Implementarea algoritmilor cuantici – studiu de caz

Adina BĂRÎLĂ

Suceava, 2014

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

Bitul cuantic

- Un qubit este un sistem cu două stări de bază notate prin $|0\rangle$ și $|1\rangle$.
- Starea generală a unui qubit este o superpoziție sau o combinație liniară a stărilor de bază, adică:

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

unde amplitudinile α și β sunt numere complexe

$$|\alpha|^2 + |\beta|^2 = 1$$

- Notația matriceală:

$$|0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \qquad |1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$$

$$\begin{pmatrix}\alpha\\\beta\end{pmatrix} = \alpha|0\rangle + \beta|1\rangle = (\alpha, \beta)^T$$

Registru cuantic

- O mulțime de n qubiți formează un registru cuantic de dimensiune n.
- Starea unui qubit dintr-un registru cuantic de dimensiune n are forma:

$$|\psi\rangle = \sum_{k=0}^{2^n-1} C_k \, |k\rangle$$

unde

$$|k\rangle = |k_{n-1}\rangle \ldots |k_1\rangle |k_0\rangle$$

$|k_j\rangle$ reprezentând starea qubit-ului al j-lea și

$$\sum_{k=0}^{2^n-1} |C_k|^2 = 1$$

Măsurarea unui qubit

- Sistemul cuantic poate fi observat doar în stările de bază, însă el poate să existe în orice superpoziție a stărilor de bază atât timp cât nu este măsurat.

- Măsurarea qubiților este singura operație neunitară pe care un calculator cuantic trebuie să fie capabil să o execute în timpul calculului.

- Dacă un qubit este în starea $\alpha|0\rangle + \beta|1\rangle$ atunci măsurarea valorilor sale dă rezultatul 0 cu probabilitatea $|\alpha|^2$ (lăsându-l în starea $|0\rangle$ și rezultatul 1 cu probabilitatea $|\beta|^2$ (lăsându-l în starea $|1\rangle$).

Porţi cuantice

- Pentru a implementa un calcul pe un calculator cuantic trebuie să existe posibilitatea de a controla evoluţia în timp a stărilor dintr-un registru cuantic. Această evoluţie este descrisă de un operator unitar . Un astfel de operator constituie o poartă cuantică.

- Acţiunea unei porţi este echivalentă cu aplicarea operatorului corespunzător asupra stării iniţiale a qubitului.

- Deoarece calculul cuantic este reversibil, în orice poarta cuantica numarul de intrari si de iesiri trebuie sa fie acelasi. În plus, sunt permise în circuitele cuantice numai porţile cuantice care pastreaza tot timpul norma

$$\sum_x |c_x|^2 = 1$$

Porţi cuantice pe un singur qubit

- O poartă cuantică elementară pe un singur qubit este descrisă cu ajutorul unei matrici unitare 2x2 de forma:

$$U = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

care transformă starea $|0\rangle$ a unui qubit în $a|0\rangle+b|1\rangle$, respectiv starea $|1\rangle$ în $c|0\rangle+d|1\rangle$.

Porţi cuantice pe un singur qubit

- Unitate I $\mathrm{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
- Hadamard H $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
- Pauli X, Y, Z $\mathrm{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

$\mathrm{Y} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$

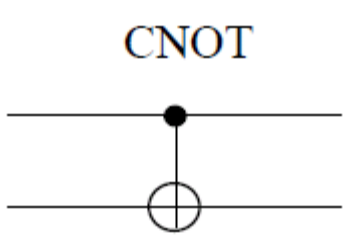
$\mathrm{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$

Porţi cuantice pe doi qubiţi

Poarta Controlled-NOT (CNOT)

$$\text{CNOT}\,|00\rangle = |00\rangle$$
$$\text{CNOT}\,|01\rangle = |01\rangle$$
$$\text{CNOT}\,|10\rangle = |11\rangle$$
$$\text{CNOT}\,|11\rangle = |10\rangle$$

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

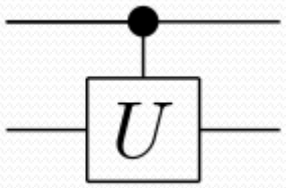


- Această poartă reprezintă analogia cuantica a porţii clasice revesibile XOR, $|a\rangle|b\rangle \rightarrow |a\rangle|a \oplus b\rangle$, cu $a, b \in \{0, 1\}$
- Ca şi poarta clasică XOR, poarta CNOT schimbă starea celui de-al doilea qubit (ţintă) dacă primul qubit (de control) este în starea $|1\rangle$ şi o lasă pe loc dacă primul qubit este în starea $|0\rangle$.

Porţi cuantice pe doi qubiţi

- **Poarta Controlled-U**

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

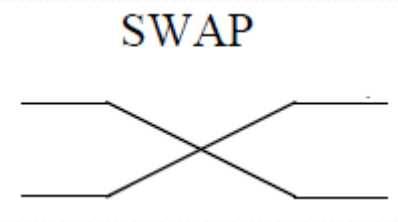


- **Poarta SWAP** este analogia transformarii clasice CROSSOVER:

$$|a\rangle|b\rangle \rightarrow |b\rangle|a\rangle$$

care interschimba starile a doi qubiţi şi are matricea:

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



Porţi cuantice pe doi qubiţi

- **Poarta Cph (controlled phase)** – acţionează asupra unei stări 2-qubit şi nu are analog clasic. Matricea corespunzătoare acestei porţi este:

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

Simulatoare de calculatoare cuantice

- programe de calculator care pot fi rulate pe o mașină clasică

  pentru a simula acţiunile unui computer cuantic

Clasificarea simulatoarelor

O clasificarea a simulatoarelor a fost realizată în 2006 de către H. De Raedt şi K. Michielsen. Ei au împărţit software-ul existent în cinci categorii:

- 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

Clasificarea simulatoarelor

- **Limbajele de programare pentru calculatoare cuantice** exprimă semantica unui proces de calcul într-o manieră abstractă și generează automat o secvență de operații elementare pentru a controla calculatorul.
  - ✓ Quantum Computation Language – QCL - este un limbaj de programare de nivel înalt pentru programare cuantică dezvoltat de Bernhard Ömer. Este independent de arhitectură și are o sintaxă asemănătoare limbajelor procedurale clasice precum C și Pascal.
  - ✓ Quantum Superpositions - bibliotecă PERL care permite programatorilor să utilizeze variabile care pot păstra mai multe valori în același timp
  - ✓ Quantum Entanglement – bibliotecă PERL care permite utilizatorilor să pună variabile într-o superpoziție de stări, să interacționeze între ele și să le observe
  - ✓ Quantum Fog - aplicație Macintosh pentru modelarea situațiilor fizice care prezintă comportament cuantic
  - ✓ QDD - bibliotecă C++ care oferă un set relativ intuitiv de construcții de calcul cuantic în contextul unui mediu de programare C++
  - ✓ Quantum Lambda Calculus - limbaj funcțional bazat pe Scheme pentru exprimarea și simularea algoritmilor cuantici

Limbajul QCL

- este un limbaj de programare cuantică de nivel înalt, scris în C++
- este open-source
- rulează sub Linux
- folosește o reprezentare a stării cuantice sub formă de numere complexe

Caracteristicile sale principale sunt:

- 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

Limbajul QCL

Un program QCL este o secvență de instrucțiuni și definiții citite dintr-un fișier sau direct de la prompt (în acest caz intrarea este restricționată la o linie terminată prin caracterul ‘;’)

Instrucțiunile pot fi comenzi simple, apeluri de proceduri, stucturi de control complexe și sunt executate cand apar.

```
qcl> if random()>=0.5 { print "red"; } else { print "black"; }
: red
```

Limbajul QCL

Type	Description	Examples
`int`	integer	`1234, -1`
`real`	real number	`3.14, -0.001`
`complex`	complex number	`(0,-1), (0.5, 0.866)`
`boolean`	logic value	`true, false`
`string`	character string	`"hello world", ""`

Op	Description	Example	Type	Value
`^`	power	`(0,1)^-1.5`	`complex`	$-\frac{1}{\sqrt{2}}(1+i)$
	integer power	`(-2)^11`	`int`	-2048
`-`	unary minus	`--1`	`int`	1
`*`	multiplication	`(0,1)*(0,1)`	`complex`	-1
`/`	division	`3./2`	`real`	$\frac{3}{2}$
	integer division	`3/2`	`int`	1
`mod`	integer modulus	`100 mod 16`	`int`	4
`+`	addition	`1.5+1.5`	`real`	3
`-`	subtraction	`(1,2)-(0,2)`	`complex`	1

Funct.	Description	Funct.	Description
`sin(x)`	sine of x	`sinh(x)`	hyperbolic sine of x
`cos(x)`	cosine of x	`cosh(x)`	hyperbolic cosine of x
`tan(x)`	tangent of x	`tanh(x)`	hyperbolic tangent of x
`cot(x)`	cotangent of x	`coth(x)`	hyperbolic cotangent of x

Funct.	Description
`Re(z)`	real part of z
`Im(z)`	imaginary part of z
`abs(z)`	magnitude of z
`conj(z)`	complex conjugate of z

Limbajul QCL

Tip	Descriere
`qureg`	un registru cuantic general
`quconst`	registru constant
`quvoid`	registru nealocat
`quscratch`	registru scratch

Expr.	Description	Register
`a`	reference	$\langle a_0, a_1 \ldots a_n \rangle$
`a[i]`	qubit	$\langle a_i \rangle$
`a[i:j]`	substring	$\langle a_i, a_{i+1} \ldots a_j \rangle$
`a[i\l]`	substring	$\langle a_i, a_{i+1} \ldots a_{i+l-1} \rangle$
`a&b`	concatenation	$\langle a_0, a_1 \ldots a_n, b_0, b_1 \ldots b_m \rangle$

```
procedure quRoulette() {
    qureg q[5];
    int field;
    int number;
    input "Enter field number:",field;
    {
        Mix(q);
        measure q,number;
        reset;
    } until number<=36;
    if field==number {
        print "Number",number,"You won!";
    } else {
        print "Number",number,"You lose.";
    }
}
```

Tip rutină
`procedure`
`operator`
`qufunct`
funcţie

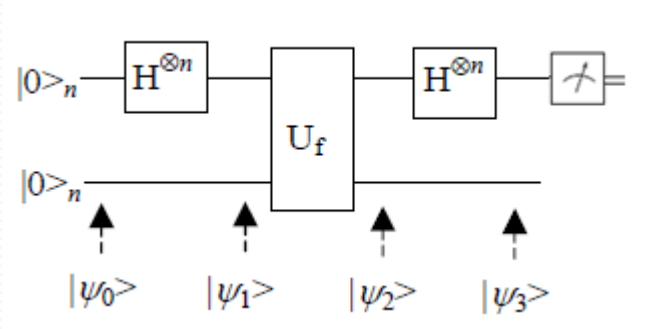
Limbajul QCL

```
extern qufunct Not(qureg q);

extern qufunct CNot(qureg q,quconst c);

extern operator Mix(qureg q);

extern qufunct Fanout(qureg a, qureg b);
FANOUT: |x,y⟩→|x,x⊕y⟩

extern qufunct Swap(qureg a, qureg b);
SWAP: |x,y⟩→|y,x⟩
```

Algoritmul lui Simon

Fie funcţia f : $\{0, 1\}^n \rightarrow \{0, 1\}^n$, unde n este un întreg pozitiv. Există un sir s ($s \in \{0,1\}^n$), astfel încât $f(x) = f(y) \Leftrightarrow y = x \oplus s$, pentru orice $x, y \in \{0; 1\}^n$. Problema consta în determinarea lui s.

Circuitul cuantic corespunzător algoritmului lui Simon:



U_f este o transformare unitară definită astfel:

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

Algoritmul lui Simon

Stările sistemului sunt:

$|\Psi_o\rangle = |o\rangle_n |o\rangle_n$

$$|\Psi_1\rangle = \left(H^{\otimes n} \otimes I_{2^n}\right)|0\rangle_n|0\rangle_n = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle_n|0\rangle_n$$

$$|\Psi_2\rangle = U_f \left(\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle_n|0\rangle_n \right) = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle_n|f(x)\rangle_n$$

$$|\Psi_3\rangle = 2^{-n} \sum_{y=0}^{2^n-1} \sum_{x=0}^{2^n-1} (-1)^{x\cdot y} |y\rangle_n |f(x)\rangle_n$$

$$= 2^{-(n+1)} \sum_{x,y=0}^{2^n-1} \left[(-1)^{x\cdot y} + (-1)^{(x\oplus a)\cdot y}\right] |y\rangle_n |f(x)\rangle_n$$

Algoritmul lui Simon

Se măsoară primul registru. Daca $s \cdot y = 1$ termenii din coeficientul lui $|y\rangle$ interferă distructiv, doar starile $|y\rangle$ cu $s \cdot y = 0$ ramanand in suma peste y. Rezultatul masuratorii este deci selectat aleator dintre toate valorile posibile y ce satisfac $s \cdot y = 0$, fiecare valoare avand probabililatea $2^{-(n+1)}$. Ruland algoritmul de mai multe ori, de fiecare data se obtine o alta valoare y care satisface $s \cdot y = 0$. Odata ce am gasit $n-1$ astfel de valori independente se pot rezolva equatiile $s \cdot y_i = 0$, $i = 1,.., n$ pentru a determina valoarea unica a lui s (care are n biti).

Algoritmul lui Simon pentru n=2

Fie $f:\{0,1\}^2 \rightarrow \{0,1\}^2$ definită astfel:

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

$f(00) = f(10) \leftrightarrow 10 = 00 + 10$

$f(01) = f(11) \leftrightarrow 11 = 01 + 10$

Deci șirul s este 10.

Algoritmul lui Simon pentru n=2

Definirea transformării Uf

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

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

Algoritmul lui Simon pentru n=2

Definirea transformării Uf - continuare

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

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

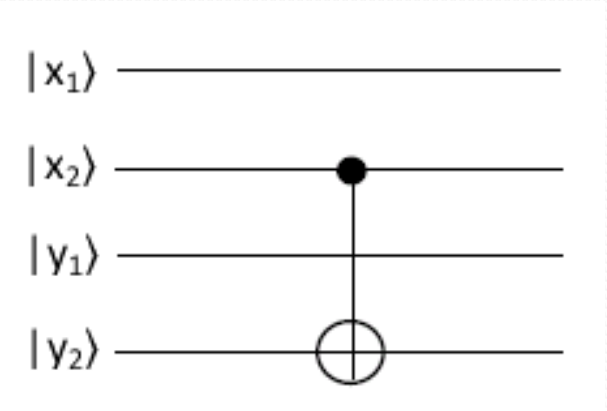
Algoritmul lui Simon pentru n=2

Transformarea U_f - reprezentare matriceală

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

Algoritmul lui Simon pentru n=2

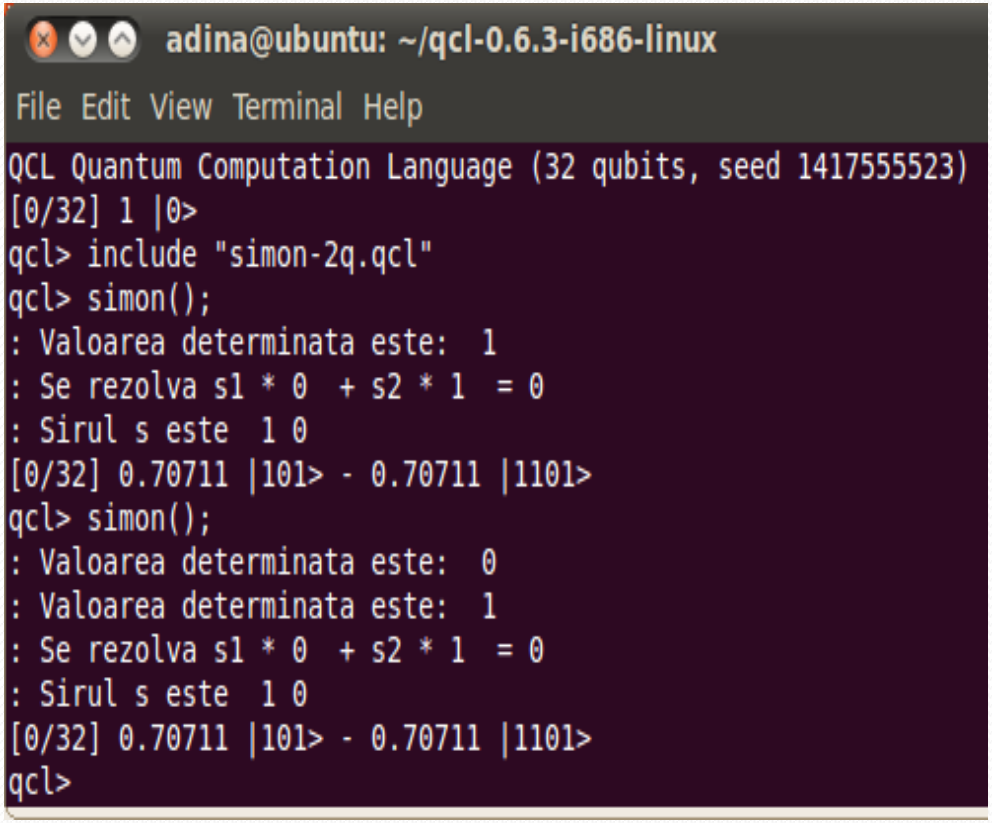
Transformarea U_f este următoarea:



Algoritmul lui Simon pentru n=2

```
//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);
```

Algoritmul lui Simon pentru n=2



http://eed.usv.ro/SistemeDistribuite/index.php?shift=home

.:: Seminarul Sisteme Distrib...

File Edit View Favorites Tools Help

abouttabs Suggested Sites Web Slice Gallery

seminarul ştiinţific

SISTEME DISTRIBUITE

EDITIA 2014
EDITIA 2013
EDITIA 2012
EDITIA 2011
EDITIA 2010
EDITIA 2009
EDITIA 2008

Google

Go

CCSC
CENTRUL DE CERCETARE STIINTA CALCULATOARELOR

Seminarul ştiinţific cu participare naţională

Sisteme Distribuite

Ediţia a XII-a, Suceava, 17 decembrie 2014

Tematica seminarului:
- Calcul paralel si distribuit
- Inteligenta artificiala distribuita
- Sisteme industriale distribuite
- Baze de date distribuite
- Recunoasterea formelor si procesarea imaginilor
- Sisteme informatice educationale
- Tehnologii web
- Tehnologii informationale

Seminarul se adreseaza in primul rand tinerilor cercetatori si cercetatorilor in curs de devenire: doctoranzi si masteranzi.

Pot fi trimise spre publicare articole de specialitate, lucrari de cercetare, studii de caz cu caracter

seminarul ştiinţific

SISTEME DISTRIBUITE

EDITIA 2014
EDITIA 2013
EDITIA 2012
EDITIA 2011
EDITIA 2010
EDITIA 2009
EDITIA 2008

CENTRUL DE CERCETARE STIINTA CALCULATOARELOR

Chairmen:
Vasile Gheorghita GAITAN
Cristina Elena TURCU
Stefan-Gheorghe PENTIUC

Lucian Andries, Vasile Gheorghita GAITAN
Overview of a Microcontroller with a hardware Scheduler

Adina BARILA
Implementation of quantum algorithms. A study case

Ioan UNGUREAN, Nicoleta Cristina GAITAN
A solution to integrate de Internet of Things concept in a SCADA application

Nicoleta Cristina GAITAN
An overview regarding the improvement of the nMPRA architecture

Adrian POPOVICI, Ioan UNGUREAN
An Architecture for the Industrial Internet of Things

Ionel ZAGAN, Vasile Gheorghita GAITAN
CPU architecture description based on fine-grained multithreading and hardware scheduler engine

Catalin LUPU, Valeriu LUPU
Securing internet-banking applications by using biometrics