QCL IMPLEMENTATION OF THE BERNSTEIN-VAZIRANI ALGORITHM

BY

SIMONA ARUŞTEI* and VASILE MANTA*

Abstract. In this paper we present an implementation in the quantum computer simulator QCL of the Bernstein-Vazirani algorithm. The analysis of this algorithm is made using the formalism of quantum gates. This formalism allows the decomposition of the computational process into elementary operations for an adequate hardware. The control gate used in the description of the algorithm is simulated by a set of 2-qubit CNot elementary quantum gates. We give an example by considering the case of a 5-dimensional input register.

Key words: quantum computing, quantum gate, quantum algorithm

2000 Mathematics Subject Classification: 81P68, 68Q05

1. Introduction

At present, the processing and the communication of the information is totally based on the phenomena of the classical physics. In the process of building current computers the quantum effects are taken into consideration for diminishing these effects and obtaining a certain macroscopic behaviour. However, before the year 2020, miniaturization will reach subatomic scales, and quantum phenomena will drastically affect the behaviour of semiconductors and microchips.

The research in quantum informatics appeared as a consequence of Richard Feynman's suggestions, who in 1982 suggested that quantum phenomena can be used in conceiving quantum computers. The application of the principles of quantum physics in the computer area led to the concept of quantum computer, in which the data aren't stored in bits like in the conventional memory, but as a combined state of several systems with 2-qubits states [2, 3]. According to the principles of quantum physics the computing power of a quantum machine is

immense compared to the one of a classic computer because, unlike the classic computers, in a quantum register all states can be accessed simultaneously (at least theoretically) [2, 3, 7].

Feynman's idea wasn't immediately embraced because not even from the theoretical point of view was it clear how such a quantum computer would operate. The research in quantum informatics has suddenly gained immense interest when in 1994 Peter Shor [9] invented an algorithm for a quantum computing for solving the problem of the factorization of a whole number in polynomial time (currently in classic informatics there are only algorithms with exponential execution time for this problem). This algorithm was improved in 1999 by Richard Hughes from Los Alamos National Laboratory. The importance of this algorithm is crucial because the main feature that most security techniques (RSA system with public key) are based on, is the difficulty of the factorization of a number using a polynomial algorithm.

Right after the presentation of the factorization algorithm new quantum algorithms appeared: computing the logarithm of a natural number (Shor 1994), another factorization algorithm (Jozsa 1997), mean calculus algorithms (Grover 1997), database search algorithm (Grover 1999), etc. [1, 4]. Despite all these, currently there are few quantum algorithms with a time complexity smaller than the classical case [3]. The known algorithms can be divided in three groups. The first group is formed by the algorithms based on a common property of the result values, for example the period of a function (Shor's algorithm). The second group contains algorithms for which a transformation of the state comes with an increase of the probability of obtaining a certain result (amplification) (Grover algorithm). The third class contains algorithms with characteristics of both previous classes (the approximate counting algorithm).

2. Quantum Computation – Basic Concepts

Just like a classical computer, a quantum computer is built out of three main parts: processor, memory and input/output. A quantum computer can be formally described by $M = (\mathrm{H}, \mathrm{O}, \mathrm{T}, \delta, \beta)$, where H - represents the states space (C^{2^n} Hilbert space) of the quantum system, O- the set of unitary transformations, T- the set of measurement commands, δ - is an initialization operator and β - describes the final measurement.

The quantum analogous of the classical bit is the qubit. A qubit is a quantum system whose states can be completely described by the superposition of two orthonormal basis states, labelled $|0\rangle$ and $|1\rangle$ (in a Hilbert space $\mathrm{H} = \mathrm{C}^2$, $|0\rangle = (1 \quad 0)^T$, $|1\rangle = (0 \quad 1)^T$). Any state $|\psi\rangle$ can be described by:

(1) $$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \quad |\alpha|^2 + |\beta|^2 = 1 .$$

The orthonormal system $\{|0\rangle, |1\rangle\}$ is called computational basis. The machine state $|\psi\rangle$ of an n-qubit quantum computer is an unit vector in Hilbert space $\mathrm{H} = C^{2^n}$:

(2) $$|\psi\rangle = \sum_{d_{n-1}\ldots d_0} c_{d_{n-1}\cdots d_0} |d_{n-1}\cdots d_0\rangle, \ \sum \left| c_{d_{n-1}\ldots d_0} \right|^2 = 1$$

The basis vectors $|d_{n-1}\ldots d_0\rangle$ can be interpreted as binary numbers and relabelled as:

(3) $$|k\rangle = \sum_{i=0}^{n-1} 2^i d_i$$

A quantum register **s** is represented by a sequence of qubits. If **s** is an n-qubit quantum register and U is an operator in the states space H, then the operator $U(\mathbf{s})$ applied to a register is called a quantum gate.

The Hadamard (H) and the NOT (X) gates are examples of quantum gates that act on a single qubit:

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

For the X gate we have: $X : |x\rangle \to |\bar{x}\rangle$. The Hadamard gate is useful because applying it to either of the basis states $|0\rangle$ and $|1\rangle$ produces an equal mixture of both of them:

(5) $$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

and

(6) $$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) .$$

The most important 2-qubit gate is the *CNOT* (controlled-not gate) which operates as: $CNOT : |x\rangle_1 |y\rangle_1 \to |x\rangle_1 |x \oplus y\rangle_1$. The matrix form of this gate is:

(7) $$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

$CNOT$ is a generalization of the classical XOR gate, since its action may be summarized as $|x, y\rangle \to |x, y \oplus x\rangle$, where $\oplus$ is addition modulo two, which is the same as XOR.

Generally, if U is a unitary m-qubit gate, a controlled U-gate with n control qubits is defined as:

(8) $$C^m[U] = \begin{pmatrix} I & \cdots & 0 & 0 \\ \vdots & \ddots & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & U \end{pmatrix}.$$

We remark that all quantum gates are unitary operators.

3. Bernstein-Vazirani Algorithm

Let a be an unknown non-negative integer less than 2^n. The binary representation of a using n bits is $a_{n-1}a_{n-2}\cdots a_0$. Let $f(x)$ be a Boolean function $f : \{0,1\}^{\otimes n} \to \{0,1\}$ defined as

(9) $$f(x) = a \cdot x = a_{n-1}x_{n-1} \oplus a_{n-2}x_{n-2} \oplus \cdots \oplus a_0 x_0$$

The function f actually calculates the binary inner product of a and another such number x. Suppose we have a subroutine that evaluates $f(x)$. The Bernstein-Vazirani problem consists in determining all the bits of a with a single invocation of the subroutine.

The mth bit of a is $a \cdot 2^m$, since the binary expansion of 2^m has 1 in position m and 0 in all the other positions. On a classical computer the n bits of a are determined applying f to the n values $x = 2^m$, $0 \le m < n$. Thus, any classical algorithm requires n calls of the subroutine that evaluates $f(x)$. On the other hand, on a quantum computer, a single invocation is enough to determine a completely, regardless of how big n is.

In the following we describe the algorithm using quantum gates (Hadamard gate H, NOT gate X). We consider a n-qubit quantum register $|x\rangle_n$ and a 1-

qubit quantum register $|y\rangle_1$. Analogous to Deutsch algorithm [2] the control gate U_f is built using function f:

$$U_f : |x\rangle_n |y\rangle_1 \to |x\rangle_n |y \oplus f(x)\rangle_1 \tag{10}$$

The unitary operator U_f acts on $n+1$ qubits and is also called an oracle operator. The schematic representation of the unitary transformation U_f for evaluation of the function f is depicted in Fig. 1.

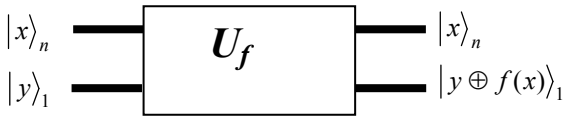

Fig.1 – The action of unitary transformation U_f

The 1-qubit register is also called output register, while the n-qubit register is called input register. The initial state of the quantum registers is $|\psi_0\rangle = |0\rangle_n |0\rangle_1$. The 1-qubit register is passed in the $|1\rangle_1$ state using the NOT operator, $X|0\rangle_1 = |1\rangle_1$ and then a superposition of states is acquired using the Hadamard gate:

$$HX|0\rangle_1 = H|1\rangle_1 = \frac{1}{\sqrt{2}}\left(|0\rangle_1 - |1\rangle_1\right) \tag{11}$$

U_f applied to the computational basis states $|x\rangle_n |y\rangle_1$ flips the value y of the output register if and only if $f(x) = 1$, and the resulted state is:

$$|\psi_1\rangle = U_f |x\rangle_n \frac{1}{\sqrt{2}}\left(|0\rangle_1 - |1\rangle_1\right) = (-1)^{f(x)} |x\rangle_n \frac{1}{\sqrt{2}}\left(|0\rangle_1 - |1\rangle_1\right). \tag{12}$$

Relation (12) uses the quantum parallelism of states, so, by taking the state of the 1-qubit output register to $H|1\rangle_1$, we convert a bit flip to an overall change of sign.

The action of H on a single qubit can be compactly summarized as:

$$H|x\rangle_1 = \frac{1}{\sqrt{2}}\left(|0\rangle_1 + (-1)^x |1\rangle_1\right) = \frac{1}{\sqrt{2}} \sum_{y=0}^{1} (-1)^{xy} |y\rangle_1 \tag{13}$$

The operator H, which acts on a single qubit, can be generalized on n qubits like in the following [2]:

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

where the product $x \cdot y$ is defined as in (9). Because -1 is raised to the power $\sum x_i y_j$, all that matters about that sum is its value modulo 2.

Thus, if the n-qubit input register is in the initial state $H^{\otimes n}|0\rangle_n$ and the 1-qubit register is passed in the state $H|1\rangle_1$, applying U_f and then $H^{\otimes n}$ to the input register, it results in:

(15) $$\left(H^{\otimes n} \otimes \mathbf{1}\right) U_f \left(H^{\otimes n} \otimes H\right) |0\rangle_n |1\rangle_1 =$$
$$\frac{1}{2^n} \sum_{x=0}^{2^n-1} \sum_{y=0}^{2^n-1} (-1)^{f(x)+x\cdot y} |y\rangle_n \frac{1}{\sqrt{2}} \left(|0\rangle_1 - |1\rangle_1\right)$$

The sum over x can be done first. The function $f(x)$ being $a \cdot x$, it produces the factor:

(16) $$\sum_{x=0}^{2^n-1} (-1)^{(a\cdot x)} (-1)^{(y\cdot x)} = \prod_{j=1}^{n} \sum_{x_j=0}^{1} (-1)^{(a_j + y_j) x_j} .$$

If there is at least one bit y_j of y different from the corresponding bit a_j of a, i.e. if $y \neq a$, at least one term in the product from equation (16) vanishes. Therefore the entire computational process (15) reduces to:

(17) $$H^{\otimes(n+1)} U_f H^{\otimes(n+1)} |0\rangle_n |1\rangle_1 = |a\rangle_n |1\rangle_1 .$$

where a final H is applied to the 1-qubit output register to make the resulting expression look a little neater and more symmetric (using the fact that $H^2 = I$).

So by putting the input and output registers into the appropriate initial states, after a single invocation of the subroutine followed by an application of $H^{\otimes n}$ to the input register, the state of the input register becomes $|a\rangle$. All n bits of the number a can now be determined by measuring the input register, even though we have called the subroutine only once.

Here we have to make the followings observations: A measurement is described by a self-adjoint operator M, which has the spectral decomposition $M = \sum_m m P_m$, where P_m is the projector onto the eigenspace of the eigenvalue m. The eigenvalues m correspond to the possible outcomes of measurement. Measuring $|\psi\rangle$ will give the result m with probability:

(18) $$p(m) = \langle \psi | P_m | \psi \rangle .$$

thereby reducing $|\psi\rangle$ to the post-measurement state:

(19) $$|\psi'\rangle = \frac{1}{\sqrt{p(m)}} P_m |\psi\rangle .$$

So when the measurement is effected the system must find itself in a state of the computational basis and not in a superposition of such states.

In Fig. 2 we present the circuit equivalent with relation (17) for the case where $n = 5$ and $a = 19 = 10011$. The 5-qubit input register in state $|0\rangle_5 \equiv |0000\rangle$ along with the output register in state $|1\rangle_1$ make up a register of dimension $n+1 = 6$. The 6-qubit Hadamard transform is applied to this register, followed by U_f and then another Hadamard transform on 6-qubit. The action of U_f is reproduced by a set of CNot (controlled-not) gates.

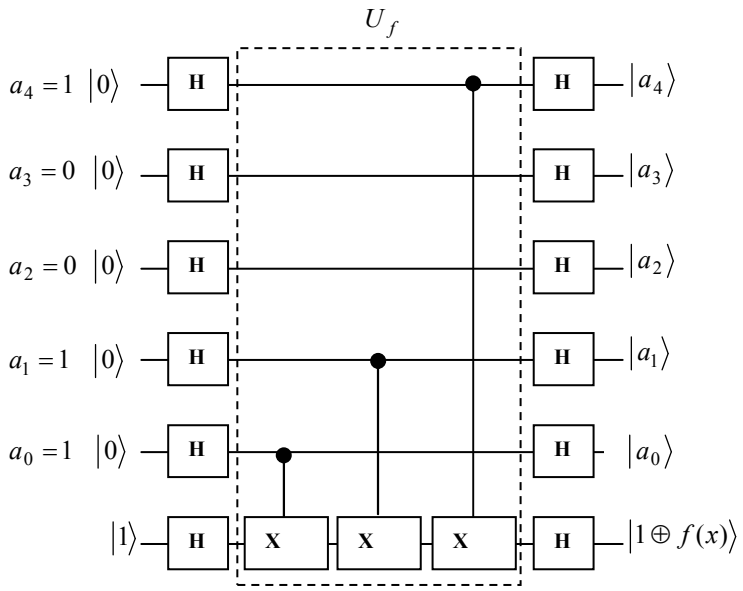

Fig. 2 – The quantum circuit for Bernstein-Vazirani problem in the case $n = 5$

4. The QCL Implementation

The programming language QCL (Quantum Computation Language) is a quantum simulator. It was conceived by Omer [5,6] and the first version

appeared in 1998 and the last one in 2004. It is open-source and it runs under Linux operating system. QCL is a procedural high level language and has a C like syntax. The main features are:

- a classical control language with functions, flow-control, interactive I/O and various classical data types (int, real, complex, boolean, string);
- 2 quantum operator types: general unitarian (operator) and reversible pseudo-classic gates (qufunct);
- inverse execution, allowing for on-the-fly determination of the inverse operator though caching of operator calls;
- various quantum data types (qubit registers) for compile time information on access modes (qureg, quconst, quvoid, quscratch);
- convenient functions to manipulate quantum registers (q[n] - qubit, q[n:m] - substring, q&p - combined register;
- automatic scratch space management;
- universal language: can implement and simulate all known quantum algorithms.

Below we list the implementation of relation (17) in QCL.

```
/* The Bernstein-Vazirani algorithm for a register x
with 5 qubits */
   qureg x[5];  qureg y[1];  int m;  int i;
   int vector bitv[5];
   for i=0 to #x-1 {
      if random()>0.5 {bitv[i]=1;}
   }
   int a=0;
   for i=0 to #x-1 {
      a = a + bitv[i]*2^i;
   }
   print "Chosen value for a is", a;
/* here we pass y from state |0> to state |1> */
   X(y);
/* now we concatenate the registers and apply the
Hadamard operator to all registers*/
   H(x&y);
/* we apply the Uf operator (we simulate the action
of Uf with CNot gates)*/
   for i=0 to #x-1{
      if (bitv[i]>0){ CNot(y,x[i]);}
   }
/* we concatenate the registers and apply the
Hadamard operator to all registers*/
   H(x&y);
// finally, we measure the register x
```

```
measure x, m;
print "Determined value of a is",m;
```

4. Conclusions

In this paper we have considered the Bernstein-Vazirani algorithm. The analysis of this algorithm has been described from the software engineering point of view using the formalism of quantum gates. Using this analysis the final state was obtained exploring the effect of the Hadamard gates on the initial state of the qubits and on the state subsequently produced by the action of U_f. For a particular case we have realized the quantum circuit for determination of a value. Finally, we have developed an implementation of this algorithm using the quantum language simulator QCL. The control gate U_f used in the description of the algorithm was simulated by a set of 2-qubits CNot elementary quantum gates.

Submitted:
Accepted:

**"Gh. Asachi" Technical University,*
Department of Computer Engineering
Iaşi, Romania,
e-mail:{sarustei, vmanta}@cs.tuiasi.ro

R E F E R E N C E S

1. C o l e J. H., H o l l e n b e r g L. C. L., P r a w e r S., *An Algorithm for Simulating the Ising Model on a Type-II Quantum Computer*. Computer Physics Communications, **161**(**1-2**), 18-26, (**2004**).
2. M e r m i n D., *Lectures Notes on Quantum Computer*. Cornell University, Ithaca, New York, **2006**.
3. N i e l s e n M., C h u a n g I., *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, UK, **2000**.
4. N i s h i m u r a H., Y a m a k a m i T., *Polynomial Time Quantum Computation with Advice*. Information Processing Letters, **90**, 195-204, (**2004**).
5. O m e r B., *Quantum Programming in QCL*. Technical University of Vienna, Vienna, Austria, **2000**.
6. O m e r B., *Structured Quantum Programming in QCL*. Technical University of Vienna, Vienna, Austria, **2003**.
7. P r e s k i l l J., *Lectures Notes for Physics - Quantum Information and Computation*. California Institute of Technology, California, USA, **1998**.
8. D e R a e d t H., M i c h i e l s e n K., *Computational Methods for Simulating Quantum Computers*. arXiv:quant-ph/0406210, **2004**.
9. S h o r P.W., *Algorithms for quantum computation: Discrete logarithms and factoring*. Proc. 35th Annual Symposium on Foundations of Computer Science, Los Alamitos, CA, USA, November, 20-22, **1994**, pp. 124-134.

IMPLEMENTARE QCL A LGORITMULUI BERNSTEIN-VAZIRANI

(Rezumat)

În această lucrare este prezentată implementarea algoritmului Bernstein-Vazirani în simulatorul de calcule cuantice, QCL. Analiza algoritmului a fost realizată pe baza formalismului porților cuantice. Acest formalism permite descompunerea procesului de calcul în operații elementare pentru un hardware adecvat. Poarta de control utilizata în descrierea algoritmului este simulată prin intermediul unui set de porți cuantice elementare CNot pe 2 qubiți. Se exemplifică aplicarea algoritmului considerând cazul unui registru de intrare pe 5 qubiți și se prezintă circuitul cuantic echivalent procesului de calcul.