Volume 7, No. 1, January-February 2016
ISSN No. 0976-5697
International Journal of Advanced Research in Computer Science
RESEARCH PAPER
Available Online at www.ijarcs.info

Decomposition of Quantum Gates with Primitives Quantum Operations

Paramita Ray
Department of Computer Science
Dinabandhu Andrews institute of technology & management
Kolkata, India

Abstract: In classical computers, a bit is either 0 or 1 at a particular time. A quantum bit can exist not only in a state corresponding to the logical state 0 or 1 as in a classical bit, but also in states corresponding to a blend or superposition of these classical states. In other words, a qubit can exist as a zero, a one, or simultaneously as both 0 and 1, with a numerical coefficient representing the probability for each state. Quantum Computers use Quantum Mechanical properties to provide an exponential speed up in time and query processing capabilities [11]. In this section, we will try to decompose basic quantum gate and simulate with using Quantum computing language (QCL). QCL is a high level, architecture independent [4] programming language for quantum computers, with a syntax derived from classical procedural languages.

Keywords: Basic Quantum Gates, PMD, Quantum computing language, Quantum register, Quantum mechanic

I. INTRODUCTION

The idea of a quantum computer was first proposed by Nobel laureate Richard Feynman [10], in 1981. This new types of computer, "built of quantum mechanical elements which obey quantum mechanical laws", might one day perform efficient and accurate simulations of quantum systems. Quantum computers,[9] exploit the unique, non-classical properties of the quantum systems and allowing them to process exponentially large quantities of information in only polynomial time [11].

A. Qubits

Qubits are represented using Dirac notation in quntum mechanics. Bra-ket or (|>) notation [1] is use to represent qubit. So the expression |0> represents quantum zero, and |1> represents quantum one[9]. We can mathematically represent the state of a qubit at any given time is as a two-dimensional state space in C^2 with orthonormal basis vectors |1> and |0>. The superposition $|\psi\rangle$ of a qubit is represented as a linear combination of those basis vectors:

$$|\psi\rangle = a_0 |0\rangle + a_1 |1\rangle$$

Where a0 is the complex scalar amplitude of measuring |0>, and a1 the amplitude of measuring the value |1>. Amplitudes may [8] be thought of as "quantum probabilities" in that they represent the chance that a given quantum state will be observed when the superposition is collapsed.

B. Superposition State

Superposition is the first distinguishing trait of a quantum system. Rather than existing in one distinct state at a time, a quantum system is actually in all of its possible states at the same time [7]. With respect to a quantum computer, this means that a quantum register exists in a superposition of all its possible configurations of 0's and 1's [3] at the same time, unlike a classical system whose register contains only one value at any given time.

If Consider a 3 bit qubit register, an equally weighted superposition of all possible states would be denoted by:

$$|\psi> = \quad |000> + \quad |001> + \ldots + \quad |111>$$

II. BACKGROUND

A. Quantum Registers

Quantum registers made up of multiple qubits. When quantum registers collapsed are bit strings[1] whose length determines the amount of information they can store. In superposition [3], each qubit in the register is in a superposition [10] of |1> and |0>, and a register of n qubits is in a superposition of all 2^n possible bit strings that could be represented using n bits.

For example a three-qubit register would thus have the following expansion:

$$|\psi_2\rangle = a_0 |000\rangle + a_1 |001\rangle + a_2 |010\rangle + a_3 |011\rangle + a_4 |100\rangle + a_5 |101\rangle + a_6 |110\rangle + a_7 |111\rangle$$

Or in vector form, using the computational basis:

$$|\psi_2\rangle = a_0 \begin{bmatrix}1\\0\\0\\0\\0\\0\\0\\0\end{bmatrix} + a_1 \begin{bmatrix}0\\1\\0\\0\\0\\0\\0\\0\end{bmatrix} + a_2 \begin{bmatrix}0\\0\\1\\0\\0\\0\\0\\0\end{bmatrix} + a_3 \begin{bmatrix}0\\0\\0\\1\\0\\0\\0\\0\end{bmatrix} + a_4 \begin{bmatrix}0\\0\\0\\0\\1\\0\\0\\0\end{bmatrix} + a_5 \begin{bmatrix}0\\0\\0\\0\\0\\1\\0\\0\end{bmatrix} + a_6 \begin{bmatrix}0\\0\\0\\0\\0\\0\\1\\0\end{bmatrix} + a_7 \begin{bmatrix}0\\0\\0\\0\\0\\0\\0\\1\end{bmatrix}$$

B. Quantum logic gates

Quantum logic gates are the building blocks of quantum circuits, like classical logic gates [1] are] for conventional digital circuits. Unlike many classical logic gates, quantum logic gates are reversible [2].

1) **Hadamard Gate**

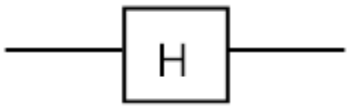

Figure 1: Hadamard Gate

Matrix representation of the Hadamard Gate (Fig2) is following:

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

Figure 2: Matrix representation of the Hadamard Gate

Whenever we apply the H gate parallelly[2] to a register initially at an arbitrary basis state x, where x is a binary number with n bits, the output state is a superposition of all the different 2^n basis states[5], where the absolute value of the amplitude of each basis state is $\frac{1}{\sqrt{2^n}}$ and the sign of the amplitude of such a basis state |z> in the resulting superposition[10] is positive if an even number of bits which were[3] 1 in the original state |x> are still 1 in |z>, and negative otherwise. In other words, the parallel application of H gates to all qubits of an n-qubit register initially at state |x> results in the state.

$$\frac{1}{\sqrt{2^n}} \sum_{z=0}^{2^n-1} (-1)^{x.z} |z\rangle,$$

where $x.z = \sum_{i=1}^{n} x_i.z_i$, such that x_i is the ith bit of x.

2) Controlled NOT (CNOT) Gate

The controlled NOT gate (or CNOT) acts on 2 qubits, and performs the NOT operation on the second qubit only when the first qubit is |1>, and otherwise leaves it unchanged. It is represented by the matrix.

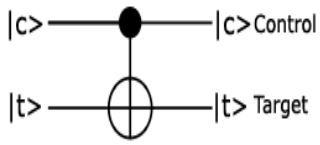

Figure 3: CNOT Gate

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

Figure 4: Matrix representation of the CNOT Gate

3) Taffoli Gate

The Toffoli gate, is a 3-bit gate, which is universal for classical [3] computation. It is also known as the controlled-controlled-NOT gate. If both control qubits are set, then the amplitudes of the target qubit are flipped.

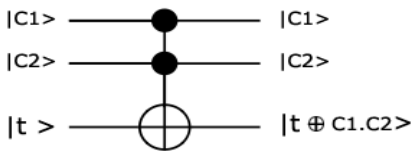

Figure 5: Toffoli Gate

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

Figure 6: Matrix representation of the Toffoli **Gate**

III. PHYSICAL MACHINE DESCRIPTIONS (PMD)

Quantum logic circuits are made of using quantum gates that works on qubits. A multi-qubit gate can be decomposed into a sequence of one-qubit and two-qubit quantum gates. But a one-qubit or two-qubit gate can not be directly implementable in a physical quantum machine. So it can be further decomposed using the set of supported primitive quantum operations in the physical machine description **(PMD)** [6] of the quantum machine. In quantum mechanics, the time evolution of a closed quantum system can be described by a unitary operator determined by its Hamiltonians. But different quantum systems have different Hamiltonians and they also have different PMDs.

So one must be able to control the Hamiltonians of the system for performing a quantum operation. But the problem is that an operation may be easily performed in one system but with difficulty in another. Hence, one PMD may be more suitable for implementing a quantum logic gate than another.

PMDs of the following six quantum systems

- **Quantum dot (QD):** Here a qubit is represented by the spin states of two electrons in a double electro statically defined quantum dot, which has two potential wells with a tunneling barrier between them.
- **Superconducting (SC):** In this system qubits can be represented by using charged carriers [6]. Two electrons can bind together to form a Cooper pair at low temperature. Such a pair can be confined within an electrostatic box and used to represent quantum information.
- **Ion trap (IT):** Here quantum system is based on a 2D lattice of confined ions, and each ion represents a physical qubit that can be moved within the lattice to accommodate local interactions [6].
- **Neutral atom (NA):** In this quantum system, trapped neutral atoms [6] that can be isolated from the environment and whose simple quantum-level structure can be exploited.
- **Linear photonics (LP):** Here, a probabilistic two-photon gate is teleported into a quantum circuit with high probability.

- **Nonlinear photonics (NP):** This quantum system is based on weak cross-Kerr nonlinearities

IV. OUR WORK

Here we will try to decompose basic gates with some quantum operations and implements it using Quantum computing language (QCL).

V. QUANTUM COMPUTING LANGUAGE (QCL)

QCL (an acronym for "quantum computation language") is an experimental structured quantum programming language [10]. A QCL interpreter, written in C++, including a numerical simulation library (libqc) to emulate the quantum [4]backend is available from **http://tph.tuwien.ac.at//~oemer/qcl.html.**
According to the hybrid architecture as introduced in below figure, the numerical simulations are handled by a library (libqc) to separate the classical program state from the quantum machine state.

VI. CIRCUITS SYNTHESIS (PMD)

Here we show how each of the gates can be implemented on each of the six PMDs. In this project mainly we decompose Hadamard Gate,CZ, CNOT,and Toffoli Gate.

A**. H Gate**

An *H* Gate is not directly supported in four of the six PMDs (except for the photonic systems).So we can construct it through a sequence of rotations. *H* gate can be obtained by cascading some rotations that include *Ry*[6] in most of the PMDs. Since *Ry* is not available in QD. So *H* can be obtained from a sequence composed of *Rz* and *Rx*:

$$H = P(\frac{\pi}{2}) \cdot R_z(\frac{\pi}{2}) \cdot R_x(\frac{\pi}{2}) \cdot R_z(\frac{\pi}{2})$$

```
operator hadamard(qureg a)
{
const n=#a;
int i;
for i=0 to n-1 step 1
{
RotZ(pi/2,a[i]);
RotX(pi/2,a[i]);
RotZ(pi/2,a[i]);

if a[i]{Phase(pi/2);}else{ Phase(pi/2);}
}
}

qureg b[4];
hadamard(b);
```

B. **CZ Gate**

CZ is the most supported two-qubit gate among the six PMDs [6], and is hence supported by SC). A *CZ* gate can be derived from a *CNOT* gate (for the NP system) because *CZ* and *CNOT* are dual gates along the *z*-axis and *x*-axis [12].

1) **NP System**

```
operator CZ(qureg a,qureg b)
{
 H(b);
CNot(b,a);
H(b);

}
qureg x[1];
qureg y[1];
Not(x);
CZ(x,y);
```

```
operator CZ(qureg a,qureg b)
{
 RotY(pi/2,b);
CNot(b,a);
RotY(-pi/2,b);
}

qureg x[1];
qureg y[1];
Not(x);
CZ(x,y);
```

2) **IT System**

```
operator CZ(qureg a,qureg b)
{
 if a[0]{Phase(pi/2);}else{ Phase(pi/2);}
 RotZ(pi/2,a);
RotZ(pi/2,b);
if a[0]{Phase(-pi/4);}else{ Phase(-pi/4);}
RotZ(-pi/2,b);
RotZ(pi,b);
}
qureg x[1];

qureg y[1];
Not(x);
CZ(x,y);
```

C. **CNOT Gate**

CNOT is a fundamental part of quantum computation, only the two photonic systems directly support it. As shown in Figure. 7(*a*), *CNOT* can be implemented using H and *CZ* gates.

```
operator  CZ(qureg a,qureg b)
{
 RotY(pi/2,b);
CNot(b,a);
RotY(-pi/2,b);
}

qureg x[1];
qureg y[1];
Not(x);
CZ(x,y);
```

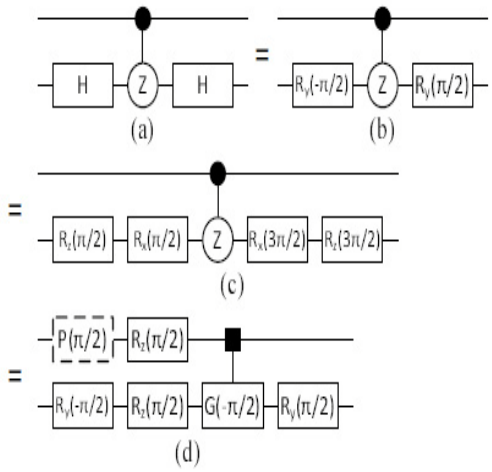

Figure7: CNOT implementation a) from H gate b)SC ,NA c) QD and d)IT system[6]

QCL Implementation

1) **Using H gate**

```
operator CNOT (qureg a,qureg b)

{
H(b);
CZ(a,b);
H(b);
}
```

```
operator  CZ(qureg x,qureg y)

{ if x[0]{Phase(pi/2);}else{ Phase(pi/2);}
 RotZ(pi/2,x);
RotZ(pi/2,y);
if x[0]{Phase(-pi/4);}else{ Phase(-pi/4);}

RotZ(-pi/2,y);
RotZ(pi,y);
}
```

2) **SC and NA System**

```
operator CNOT (qureg a,qureg b)

{

RotY(-pi/2,b);

CZ(a,b);

RotY(pi/2,b);
}
```

```
operator  CZ(qureg x,qureg y)

{ if x[0]{Phase(pi/2);}else{ Phase(pi/2);}
 RotZ(pi/2,x);
RotZ(pi/2,y);
if x[0]{Phase(-pi/4);}else{ Phase(-pi/4);}

RotZ(-pi/2,y);
RotZ(pi,y);
}
```

3) **QD System**

```
operator CNOT (qureg a,qureg b)

{ RotZ(pi/2,b);
RotX(pi/2,b);

CZ(a,b);

RotX(3*pi/2,b);

RotZ(3*pi/2,b);
}
```

```
operator  CZ(qureg x,qureg y)

{ if x[0]{Phase(pi/2);}else{ Phase(pi/2);}
 RotZ(pi/2,x);
RotZ(pi/2,y);
if x[0]{Phase(-pi/4);}else{ Phase(-pi/4);}

RotZ(-pi/2,y);
RotZ(pi,y);
}
```

```
operator CNOT (qureg a,qureg b)

{
Phase(pi/2,a)
RotZ(pi/2,a);
RotY(-pi/2,b);
RotZ(pi/2,b);
G(a,b);
RotY(pi/2,b);


}
```

```
operator G(qureg x,qureg y)
{
if x[0]{Phase(pi/4);}else{ Phase(pi/4);}
CNot(y,x);
RotZ(pi/2,y);
CNot(y,x);

}
```

D. **Taffoli Gate**

It is a three-qubit gate. To implement it efficiently, we hierarchically decompose it into PMD-independent two-qubit gates[6] first and then into primitive quantum operations supported by various PMDs.

1) **SC System**

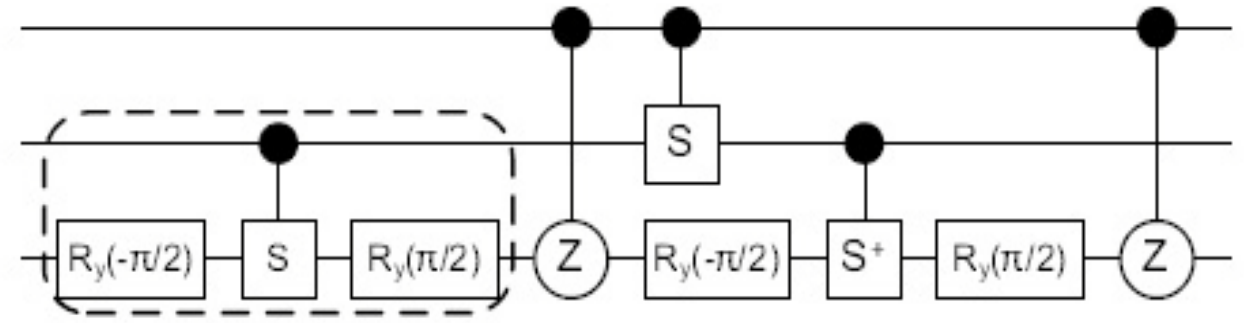

Figure8: Toffoli suitable for SC System[6]

QCL Implementation(SC System)

```
qureg a[1];
qureg b[1];
qureg c[1];
Not(a[0]);
Not(b[0]);
Not(c[0]);
/*-Ry implementation */
RotY(-pi/2,c[0]);
/* cs structure(special case of cp
 i.e cp(pi/2) */
RotZ(pi/4,b[0]);
if b[0]{Phase(pi/8);}else{ Phase(pi/8);}
RotZ(pi/4,c[0]);
if c[0]{Phase(pi/8);}else{ Phase(pi/8);}
CNot(c[0],b[0]);
if c[0]{Phase(pi/8);}else{ Phase(pi/8);}
RotZ(pi/4,c[0]);
CNot(c[0],b[0]);
/* -Ry implementation */
RotY(-pi/2,c[0]);
/* cs+ structure */
CNot(c[0],b[0]);
if c[0]{Phase(pi/8);}else{ Phase(pi/8);}
RotZ(pi/4,c[0]);
CNot(c[0],b[0]);
RotZ(pi/4,b[0]);
if b[0]{Phase(pi/8);}else{ Phase(pi/8);}
RotZ(pi/4,c[0]);
if c[0]{Phase(pi/8);}else{ Phase(pi/8);}
```

```
/* Ry implementation */
RotY(pi/2,c[0]);

/* CZ structure(special case of cp i.e cp(pi) */
CPhase(pi,a);
/* cs structure(special case of cp i.e cp(pi/2) */
RotZ(pi/4,a[0]);
if a[0]{Phase(pi/8);}else{ Phase(pi/8);}
RotZ(pi/4,b[0]);
if b[0]{Phase(pi/8);}else{ Phase(pi/8);}

CNot(b[0],a[0]);
if b[0]{Phase(pi/8);}else{ Phase(pi/8);}
RotZ(pi/4,b[0]);
CNot(b[0],a[0]);
```

```
/*Ry implementation */
RotY(pi/2,c[0]);
/* CZ structure(special case of cp i.e cp(pi) */
CPhase(pi,a);
```

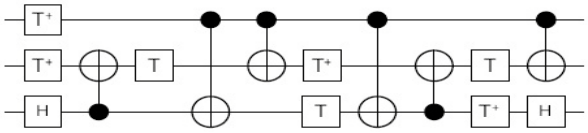

Figure 9. **:**Toffoli suitable for NP and QD System[6]

QCL Implementation

```
qureg a[1];
qureg b[1];
qureg c[1];
/* T+ structure */
if a[0]{Phase(pi/8);}else{ Phase(pi/8);}
RotZ(pi/4,a[0]);
/* T+ structure */
if b[0]{Phase(pi/8);}else{ Phase(pi/8);}
RotZ(pi/4,b[0]);
H(c);
CNot(b,c);
```

```
/* T structure */
RotZ(pi/4,b[0]);
if b[0]{Phase(pi/8);}else{ Phase(pi/8);}
CNot(c,a);
CNot(b[0],a[0]);
/* T+ structure */
if b[0]{Phase(pi/8);}else{ Phase(pi/8);}
RotZ(pi/4,b[0]);
/* T structure */
RotZ(pi/4,c[0]);
if c[0]{Phase(pi/8);}else{ Phase(pi/8);}
```

```
CNot(c,a);
CNot(b,c);
/* T structure */
RotZ(pi/4,b[0]);
if b[0]{Phase(pi/8);}
else{ Phase(pi/8);}
```

```
/* T+ structure */
if c[0]{Phase(pi/8);}else{ Phase(pi/8);}
RotZ(pi/4,c[0]);
CNot(b[0],a[0]);
H(c);
```

VII. CONCLUSIONS

Here we decomposed basic gates (Hadamard, CNOT,Toffoli) using the set of supported primitive quantum operations in the physical machine description **(PMD)** [6] of the quantum machine. We use quantum computing Language for simulation of Gates.

VIII. ACKNOWLEDGEMENTS

I am thankful all of my friends, colleagues of Dinabandhu andrews institute of technology & management.

IX. REFERENCES

1) S. M. Aamini, Quantum algorithms of decision and planning problems, Oriental, J. Math., Vol. 1, Number 1,2009, Pages 27-29, Oriental Academic Publishers.
2) Emma Strubell, An Introduction to Quantum Algorithms,Chawathe Spring 2011, COS498.
3) Christoph D¨urr, Peter Høyer. A quantum algorithm for finding the minimum. http://arxiv.org/abs/quant-ph/9 607014, vol 2 18 Jul 1996,Cornell University Library.
4) Bernhard Omer, Quantum Programming in QCL. http://tph.tuwien.ac.at/~oemer/doc/quprog.pdf, 20th January 2000, Institute of Information Systems Technical University of Vienna.
5) Simona Arustei and Vasile Manta, QCL IMPLEMENTATION OF THE BERNSTEIN-VAZIRANI ALGORITHM, January 2008, University Tehnică.
6) Chia-Chun Lin, Amlan Chakrabarti and Niraj K. Jha, *Fellow, IEEE,* Optimized Quantum Gate Library for Various Physical Machine Descriptions, Volume:21 , Issue: 11 29 January 2013, Very Large Scale Integration (VLSI) Systems.
7) Dan Kenigsberg, Grover's Quantum Search Algorithm and Mixed States, http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-get.cgi/2001/MSC/MSC-2001-01.pdf, October 2001, Israel Institute of Technology, Heshvan 5762 Haifa.
8) M. Z. Rashad, Quantum parity algorithms as oracle calls, and application in Grover Database search,Vol.3, No.2, March 2012, Advanced Computing: An International Journal (ACIJ).
9) Amlan Chakrabarti, Chia-Chun Lin and Niraj K. Jha, Design of Quantum Circuits for Random Walk Algorithms, 19-21 Aug. 2012 Computer Society Annual Symposium on VLSI.
10) Mark Oskin, Quantum Computing - Lecture Notes, http://www.cs.washington.edu/homes/oskin, spring quarter/2002 University of Washington.
11) Michael A. Nielsen, Isaac L.Chuang, Quantum Computation and Quantum Information,vol 1,January2011.Cambridge University Press.
12) Yadollah Farahmand ,Zabialah Heidarnezhad , Fatemeh Heidarnezhad , Kh. Kh. Muminov ,Fatemeh Heydari and Ghasem Saidi Tabar, The structure of qubit and quantum gates in quantum computers, http://www.orientjchem.org/vol30no4/the-structure-of-qubit-and-quantum-gates-in-quantum-computers/, Volume 30.