Quantum Information and Computing-Quantum Circuits

Dipan Kumar Ghosh
Department of Physics
Indian Institute of Technology Bombay
Powai, Mumbai 400076

March 16, 2017

Introduction

In the last lecture, we talked about various quantum gates. There are some gates which may be considered as *Universal* in terms of which all boolean operations may be expressed. Some of these universal gate sets have been identified as follows:

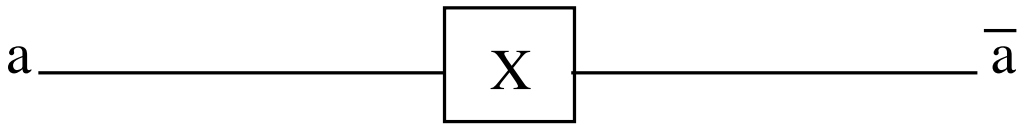
1. All single qubit gates and the CNOT gate
2. CNOT, Hadamard, T gate and X gate
3. Toffoli and Hadamard

In this lecture we will introduce the concept of a quantum circuit, which consists of having inputs, subjecting them to quantum gates, getting output and performing measurement of the output. The circuit consists of sequential application of quantum operations on quantum registers which, in general, contains linear combination of quantum states. In diagrams they are connected by straight lines looking similar to wires in a classical circuit. However, the diagram simply represents logic flow from left to right. The connectivity need not be by physical wires but may be connected by fiber links or by wireless connecting elements. Gates are represented by blocks with their symbols embedded in them. Unlike classical circuits, looping, i.e. feedback is not permitted, i.e the flow of logic is unidirectional. FAN-IN operations (where the gate takes in more than one input to give rise to a single output) is not permitted as such an operation would not be reversible. Likewise FAN-OUT (in which an output is connected to more than one load) is also not permitted as it would essentially imply copying of the same output, violating the No-Cloning Theorem. Unless expressly mentioned, the qubits are in computational basis.

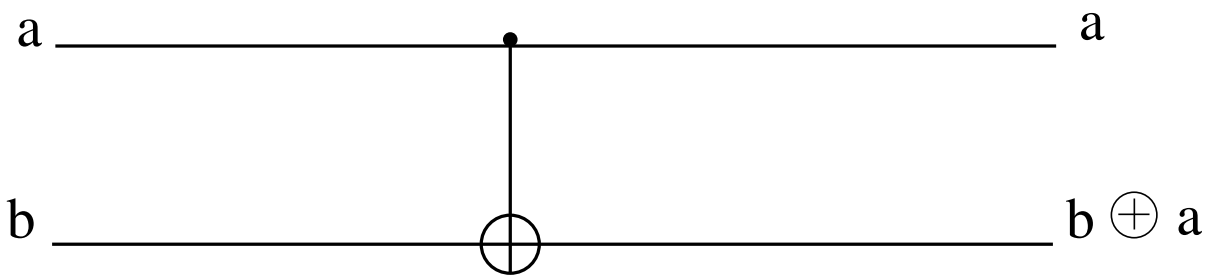
1 Implementation of classical logic gates

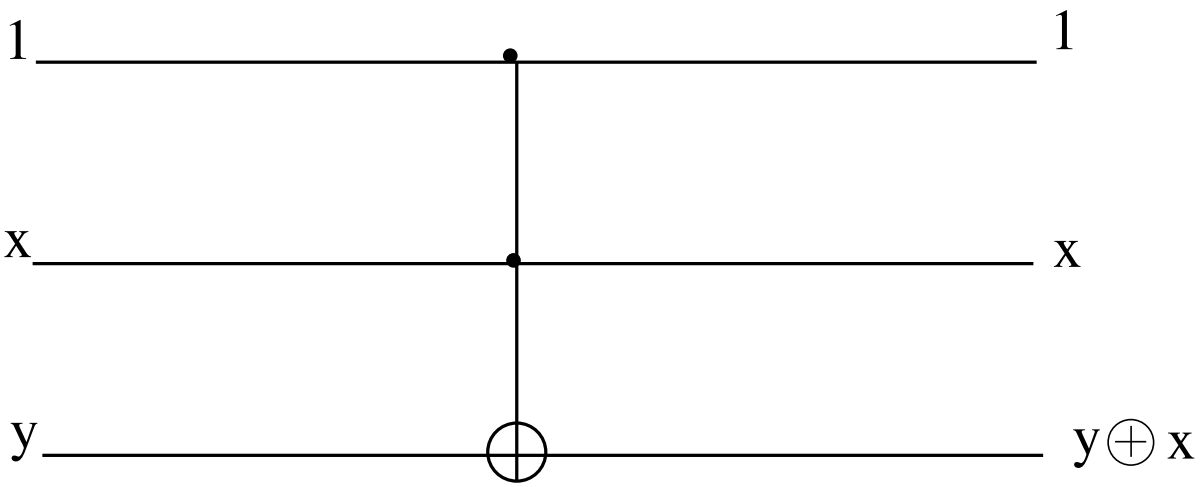
We will start by showing how to implement simple classical logic on a quantum computer. The NOT gate is simply simulated by an X-gate. XOR or addition modulo 2 is just the CNOT gate with the result of XOR for the two inputs (control and target) being output in the target register. Two alternate circuits for producing XOR in the following, one uses a CNOT while the other uses CCNOT gate.

NOT :



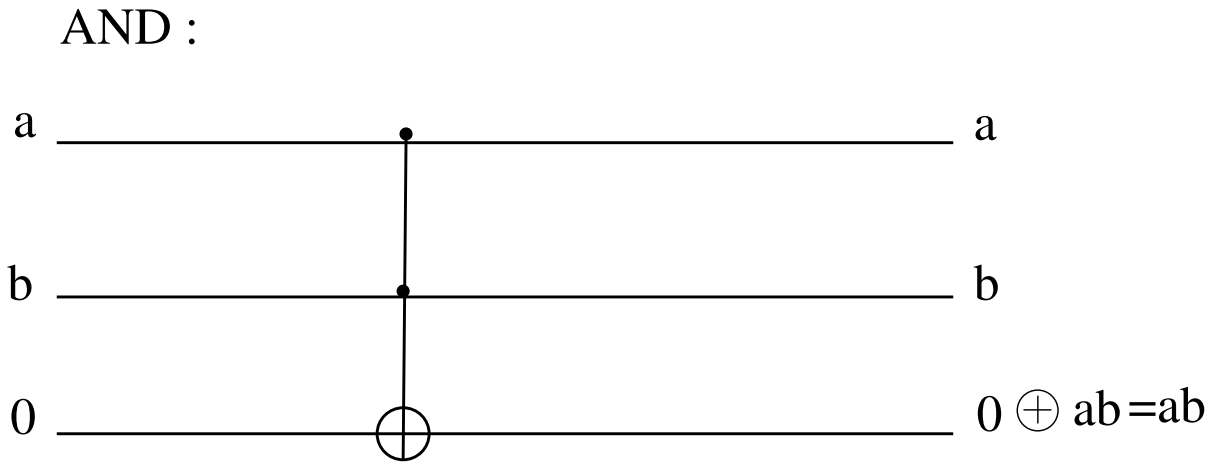
XOR :



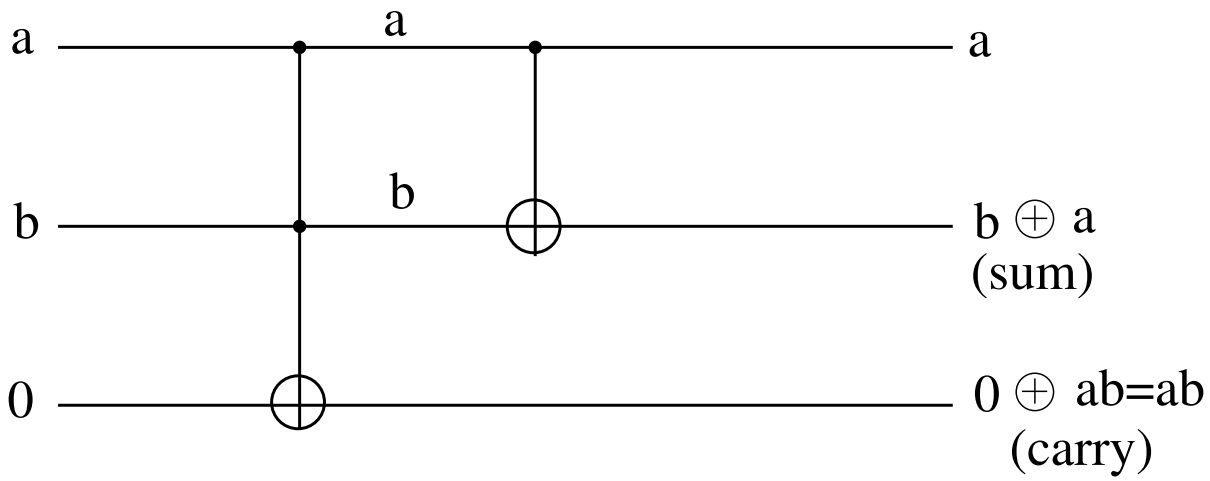


One can simulate the classical AND using a CCNOT gate where the two inputs are

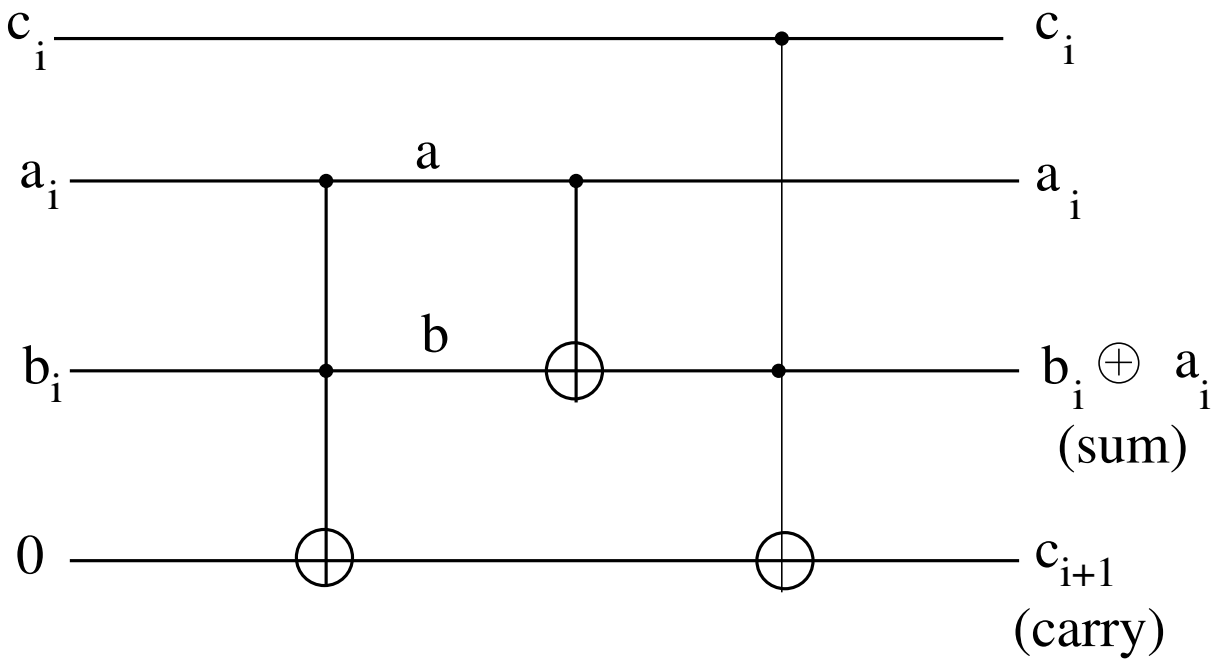
the two control bits. The target is initially set to bit 0 so that the result of the AND operation appears in the target register. The circuit is shown below.



As non-trivial examples, we give the quantum circuits for a Half-Adder and Full-Adder circuits. In case of the Half adder, two single bits are added with the result of addition being in one register and a possible carry, which arises only when each bit that is being added is 1, being stored in another register. The circuit uses a CCNOT gate and a CNOT gate. The target bit into which the result of carry is to be stored is initially set to 0. The sum is output into one of the control registers (one could use either of them by simply interchanging the target and the control of the CNOT gate).

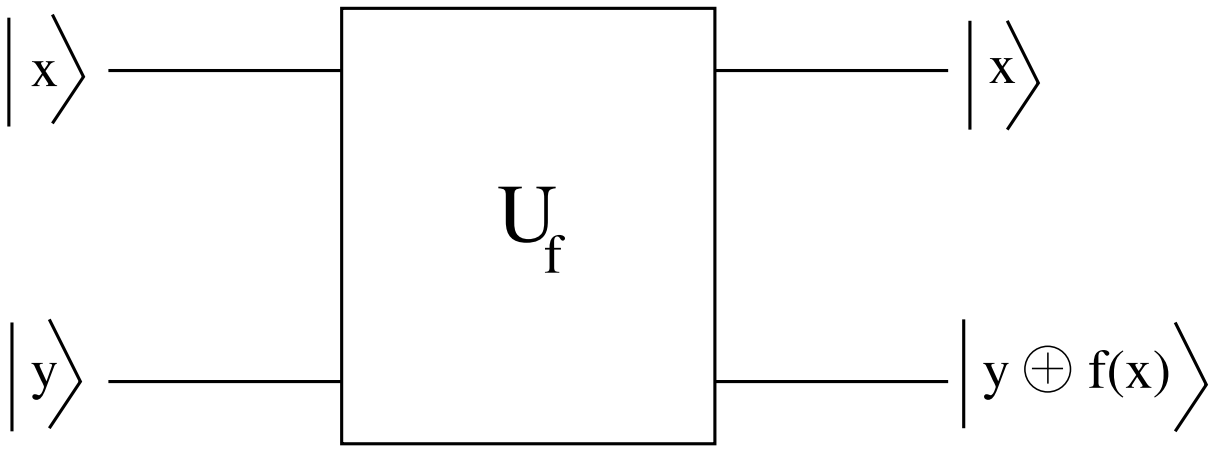


The FULL-Adder circuit has two bits a_i and b_i to be added along with the carry c_i from the previous step. The sum is stored against either of the first two bits while the carry to the next level c_{i+1} is stored in an ancilla bit which is initially set to 0.



2 Oracle

An essential component of quantum computing is known as an **Oracle**. An oracle is very similar to a subroutine in classical computers. In writing a program when we make a subroutine call, it is assumed that a program exists corresponding to the subroutine. In a similar way, in a computer the calculation of a function is done by an *oracle*, which has information about the unitary transformations required for computation of the function. It is like a black-box which takes in the data required for computation of the function as the input and outputs the value of the function to a target register which was initially set to $\mid 0\rangle$. There are some points to be noted. Firstly, because of parallelism the oracle can simultaneously compute the function for a large number of inputs. However, when one makes a measurement of the function, one would get a random output from among the functions that have been calculated. The input register would contain a linear combination all those states which correspond to the function which has been filtered out. Further, if the target register is set to 1, one can get the complement of the function instead of the function itself.



As an illustration, suppose we wish to calculate a particular function corresponding to any n-qubit input. One could start with an input register which is set to $\mid 00\ldots0\rangle = \mid 0\rangle^{\otimes n}$. When each of these null states is passed through a Hadamard gate, one would get a uniform linear combination of n qubit basis states as,

$$\begin{aligned}\mid 0\rangle^{\otimes n} \xrightarrow{H^{\otimes n}} &= \mid 0\rangle \otimes \mid 0\rangle \ldots \mid 0\rangle \xrightarrow{H^{\otimes n}} \frac{\mid 0\rangle + \mid 1\rangle}{\sqrt{2}} \otimes \frac{\mid 0\rangle + \mid 1\rangle}{\sqrt{2}} \ldots \frac{\mid 0\rangle + \mid 1\rangle}{\sqrt{2}} \\ &= \frac{1}{\sqrt{2^n}} \sum_x \mid x\rangle\end{aligned}$$

If this is fed to the oracle, the function will be simultaneously calculated for all the states of the n-qubit computational basis.