Quantum parity algorithms as oracle calls, and application in Grover Database search

M. Z. Rashad
Faculty of Computers and Information sciences,
Mansoura University, Egypt
Magdi_z2011@yahoo.com

Abstract

This paper discusses the determination of the parity of a string of N binary digits, the well-known problem in classical as well as quantum information processing. It can be formulated as an oracle problem. It has been established that quantum algorithms require at least N /2 oracle calls. We present an algorithm that reaches this lower bound and is also optimal in terms of additional gate operations required.

***Keywords**: parity, quantum computing, oracle.*

1 Introducing Quantum Computing

Quantum computing is a promising approach of computation that is based on equations from Quantum Mechanics.

A Bit is the basic computational unit of computing. It encodes a 0 or a 1. A register of *n* bits can store **ANY** *n*-bit number. A *qubit* (quantum *bit*) exists in a *superposition* of states, and encodes the values 1 and 0 simultaneously. A quantum register of *n* qubits stores **ALL** *n*-bit numbers, i.e. 2^n values[1].

Quantum State

The quantum state $|\psi\rangle$ represents a qubit if there are $\alpha, \beta \in C$, where C is the set of Complex numbers, such that

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \qquad \text{or}$$
$$|\psi\rangle = \sin\theta\,|0\rangle + \cos\theta\,|1\rangle$$

With $|\alpha|^2 + |\beta|^2 = 1$. $|0\rangle$ and $|1\rangle$ are the computational basis states. Measuring the state $|\psi\rangle$ with respect to $\{|0\rangle, |1\rangle\}$ basis will give $|0\rangle$ with probability $|\alpha|^2$ and $|1\rangle$ with probability $|\beta|^2$. The states $|+\rangle$ and $|-\rangle$ defined as

$$|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$
$$|-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$$

Matrix notation

A 2-level quantum system can store a single qubit state .We will have

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

Also, we can say that: $|0\rangle = 01$ and $|1\rangle = 10$. That is $|x\rangle$ = binary (x + 1).

The symbol $|.\rangle$ is called a ***ket***, while the symbol $\langle .|$ is called a ***bra***. $\langle \psi|$ represents the conjugate transpose of $|\psi\rangle$

$$|\psi\rangle = \begin{pmatrix} z1 \\ \vdots \\ zn \end{pmatrix} \qquad \langle \psi| = \begin{pmatrix} z1 \\ \vdots \\ zn \end{pmatrix}^{t} = \left(\overline{z1}, \ldots, \overline{zn}\right)$$

where $z_1, \ldots, z_n \in$ Complex.

Writing $|1\rangle\langle 0| + |0\rangle\langle 1|$ means mapping $|1\rangle$ to $\langle 0|$ and $|0\rangle$ to $\langle 1|$. Note that $|\psi\rangle\langle\psi| = 1$.

Combing Qubits

Let A and B be quantum systems with state spaces H_A and H_B , the state space of the joint quantum system is

$$H_A \otimes H_B$$

For two qubits,

$$|00\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad |01\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \quad |10\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$|11\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

This is made by **tensor product**

$$\begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} ac \\ ad \\ bc \\ bd \end{pmatrix}$$

This means that :

- An operation on a single qubit will in general affect all coefficients of the joint state vector.
- A single qubit operation is highly parallel operation.
- Adding a single quantum bit doubles the memory

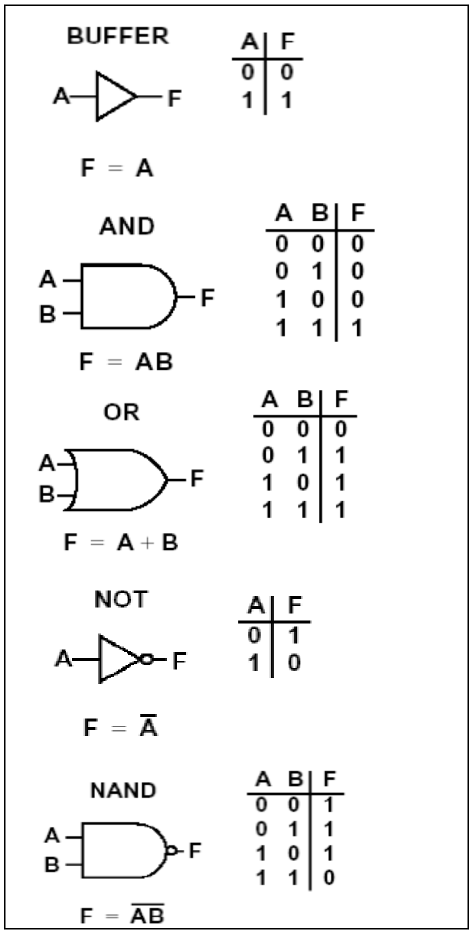
A note about operations on a Quantum Computer is that apart from measurements (Input/Output Operations) any quantum operation *U* is *Linear*, *Length-preserving* (i.e. input vectors and output vectors have the same number of components), *Unitary* (i.e. $UU^{t} = I$, where *I* is the identity matrix), *Reversible*, and *Deterministic* .

For detailed discussions on quantum computing and information, you can selectively refer to ([2], [3], [4]).

2. Simulating Quantum Algorithms.

Any simulator for quantum algorithms must be capable of performing heavy mathematical matrix transforms. The design of the simulator itself usually takes the form of circuit model of connected gates.

The circuit model of computation is equivalent to Turing machine model but is nearer to real computers ([2]). The building blocks that can build any circuit are called logic gates. The functionality of logic gates is described in terms of truth table, which specifies all possible configurations of input and the corresponding output. The elementary logic gates with their truth tables are given below, where A and B denote inputs and F denotes output.



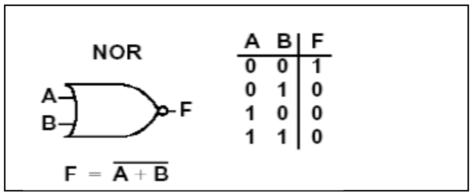
A	F
0	0
1	1

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

A	F
0	1
1	0

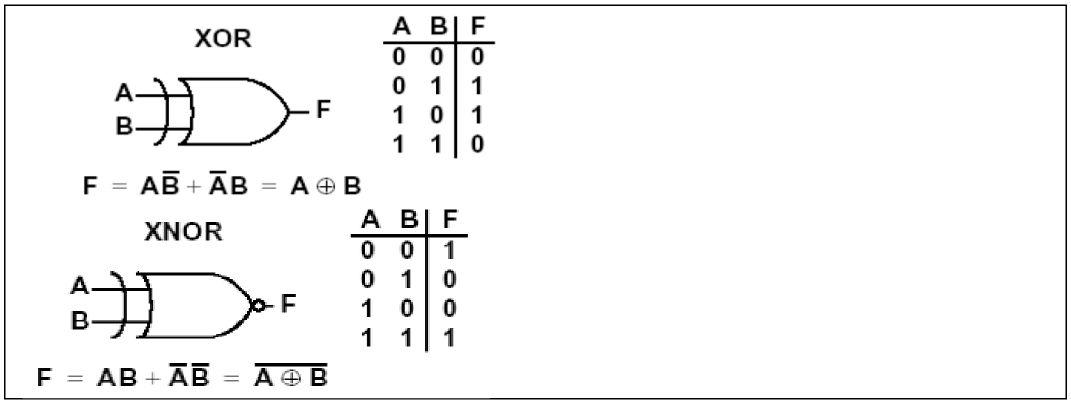
A	B	F
0	0	1
0	1	1
1	0	1
1	1	0



A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

(Figure 2: elementary classical logic gates)

A set of connected gates accomplishing a certain computation is called a circuit. A gate called XOR (*exclusive OR*) and its inverse are given below. Actually, XOR is a common used circuit, so we handle it as if it is an elementary gate.


XOR
A
B
F
F = $A\overline{B} + \overline{A}B = A \oplus B$
XNOR
A
B
F
F = $AB + \overline{A}\,\overline{B} = \overline{A \oplus B}$

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

(Figure 3: another two elementary logic gates)

Note that: the NOT gate is reversible (as you can guess the input for any given output), while the AND gate is irreversible , so we can say that the AND gate erases information.

Quantum gates are the same as the classical ones, but with maintaining the *Reversibility* and *Length-preserving* by usually outputting extra bits (*ancilla bits*) which usually correspond to a sufficient number of the inputs. For example, the quantum XOR gate takes two inputs, *x* and *y* say , and outputs *x* and the main output that is **x ⊕y**.

|x, y⟩ → |x, x ⊕ y⟩

When studying quantum gates we usually classify them according to the number of qubits the gate operates on into 1D (1 dimensional), 2D, and 3D. The common 1D quantum gates are:

$$Identity = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad NOT = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

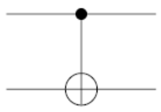
$$Hadamard = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$Phase\_Flip = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

$$\sqrt{NOT} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$$

The common 2D quantum gates are *CNOT* and *SWAP*. The *CNOT* (**Controlled-Not**) is quantum version of the XOR operation and it indicates the interaction between two qubits. This gate has many representations besides

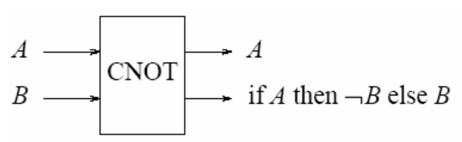
|x, y⟩ → |x, x ⊕ y

I/O transform	Gate
in → **out** $\|00\rangle \mapsto \|00\rangle$ $\|01\rangle \mapsto \|01\rangle$ $\|10\rangle \mapsto \|11\rangle$ $\|11\rangle \mapsto \|10\rangle$	 first input is the control (black)

Matrix
$\begin{pmatrix} 1 & . & . & . \\ . & 1 & . & . \\ . & . & . & 1 \\ . & . & 1 & . \end{pmatrix}$

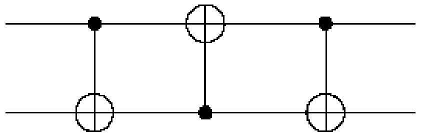
(Figure 4: Different representations of CNOT gate)

Block diagram and function



If first qubit is set then, apply NOT on the second qubit, else do nothing

The *SWAP* gate can be derived from successive *CNOT* gates.



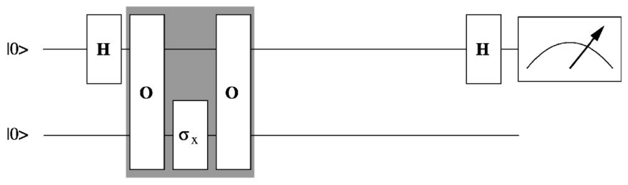
in		gate 1		gate 2		gate 3		out
$\|00\rangle$	$\mapsto$	$\|00\rangle$	$\mapsto$	$\|00\rangle$	$\mapsto$	$\|00\rangle$	$=$	$\|00\rangle$
$\|01\rangle$	$\mapsto$	$\|01\rangle$	$\mapsto$	$\|11\rangle$	$\mapsto$	$\|10\rangle$	$=$	$\|10\rangle$
$\|10\rangle$	$\mapsto$	$\|11\rangle$	$\mapsto$	$\|01\rangle$	$\mapsto$	$\|01\rangle$	$=$	$\|01\rangle$
$\|11\rangle$	$\mapsto$	$\|10\rangle$	$\mapsto$	$\|10\rangle$	$\mapsto$	$\|11\rangle$	$=$	$\|11\rangle$

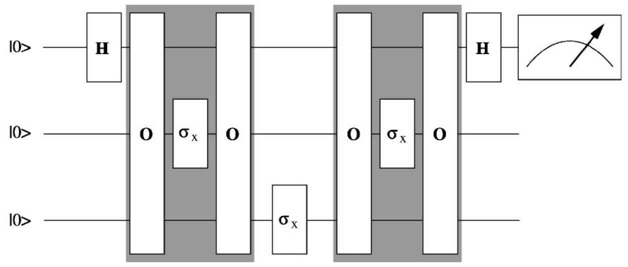
The common 3D quantum gates are *Toffoli* and *Fredkin* which may be considered as the controlled 3D versions of *CNOT* and *SWAP,* respectively. For more about classical and quantum circuit models, you can refer to ([6], [7]) and ([1], [3]) respectively.

3. Parity

Digital information processing relies on a number of error checking and correction algorithms. The most basic form of error detection checks the parity, which indicates if the number of 1's in a binary string is even or odd.

The number of computational steps required to determine the parity of a binary string increases linearly with the length of the string; this holds true for classical as well as for quantum information processors [19]. Quantum algorithms can reduce the number of steps required by a factor of 2 compared to classical algorithms [19], [20].





(Figure 5: Parity algorithm for n=2 qubits and n=3 bottom)

4. Grover Database search

Classically, searching a database of *N* elements for a certain element to specify its index requires *O(N)* comparisons, and $O\left(\frac{N}{2}\right)$ on average. Grover's algorithm achieves the same in $O\left(\sqrt{N}\right)$ ([18]). Figure 8 shows a sample database.

index	**value**
1	**Chicago**
2	**Cairo**
3	**Mansoura**
⋮	⋮

(figure 8: sample search database)

Grover's algorithm operates an *N*-element database, which sometimes is called Quantum Auto-associative Memory (QuAM). The basic idea of Grover's algorithm is to perform a check on all the elements several times and gradually increase the amplitude of the required state. After $O\left(\sqrt{N}\right)$ iterations, a measurement operation will give the correct answer with probability close to 1.

In the sample database, $|x\rangle$ and $|I(x)\rangle$ correspond to *indices* and *values* respectively. Unitary operators applied are:

$$\boldsymbol{U_1 = I - 2\,|w\rangle\langle w|}$$
$$\boldsymbol{U_2 = 2\,|p_0\rangle\langle p_0| - I}$$

where $|p_0\rangle$ is the initial state of the database and $|w\rangle$, a target state. When a target state $|w\rangle$ is 0, then

$$|p_0\rangle = \frac{1}{\sqrt{N}}\sum_{|x\rangle=0}^{N-1}|x\rangle$$

U_2 is a reverse transformation on the average, and is called a **Diffusion** operator. U_1 is a transformation of selective rotation on a target state, and is called an **Oracle** operator, with the following effect:

$$O|x\rangle = (-1)^{f(x)}|x\rangle$$

If x is the required element then the state $|x\rangle$ is rotated through π. Where $f(x)$ =1 if x is the target, and $f(x)$ =0 otherwise.

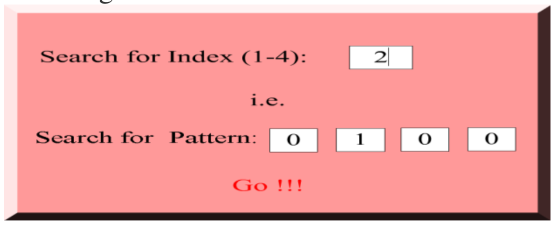
Tracing Grover's Algorithm

When it was the time to implement a case study utilizing the package, we decided to re-implement the most of the package, initially in VB.NET, into a web Flash object with ActionScript, and then embed it into a form as if it is built in VB.NET. This weird implementation is for many reasons:

- Challenging our programming skills.
- Better visuality, compared to command-line textual outputs
- Execution speed, compared to other web objects such as java applets.

The window for Grover's algorithm is shown in figure 9. At the first sight, the overall design of the window looks similar to that found in adopted Quasi. The window consists of two sky-blue panels with three buttons in between. The upper panel shows the circuit, with a blue indicator to track progress of execution which can be controlled using the *forward* and *backward* buttons. Each step is called a *stage*. The lower panel is for states.

The algorithm is restricted to the case of searching for an item out of four. We then need 2 (= $\log_2 4$) qubits and a third on for control. To specify the index of the desired item, use *restart* button which shows the following window:


Search for Index (1-4): 2

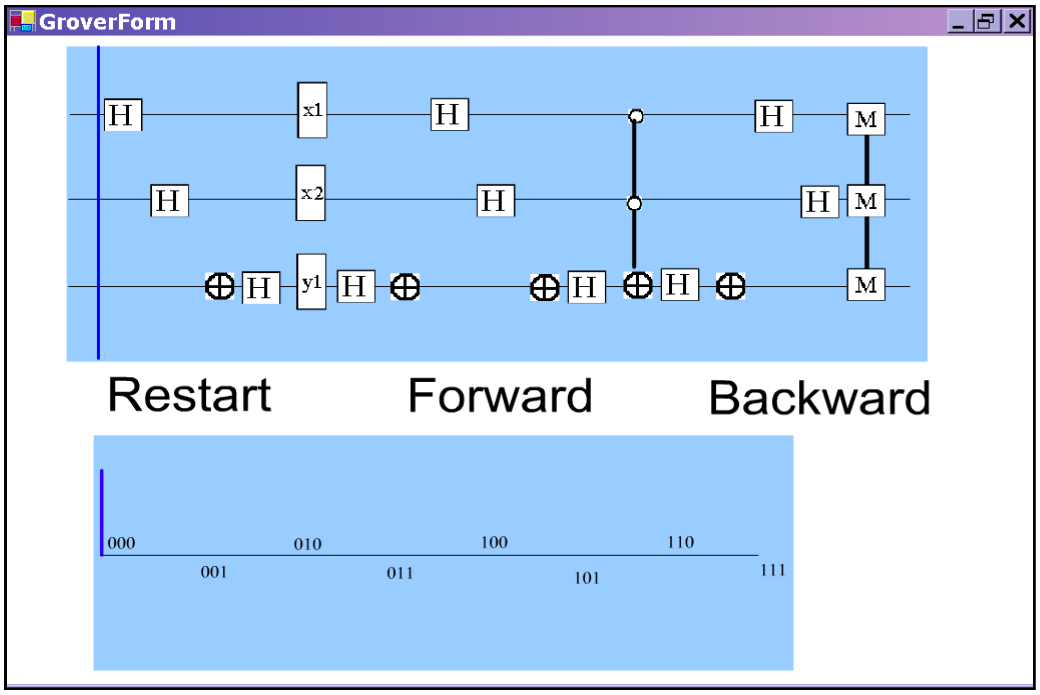
i.e.

Search for Pattern: 0 1 0 0

Go !!!

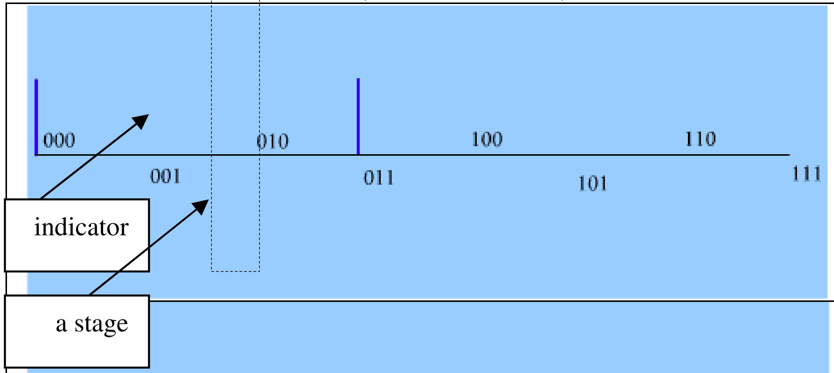
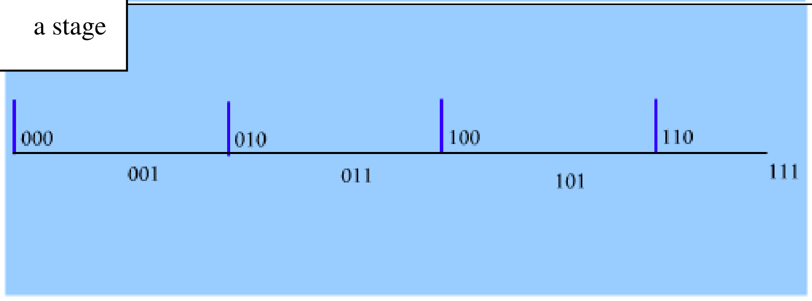
figure 10: specifying the index ()

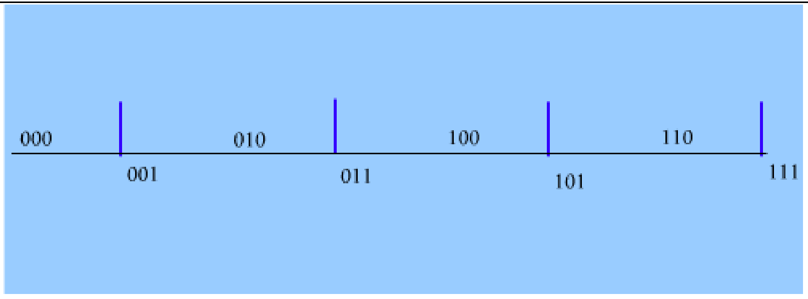
we have 16 stages and a final measure (M). A better look shows that we are applying one circuit twice each of 7 stages. We think that gate symbols are self describing as H for *hadamard* and ⊕ for *phase flip*. Stage 5 is the oracle. At the end, we will find the system in the required state.



(figure 9: Grover windows)

The initial state is (|000> 1.0 + 0.0i), and here are some steps of execution:

	**State after stage 1:** |000> 0.7071067811865475 + 0.0i |100> 0.7071067811865475 + 0.0i
	**State after stage 2:** |000> 0.4999999999999999 + 0.0i |010> 0.4999999999999999 + 0.0i |100> 0.4999999999999999 + 0.0i |110> 0.4999999999999999 + 0.0i

	**State after stage 3:** |001> 0.4999999999999999 + 0.0i |011> 0.4999999999999999 + 0.0i |101> 0.4999999999999999 + 0.0i |111> 0.4999999999999999 + 0.0i

(figure 11: first steps of execution of Grover's algorithm)

5. Results and Conclusion

Experimental results for the pure-state algorithm shown in Fig. 5, is shown below.

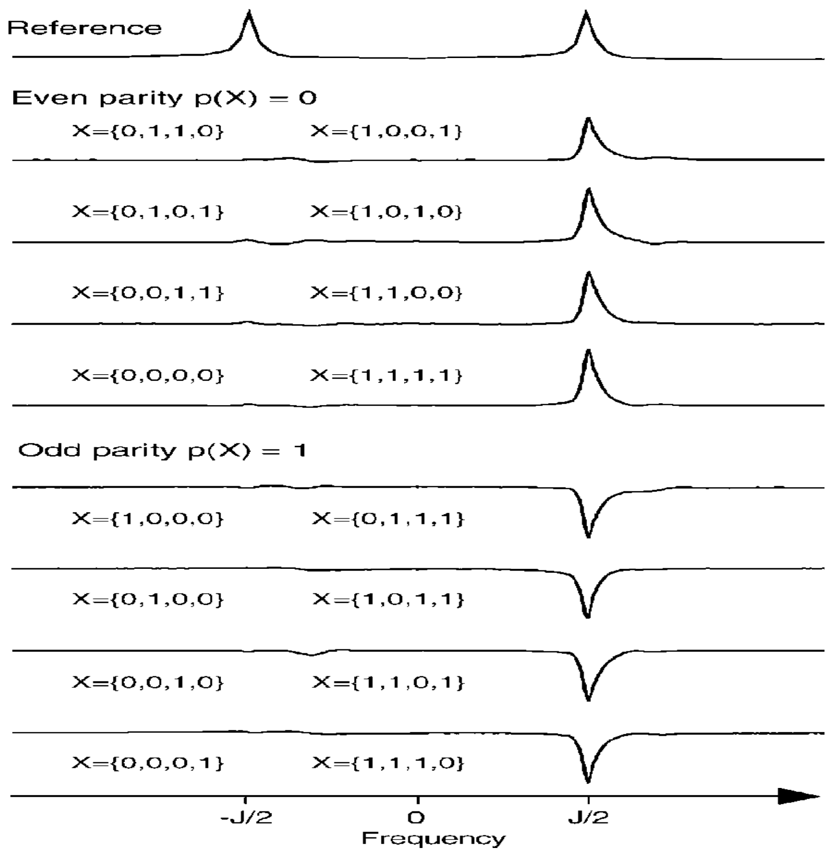


FIG. 12. Experimental results for the pure-state algorithm shown in Fig. 5.

We have introduced a family of quantum algorithms that solve the parity problem with an optimal number of quantum gate operations. It uses the black-box scheme introduced by Beals *et al.* [19] to represent the strings as oracle gates. In agreement with the lower bound established by Beals *et al.*, we can apply it after Grover search. The determination of the parity of a string of N binary digits can be formulated as an oracle problem. It has been established that quantum algorithms require at least N /2 oracle calls.

6. References

[1] Andreas Klappenecker, *"The Quantum Circuit Model", Quantum computing seminar,* Departments of Computer Science, Texas A&M University , 2001.

[2] Rob Pike, "*An Introduction to Quantum Computation and Quantum Communication*", Bell Labs, 2000

[3] G. Benenti, G. Casati, and G. Strini, "*Principles of Quantum Computation and Information, Volume 1: Basic Concepts*", World Scientific, 2004.

[4] M. Brocks, "*Quantum Computing and Communications*", Springer, 1999.

[5] N.J. Cutland, *Computability: an introduction to recursive function theory*, Cambridge University Press, 1980.

[6] M.M. Mano, *Computer System Architecture*, Prentice-Hall International, 3rd ed., 1993.

[7] M.M. Mano, *Digital Design*, Prentice-Hall International, 3rd ed, 2001.

[8] S. Gay, *Quantum Programming Languages: Survey and Bibliography*, Bulletin of the European Association for Theoretical Computer, June 2005.

[9] S. Bettelli, *Toward an architecture for quantum programming* PhD Thesis, Universita di Trento, 2002

[10] E. Knill, *Conventions for quantum pseudocode*, Los Alamos National Laboratory technical report, LAUR-96-2724, 1996.

[11] P. Maymin, *The lambda-q calculus can efficiently simulate quantum computers*, 1997. quant-ph/9702057

[12] A. van Tonder, *A Lambda Calculus for Quantum Computation*, SIAM J. Comput. 33, 1109-1135, 2004. quant-ph/0307150

[13] B. Ömer, *Quantum Programming in QCL*, Master thesis (technical physics), Technical University of Vienna, 2001.

[14] I. Glendinning, B. Ömer, *Parallelization of the QC-lib Quantum Computer Simulator Library*, Springer LNCS 3019, 2004.

[15] Sze Meng Tan, *A Quantum Optics Toolbox for Matlab 5*, University of Auckland, Private Bag 92019, Auckland

[16] QuaSi - Quantum Circuit Simulator, online version as java applet, at the site of the University of Karlsruhe

[17] Cameron Wakefield and Henk-Evert Sonder,*VB.NET Developer's Guide*, Syngress Publishing, 2001.

[18] Qcs - Quantum computer Simulator, online tutorial, at the site of www.senko-corp.co.jp

[19] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf, J. ACM **48**, 778 , 2001.

[20] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, Phys. Rev. Lett. **81**, 5442 1998.