Implementarea algoritmilor cuantici – studiu de caz

Adina BĂRÎLĂ
„Ștefan cel Mare” University of Suceava, Romania
adina@eed.usv.ro

Abstract. **Simulating of quantum algorithms is very important since quantum hardware is not available outside of the research labs. The paper presents an implementation in QCL of the Simon algorithm in the case of bidimensional registers. QCL (Quantum Computation Language) is the most advanced implemented quantum computer simulator and was conceived by Bernhard Ömer.**

Keywords: calcul cuantic, porți cuantice, algoritmi cuantici.

I.INTRODUCERE

Calculul cuantic este un nou domeniu al științei care investighează puterea de calcul a calculatoarelor bazate pe principiile mecanicii cuantice și urmărește să găsească algoritmi mai rapizi decât algoritmii clasici care rezolvă aceeași problemă. Există două modele pentru calculul cuantic: versiunea cuantică a mașinii Turing și circuitele cuantice. Ambele modele au fost introduse de Deutsch[1][2]. De asemeni, David Deutsch a inventat primul algoritm cuantic care rezolvă o problemă mai eficient decât un algoritm clasic. Alți algoritmi notabili au fost proiectați de Simon și Vazirani. Însă cele mai importante rezultate în domeniul calculul cuantic sunt considerați algoritmii lui Shor și Grover. În 1994, Peter Shor a descris un algoritm în timp polinomial pentru factorizarea întregilor [3] iar în 1996 Lov Grover a inventat algoritmul cuantic de căutare într-o bază de date [4].

Scopul articolului este să prezinte o implementare a algoritmului lui Simon în Quantum Computation Language (QCL). Secțiunea II prezintă conceptele de bază ale calculului cuantic. Secțiunea III prezintă o clasificare a simulatoarelor de calculatoare cuantice, iar secțiunea IV introduce algoritmul lui Simon și implementarea sa în QCL. Secțiunea V trasează concluzii și dezvoltări ulterioare.

ACKNOWLEDGMENT: Aceasta lucrare a beneficiat de suport financiar prin proiectul "Performanta sustenabila in cercetarea doctorala si post doctorala - PERFORM", Contract nr. POSDRU/159/1.5/S/138963", proiect cofinantat din Fondul Social European prin Programul Operational Sectorial Dezvoltarea Resurselor Umane 2007-2013

II.CALCUL CUANTIC – CONCEPTE DE BAZĂ

Unitatea de informație cuantică se numește quantum bit sau, pe scurt, qubit [5]. Un qubit este un sistem fizic care are două stări de bază, notate convențional $|0\rangle$ și $|1\rangle$, corespunzătoare valorilor clasice 0 și 1. Spre deosebire de bitul clasic, statea generală a unui qubit este o combinație liniară – sau o superpoziție – a stărilor de bază:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \quad (1)$$

unde amplitudinile α și β sunt numere complexe astfel încât:

$$|\alpha|^2+|\beta|^2=1 \quad (2)$$

Cu alte cuvinte, un qubit poate exista fie în starea $|0\rangle$, fie în starea $|1\rangle$, fie poate conține simultan valorile 0 și 1 (când ambii coeficienți ai superpoziției sunt diferiți de zero). Formal, o stare cuantică este un vector unitar într-un spațiu Hilbert.

Un sistem format din n qubiți are 2^n stări de bază iar starea sa generală este o superpoziție a tutror acestor stări:

$$\left|\psi\right\rangle = \sum_{k=0}^{2^n-1} c_k \left|k\right\rangle \quad (3)$$

unde:

$$\left|k\right\rangle = \left|k_{n-1}\right\rangle \ldots \left|k_1\right\rangle \left|k_0\right\rangle \quad (4)$$

cu $|k_j\rangle$ reprezentând starea qubit-ului al j-lea. Amplitudinile c_k sunt numere complexe ce satisfac relația:

$$\sum_{k=0}^{2^n-1} \left|c_k\right|^2 = 1 \quad (5)$$

Ca și în cazul sistemului cu un singur qubit, un sistem cu n qubiți poate stoca simultan toate stările de bază.

Evoluția unui sistem cuantic poate fi descrisă de o transformare unitară *U*. O transformare unitară care acționează pe un număr mic de qubiți se numește poartă. O poartă cuantică are același număr de intrări și ieșiri. O poartă cuantică elementară pe un qubit este descrisă de o matrice de dimensiune 2x2 astfel:

$$U = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \quad (6)$$

care transformă $|0\rangle$ în $a|0\rangle+b|1\rangle$ și $|1\rangle$ în $c|0\rangle+d|1\rangle$. Porțile Hadamard (H) și Pauli (X,Y,Z) sunt exemple de porți cuantice care acționează pe un singur qubit:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad (7)$$

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad (8)$$

Cea mai importantă poartă pe doi qubiți este poarta CNOT (controlled-not). Aceasta acționează pe doi qubiți, qubit-ul de control și qubit-ul țintă. Qubitul țintă este modificat numai dacă qubit-ul de control este setat la 1. Forma matriceală a acestei porți este:

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (9)$$

Poarta CNOT este o generalizare a porții clasice XOR, acțiunea sa putând fi scrisă ca $|x,y\rangle \rightarrow |x, y \oplus x\rangle$, unde $\oplus$ reprezintă adunarea modulo doi.

În general, dacă *U* este o poartă pe un singur qubit cu reprezentarea matriceală:

$$U = \begin{pmatrix} x_{00} & x_{01} \\ x_{10} & x_{11} \end{pmatrix} \quad (10)$$

atunci poarta controlled-*U* este o poartă pe doi qubiți cu reprezentarea matriceală:

$$C(U) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & x_{00} & x_{01} \\ 0 & 0 & x_{10} & x_{11} \end{pmatrix} \quad (11)$$

Primul qubit este qubit-ul de control.

Poarta SWAP este generalizarea cuantică a porții clasice CROSSOVER și inversează stările cuantice ale qubiților. Reprezentarea matriceală este:

$$SWAP = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (12)$$

Poarta Cph (controlled phase) acționează pe doi qubiți și nu are achivalent clasic.

$$U_{cph}(\phi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \exp(i\phi) \end{pmatrix} \quad (13)$$

Măsurarea este singura operație nereversibilă care poate fi aplicată unei stări cuantice. După măsurare, starea cuantică a unui qubit colapsează în una dintre cele două stări de bază, deci măsurarea este o operație distructivă. Dacă un qubit este în starea $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ și este efectuată o măsurarea, se obține 0 cu probabilitatea α^2 (iar starea qubitului devine $|0\rangle$) și 1 cu probabilitatea β^2 (iar starea qubit-ului devine $|1\rangle$).

III. SIMULAREA CALCULELOR CUANTICE

Simulatoarele de calculatoare cuantice sunt definite ca fiind programe de calculator care pot fi rulate pe o maşină clasică pentru a simula acţiunile unui computer cuantic. Aceste simulări implică utilizarea maşinilor care lucrează în acord cu legile fizicii clasice newtoniene pentru a simula maşini care lucrează în acord cu legile mecanicii cuantice

Simulatoarele de calculatoare cuantice pot fi clasificate astfel [6]:

- limbaje de programare pentru calculatoare cuantice
- compilatoare cuantice
- simulatoare de circuite cuantice sau simulatoare la nivel de porţi cuantice
- software care simulează modele pentru realizarea fizică a calculatoarelor cuantice
- simulatoare pur pedagogice

Limbajele de programare exprimă semantica unui proces de calcul într-o manieră abstractă şi generează automat o secvenţă de operaţii elementare pentru a controla calculatorul. Din această categorie face parte Quantum Computation Language – QCL, Q language, Quantum Superpositions, QuBit, Quantum Entanglement, Q-gol, Quantum Fog, QDD şi Quantum Lambda Calculus.

QCL (Quantum Computation Language) este un limbaj de nivel înalt pentru calculatoare cuantice, independent de arhitectură [6]. A fost conceput de Bernhard Ömer, versiunea curentă fiind apărută în 2004. QCL a fost implementat în C și rulează sub Linux. Sintaxa și tipurile de date sunt similare celor din C. Tipul de date cuantice de bază este *qreg* (quantum register) și poate fi interpretat ca un tablou de qubiți.

Caracteristicile principale ale limbajului sunt [7][8]:

- control clasic cu funcţii, flow-control, operaţii de intrare/ieşire şi diferite tipuri de date clasice (int, real, complex, boolean, string)
- două tipuri de operatori cuantici: generali (operator) şi porţi reversibile pseudo-clasice (qufunc)
- tipuri de date cuantice (qubit registers)
- funcţii pentru manipularea regiştrilor cuantici
- limbaj universal: poate implementa şi simula toţi algoritmii cuantici cunoscuţi

QCL este proiectat să lucreze cu orice arhitectură de computer cuantic. De asemeni, QCL suportă şi simularea computerelor cuantice şi oferă comenzi speciale pentru accesarea stării simulate a maşinii.

IV. ALGORITMUL LUI SIMON

A. Prezentare

Daniel Simon [9] a propus următoarea problemă: fie o funcţie *f* de forma:

$$f: \{0,1\}^n \rightarrow \{0,1\}^n$$

pentru un *n* întreg pozitiv. Funcţia *f* are proprietatea că există un şir $s \in \{0,1\}^n$, $s \neq 0$ astfel încât:

$$\forall x,y \in \{0,1\}^n,\ f(x) = f(y) \Leftrightarrow y = x \oplus s$$

Scopul este aflarea perioadei *s*.

De exemplu, dacă *n*=2, funcţia următoare este un exemplu de funcţie care satisface proprietatea cerută:

x	*f(x)*
00	00
01	01
10	00
11	01

În acest exemplu, şirul *s* este 10.

Algoritmul cuantic care rezolvă această problemă are o parte cuantică şi una clasică [10]. În partea cuantică este iterat circuitul cuantic din figura de mai jos:

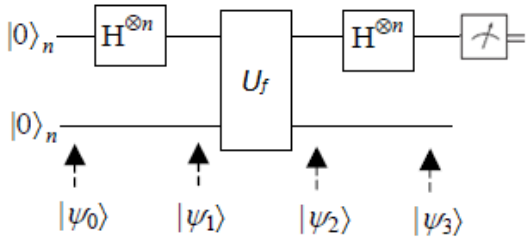

Fig.1 Circuitul cuantic pentru algoritmul lui Simon [11]

Transformarea U_f este definită prin:

$$(14) \quad U_f|x\rangle\,|y\rangle = |x\rangle\,|f(x) \oplus y\rangle$$

unde $\oplus$ reprezintă operaţia XOR pe biţi.

Circuitul cuantic pentru implementarea algoritmului Simon are doi regiştri. Fiecare registru are *n* qubiţi şi este iniţializat în starea $|00\ldots00\rangle$. Deci, starea iniţială a sistemului este:

$$(15) \quad |\psi_0\rangle = |0\rangle_n |0\rangle_n$$

După efectuarea transformărilor Hadamard pe primul registru, starea cuantică devine:

$$(16) \quad |\psi_1\rangle = \left(H^{\otimes n} \otimes I_{2^n}\right)|0\rangle_n |0\rangle_n = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle_n |0\rangle_n$$

În următorul pas, transformarea U_f acţionează pe ambii regiştri şi rezultă starea:

$$(17) \quad |\psi_2\rangle = U_f \left(\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle_n |0\rangle_n \right) = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle_n |f(x)\rangle_n$$

În final, sunt aplicate transformările Hadamard iar starea cuantică a sistemului devine:

$$(18) \quad |\psi_3\rangle = \left(H^{\otimes n} \otimes I_{2^n}\right)|\psi_2\rangle = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} H^{\otimes n} |x\rangle_n |f(x)\rangle_n$$

Ştiind că:

$$(19) \quad H^{\otimes n}|x\rangle_n = \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} (-1)^{x \cdot y} |y\rangle_n$$

starea finală poate fi scrisă ca:

$$(20) \quad |\psi\rangle_3 = \frac{1}{2^n} \sum_{x,y=0}^{2^n-1} (-1)^{x \cdot y} |y\rangle_n |f(x)\rangle_n$$

Acum este măsurată valoarea primului registru.
În cazul $s \neq 0^n$, probabilitatea de a măsura o valoare este:

$$(21) \quad P(y) = \left| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot y} |f(x)\rangle \right|^2$$

Fie A = range(*f*) şi fie $z \in A$. Din definiţia funcţiei *f*, există exact două valori posibile x_z, $x'_z \in \{0,1\}^n$ astfel încât

$$f(x_z) = f(x'_z) = z,$$

şi mai mult $x'_z = x_z \oplus s$. Deci,

$$P(y) = \left| \frac{1}{2^n} \sum_{z \in A} \left((-1)^{x_z \cdot y} + (-1)^{x'_z \cdot y} \right) |z\rangle \right|^2$$

$$(22) \quad = \left| \frac{1}{2^n} \sum_{z \in A} \left((-1)^{x_z \cdot y} + (-1)^{(x_z \oplus s) \cdot y} \right) |z\rangle \right|^2$$

$$= \left| \frac{1}{2^n} \sum_{z \in A} (-1)^{x_z \cdot y} \left(1 + (-1)^{s \cdot y} \right) |z\rangle \right|^2$$

$$(23)\quad P(y)=\begin{cases}\dfrac{1}{2^{n-1}} & if\ s\cdot y=0\\ 0 & if\ s\cdot y=1\end{cases}$$

Deci, prin măsurare se obține întotdeauna o valoare y care satisface relația $s\cdot y = 0$.

Măsurând primul registru se obține o valoare $y_1\in\{0,1\}^n$ cu $y_1\cdot s=0$. Algoritmul este executat din nou și se măsoară primul registru obținându-se o nouă valoare, $y_2\in\{0,1\}^n$ cu $y_2\cdot s=0$, $y_2\neq y_1$ și $y_2\neq 0$. Algoritmul lui Simon este repetat de n-1 ori obținându-se un sistem de n-1 ecuații de forma:

$$y_1\cdot s = 0$$
$$y_2\cdot s = 0$$
$$\dots$$
$$y_{n-1}\cdot s = 0$$

Partea de post-procesare clasică constă în rezolvarea acestui sistem de ecuații în n necunoscute (biții lui s) pentru a-l afla pe s.

B.Algoritmul lui Simon pentru cazul n=2

În cazul n = 2, circuitul cuantic are doi regiștri de dimensiune 2, ambii fiind inițializați la starea $|00\rangle$. Conform definiției transformării U_f (relația (14)), în exemplul dat mai sus, acțiunea lui U_f poate fi descrisă ca:

$U_f|00\rangle|00\rangle=|00\rangle|00\oplus f(00)\rangle=|00\rangle|00\oplus 00\rangle=|00\rangle|00\rangle$
$U_f|00\rangle|01\rangle=|00\rangle|01\oplus f(00)\rangle=|00\rangle|01\oplus 00\rangle=|00\rangle|01\rangle$
$U_f|00\rangle|10\rangle=|00\rangle|10\oplus f(00)\rangle=|00\rangle|10\oplus 00\rangle=|00\rangle|10\rangle$
$U_f|00\rangle|11\rangle=|00\rangle|11\oplus f(00)\rangle=|00\rangle|11\oplus 00\rangle=|00\rangle|11\rangle$

$U_f|01\rangle|00\rangle=|01\rangle|00\oplus f(01)\rangle=|01\rangle|00\oplus 01\rangle=|01\rangle|01\rangle$
$U_f|01\rangle|01\rangle=|00\rangle|01\oplus f(01)\rangle=|01\rangle|01\oplus 01\rangle=|01\rangle|00\rangle$
$U_f|01\rangle|10\rangle=|01\rangle|10\oplus f(01)\rangle=|01\rangle|10\oplus 01\rangle=|01\rangle|11\rangle$
$U_f|01\rangle|11\rangle=|01\rangle|11\oplus f(01)\rangle=|01\rangle|11\oplus 01\rangle=|01\rangle|10\rangle$

$U_f|10\rangle|00\rangle=|10\rangle|00\oplus f(10)\rangle=|10\rangle|00\oplus 00\rangle=|10\rangle|00\rangle$
$U_f|10\rangle|01\rangle=|10\rangle|01\oplus f(10)\rangle=|10\rangle|01\oplus 00\rangle=|10\rangle|01\rangle$
$U_f|10\rangle|10\rangle=|10\rangle|10\oplus f(10)\rangle=|10\rangle|10\oplus 00\rangle=|10\rangle|10\rangle$
$U_f|10\rangle|11\rangle=|10\rangle|11\oplus f(10)\rangle=|10\rangle|11\oplus 00\rangle=|10\rangle|11\rangle$

$U_f|11\rangle|00\rangle=|11\rangle|00\oplus f(11)\rangle=|11\rangle|00\oplus 01\rangle=|11\rangle|01\rangle$
$U_f|11\rangle|01\rangle=|11\rangle|01\oplus f(11)\rangle=|11\rangle|01\oplus 01\rangle=|11\rangle|00\rangle$
$U_f|11\rangle|10\rangle=|11\rangle|10\oplus f(11)\rangle=|11\rangle|10\oplus 01\rangle=|11\rangle|11\rangle$
$U_f|11\rangle|11\rangle=|11\rangle|11\oplus f(11)\rangle=|11\rangle|11\oplus 01\rangle=|11\rangle|10\rangle$

Transformarea U_f poate fi reprezentă sub forma unei matrici de dimensiune 2^4x2^4 și poate fi reprodusă prin porți cuantice CNOT ca în fig. 2.

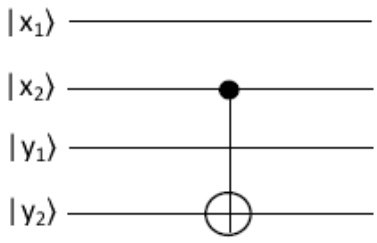

Fig.2.Transformarea U_f pentru algoritmul lui Simon

$$U_f=\begin{pmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0
\end{pmatrix}$$

C.Implementarea QCL

Implementarea algoritmului Simon în QCL (partea cuantică) este prezentată mai jos:

```
//se declara cei doi registri
//cu cate doi qubiti
qureg x[2];
qureg y[2];

int m; //valoarea masurata


{
reset;
//se aplica transformarea Hadamard
H(x);
//se aplica transformarea Uf
CNot(y[0], x[0]);
//se aplica transformarea Hadamard
H(x);
//se masoara registrul x
measure x,m;
print "Valoarea determinata este:", m;
} until (m!=0);
```

În figura 3 pot fi observate valorile măsurate la diferite execuții ale programului. La prima execuție, valoarea determinată este 1 (01_2). Deci, se rezolvă ecuația în 2 necunoscute

$$s_1\cdot 0 + s_2\cdot 1 = 0$$

unde s_1, s_2 sunt biții șirului s și toate operațiile sunt modulo 2. Această ecuație are două soluții:

$$s_1 = s_2 = 0$$

și

$$s_1 = 1\ ,\ s_2 = 0$$

Însă s trebuie să fie diferit de 0, deci singura soluție validă este s = 10 (2_{10}).

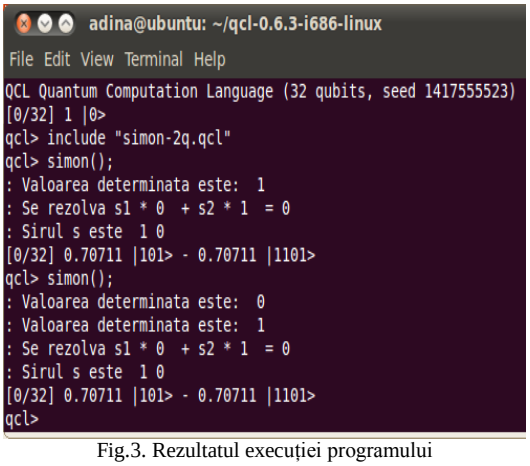

Fig.3. Rezultatul execuției programului

IV. Concluzii

În absența dispozitivelor cuantice, simulatoarele de calcul cuantic ajută programatorii să exploateze caracteristicile calculatoarelor cuantice și să înțeleagă restricțiile impuse de aceste dispozitive. În ultimii ani au fost dezvoltate multe simulatoare cuantice pentru a simula algoritmi cuantici. În acest articol a fost utilizat limbajul de programare cuantică QCL pentru a simula algoritmul dezvoltat de David Simon și cunoscut sub numele algoritmul Simon. Acesta a fost implementat pentru cazul regiștrilor bidimensionali. Transformarea oracol a fost simulată prin porți CNOT. În dezvoltările ulterioare vor fi luate în considerare implementări ale algoritmului pentru alte dimensiuni ale regiștrilor de intrare.

References

[1] D. Deutsch, “Quantum theory, the Church-Turing principle and the universal quantum computer”, Proceedings of the Royal Society of London A 400, pp. 97-117, 1985

[2] D. Deutsch, "Quantum computational networks", Proceedings of the Royal Society of London A 425, pp. 73-90, 1989

[3] P.W. Shor, “Algorithms for Quantum Computing: Discrete Logarithm and Factoring”, Proceedings of 35th Annual Symposium on Foundations of Computer Science, Los Alamitos, CA, USA, 1994, pp. 124-134

[4] L.K.Grover, “A fast quantum mechanical algorithm for database search”, Proc. 28th Annual ACM Symposium on the Theory of Computing (STOC), 1996, p. 212-219

[5] B. Schumacher, “Quantum coding”, Physical Review A, Vol. 51, No. 4, April 1995

[6] H. De Raedt, K. Michielsen. Handbook of theoretical and computational nanotechnology, volume 3, *Computational Methods for Simulating Quantum Computers*, American Scientific Publisher, Los Angeles, 2006, arXiv:quant-ph/0406210

[7] Ömer B., *Quantum Programming in QCL*, Technical University of Vienna, Austria, 2000.

[8] Ömer B., *Strucured Quantum Programming in QCL*, Technical University of Vienna, Austria, 2003.

[9] D. R. Simon, “On the Power of Quantum Computation”,SIAM Journal on Computing, no. 5, p. 1474.

[10] John Watrous, *Lecture Notes on Quantum Computing*, University of Waterloo, 2006

[11] Peter Nyman, *Simulation of Simon’s ALgorithm in Mathematica*, Quantum Computers and Computing, Vol. 8, No. 1, pp. 126, 2008