACM Transactions on Programming Languages and Systems*, **18(3):** 325-353.

[7] Zuliani, P. 2004. Non-Deterministic Quantum Programming. *Proceeding QPL 2004. Facoltà di Scienze e Tecnologie Informatiche Libera Università di Bolzano Italy*. pp 179–195.

[8] Zuliani, P. 2005. Compiling Quantum Programs. *Journal Acta Informatica*. **00:** 1–39.