UNIVERSITY OF WASHINGTON

Quantum Computing for Mere Mortals

Craig M. Dowell

Department of Physics, University of Washington, Seattle, WA 98195-1560

Abstract

Reasonably concise and accessible descriptions of quantum computing tend to focus either on the concrete low level physics or the abstract computational aspects of the problem. This is understandable given the authors' interests. It is often the case that a physics-heavy paper may be incomprehensible to those not familiar with the field, and a computer science-heavy paper may not include any more interesting physics than an abstract overview of the first week of an undergraduate quantum mechanics course. This paper attempts to strike a middle ground where just enough physics and logic are presented to make that sometimes elusive connection between a relatively real implementation of a quantum computer and a real reduction in computational complexity observed by using quantum superposition of logical states in a quantum algorithm.

1. INTRODUCTION

When asked about computers and computing, the average person probably thinks about the ubiquitous laptop and desktop devices seen in almost every home and office over much of the planet. This was not always the case. In the middle of the twentieth century, a computer was a person. Although not explicitly called computers, devices have been used to perform tasks of one sort or another for thousands of years. As early as two thousand years ago, astrolabes were used to calculate the position of the moving planets.

During the middle of the twentieth century the U. S. Air Force's Norden bombsight and the Zeiss Lotfernrohr used by the German Luftwaffe were highly sophisticated examples of mechanical devices used to perform calculations. Today we would call these devices analog computers. Various kinds of mechanical calculators have existed since antiquity, the abacus being probably the most well known example. In the early 1800s Babbage proposed his Difference Engine which was able to tabulate polynomial functions and is widely considered a prototypical modern computer.

The useful work that all of these entities, both people and devices, perform is called computation. Computation is a generic term for any type of information processing. Information processing is, in its most general sense, a process that changes information in a way that is detectable by an observer. This is an extraordinarily broad definition that can include everything from your home computer to a falling rock. It can also include the physical processes of the entire universe, so, from one perspective physicists can be considered experts in computation as defined by the laws of physics.

Typically, however, when one thinks of computing one thinks not of physical law but of computer science or information technology. In practice, processes of computation are converted into abstract models that can be expressed as algorithms, or step-by-step procedures. It is in these abstract models that one encounters terms like one, zero, true, false, logical operations, programs and threads. In 1936, Alan Turing developed the abstract notation for what we would now call a programmable computer and soon after, John von Neumann developed a model for implementing such a computer.

These abstract models must, however, be implemented in real physical systems. The "real" ones and zeros inside your computer are not abstract ideas -- they actually correspond to potentials in conductors or the polarization of magnetic domains, for example. The "clocks" inside your computer are not abstract ideas – they actually correspond to physical oscillators and at today's frequencies, "clock ticks" are electromagnetic fields propagating over transmission lines. Any processor designer will tell you that engineers have already encountered physical limits regarding what can be done. As put by David Deutsch,

> *computers are physical objects, and computations are physical processes. What computers can or cannot compute is determined by the laws of physics alone, and not by pure mathematics.*

As the number of components in a typical central processor increase, the number of atoms required to represent a bit of information decrease. Eventually, it is clear that the size of these components will approach the size of single atoms, and the physics governing their behavior will be quantum mechanics.

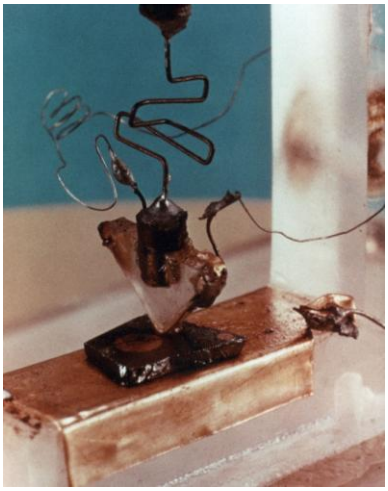
A natural consequence of the progression of Moore's Law[1] is that quantum effects must be considered in the design of computing systems. In 1980, Paul Benioff proposed that we try to work with this trend instead of fighting against it and implement computer logic operations according to the laws of quantum mechanics. This represents the beginning of the idea of quantum computation.

Computing as seen today grew out of a long sequence of disparate events. Nature gave us simple machines. It was eventually observed that combinations of simple machines could perform useful computations as analog computers. Turing and von Neumann eventually conceived the abstract models used to study and then implement programmable computers. With the introduction of the transistor and then the integrated circuit, these machines were made both smaller and powerful; and parallel developments in software for these machines brought us the general purpose programmable computers we have today.

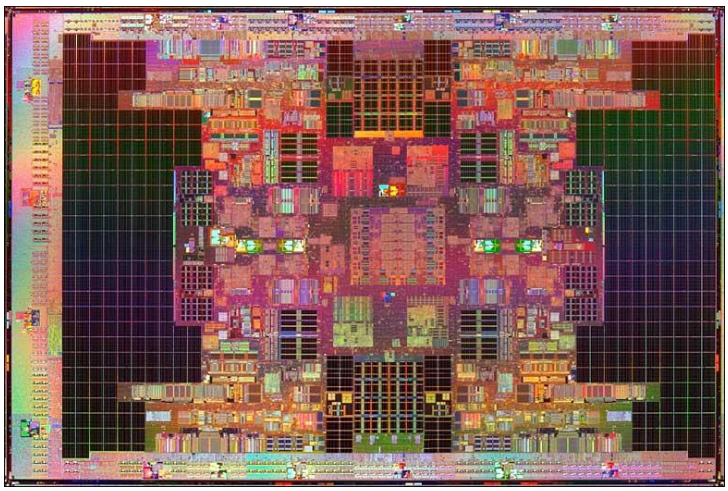
Quantum computing is in the very beginning stages of both research and development. Nature has given us quantum behavior. It has been observed that quantum mechanical systems can be used to perform useful computations faster than classical systems. Abstract models for quantum computation have been developed to allow study of problems independent of implementations and the realization of quantum computing devices is underway. Will there be an equivalent to Moore's law leading to immensely powerful quantum computers able to run circles around conventional machines? Nobody knows the answer to that question, but suspicions are strong based on the discovery of recently discovered fast versions of conventional algorithms.

There is currently no Femtosoft Corporation producing Heisenberg Picture Windows, nor is there a quantum-object-oriented language and compiler. Quantum computation is at the stage of putting together simple machines. For the time being, quantum computing and the quantum computers themselves will look and feel much more like the original point-contact transistor of Bardeen and Brattain shown in Figure 1 (a) instead of the modern microprocessor shown in Figure 1 (b)

[1] In 1965 Gordon Moore observed that the number of transistors in computer chips was doubling approximately every 18 months.



(a)



(b)

Figure1: The state of the art in transistor development over time. (a) The original point contact transistor developed by John Bardeen and Walter Brattain in 1947. (b) The six core Intel Xeon "Dunnington" processor announced in 2008.

The goal of this paper is modest – it attempts to explain how one can construct a kind of simple device on the order of that original point-contact transistor that can be made to perform simple computations in a way that hints at the kind of algorithmic improvement possible in quantum computing.

2. IMPLEMENTING COMPUTING DEVICES

There are four basic questions that need to be answered in order to implement a digital computing device:

- What is the representation of a logic level?
- How do you implement the logical operations of the device?
- How do you prepare the initial states?
- How do you measure the final states?

These questions will be answered in order for both the classical and quantum cases using brief examples of each.

2.1 LOGIC LEVEL REPRESENTATION

The most basic abstraction in a computation using binary logic is that of a binary digit or bit. A bit can take on the logical values "1" or "0." Because of the relation to Boolean logic, these logical values are sometimes called true and false, respectively.

2.1.1 CLASSICAL BITS

In electronic implementations of binary logic, a range of current or voltage levels represents a logic level and a bit of information. For example, a voltage in the range 0-1V may represent the logical value "0"while a voltage in the range 4-5V may represent a logical "1." These voltages and currents may change depending on the implementation of the logic and in fact zero volts could represent a logical "1" if so desired. The implementation of the logic level is quite independent of its abstract meaning.

Of course, the conventions for the symbols representing the logic levels are as completely arbitrary as the physical processes used to implement them. However common conventions have been developed and will be illustrated here.

2.1.2 QUANTUM BITS

Bits in quantum computing are called qubits (from a contraction of quantum bits). The first postulate of quantum mechanics is generally written as a variation of,

the state of a particle is represented by a vector $|\psi\ t\rangle$ *in a Hilbert space.*

One interprets "the state" as being loosely defined as, "everything known about the particle." If one replaces the term "particle" with "input" and "everything known" happens to be a logic level, one could infer that,

the logic level of an input is represented by a vector $|\psi\ t\rangle$ *in a Hilbert space.*

In fact, abstract quantum bits are logic levels in a binary logic system and are represented by the kets $|0\rangle$ indicating logic level zero and $|1\rangle$ indicating logic level one.

The third postulate of quantum mechanics is typically seen as a form of,

If a particle is in a state $|\psi\rangle$*, measurement of a variable corresponding to* Ω *will yield one of the eigenvalues of* ω *with probablility* $P(\omega) \propto |\langle\omega|\psi\rangle|^2$

From this one can infer that a qubit can be implemented by any quantum object with two well-defined and distinct eigenstates. Spin ½ particles come immediately to mind; and it turns out that systems using nuclear spins are very nice candidates for the implementation of quantum computing.

2.2 IMPLEMENTING LOGICAL OPERATIONS

There are two basic classical logic gates called NOT and AND. These basic gates may be composed into configurations implementing any other classical logical operation such as OR, NOR, XOR and XNOR as well as into registers and the many other building blocks of modern computers.

Because logic gates first appeared as modules in circuit schematics, they are often shown graphically as shown in Figure 2.

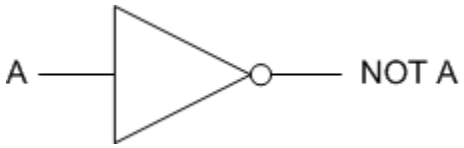


Figure 2: A typical graphical representation of a logical NOT operation used in a classical computer. The input is labeled A and the output is the logical inverse of the input.

The behavior of a logic gate is typically described by a truth table. This is literally a table that lists the output of a logic gate for every possible combination of inputs. The simplest non-trivial gate is a NOT gate that simply reverses the logic level of the input. The truth table for a NOT gate is shown in Table 1.

A	**NOT A**
0	1
1	0

Table 1: The truth table for the logical NOT operation. Inputs to the gate are shown in the column labeled **A** and outputs in the column **NOT A**.

This is called a unary operation since it takes a single input and provides a single output. Note that this is not the same thing as a *unitary* transform, which will later be used to implement quantum logical operators.

2.2.1 IMPLEMENTING LOGICAL NOT IN A CLASSICAL SYSTEM

The abstractions of modern computer science and engineering are sometimes so well ingrained that one often doesn't think of *executing* a logical operation. At the lowest levels of computer design, though, this is quite important. Logical operations do really take time and the details of how the operation is actually performed make up the execution of the operation. As an example of how one executes a logical NOT operation in Resistor-Transistor Logic (RTL) you might put together a circuit that looks like the one in Figure 3, below.

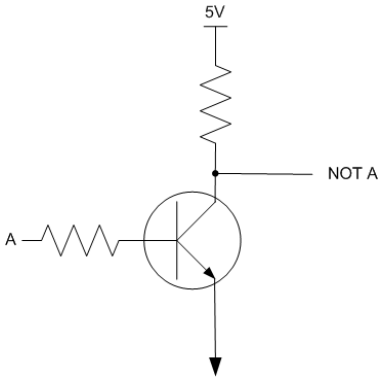


Figure 3: One possible implementation of a logical NOT gate using resistors and transistors. With appropriate definitions of logic levels, this circuit does implement the logical NOT operation.

A logical 0 must be a potential low enough to avoid forward-biasing the base-emitter junction of the transistor. This level must also be defined in order to be consistent with the collector-emitter voltage when the transistor is saturated. A logical one must be a potential high enough to turn on the transistor, but must also be consistent with the output voltage of the circuit when the transistor is off.

In order to execute this gate, one places a potential at input A and the transistor responds according to its transfer function, either turning on and saturating or turning off and allowing the collector resistor to pull the output up.

One rarely sees this level of detail when discussing logic gates in computation today; however, it is at just this level of detail that we are going to be discussing quantum computing here.

2.2.2 IMPLEMENTING LOGICAL NOT IN A QUANTUM SYSTEM

Consider the following unitary matrix operator.

$$U_{NOT} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

If the basis vectors of the quantum mechanical system are defined as,

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

it can be seen that the U_{NOT} operator does what a quantum NOT operator should do.

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}$$

or, alternatively, $U_{NOT}|0\rangle = |1\rangle$, $U_{NOT}|1\rangle = |0\rangle$ and $U_{NOT}\ \alpha|0\rangle + \beta|1\rangle\ = \alpha|1\rangle + \beta|0\rangle$.

Since the matrix of the U_{NOT} operator is the same as one of the famous Pauli spin matrices,

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

it is often called the PAULI X operator. This gate is graphically depicted as a rectangle as shown in Figure 4.

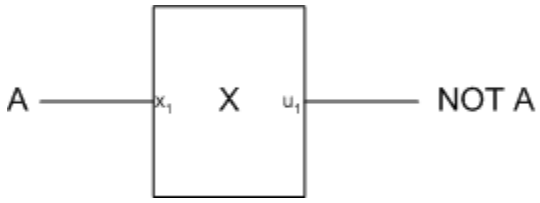


Figure 4: Graphical Depiction of a Quantum NOT Gate

As previously mentioned, spin ½ particles are very nice candidates for the implementation of quantum computing devices. It will be worthwhile to quickly review quantum mechanical spin before proceeding with a discussion of implementation details.

2.2.2.1 SPIN REVIEW

In classical mechanics, an object can undergo two types of angular momentum: The first type, orbital angular momentum ($L = r \times p$), is associated with motion of the center of mass of the object around some point – like the Earth orbiting around the sun. The second type, spin ($S = I\omega$), is associated with the motion of the object around its center of mass – like the Earth spinning on its axis.

In quantum mechanics, elementary particles carry an intrinsic angular momentum (S). This has nothing to do with motion in space since they are modeled as point-particles, but is analogous to classical spin and so the same term is used. Every kind of particle has a specific value of spin which never changes. For example, electrons, neutrons and protons have spin ½, while photons have spin 1.

The Schrödinger equation, models this behavior:

$$i\hbar\frac{\partial\Psi}{\partial t} = H\Psi \quad .$$

This equation can be solved by separation of variables. The solution constructs simultaneous eigenfunctions of three commuting operators, H, L^2 and L_z.

$$H\Psi = E\Psi, \quad L^2\Psi = \hbar^2 l(l+1)\Psi, \quad L_z\Psi = \hbar m\Psi \quad .$$

The algebraic theory of spin is exactly the same as that of orbital angular momentum, so one can write,

$$H\Psi = E\Psi, \quad S^2\Psi = \hbar^2 s(s+1)\Psi, \quad S_z\Psi = \hbar m\Psi \quad .$$

This is usually written in Dirac notation

$$H\left|s\ m\right\rangle = E\left|s\ m\right\rangle, \quad S^2\left|s\ m\right\rangle = \hbar^2 s(s+1)\left|s\ m\right\rangle, \quad S_z\left|s\ m\right\rangle = \hbar m\left|s\ m\right\rangle \quad ,$$

where s is the spin (½) and m is either (+½ or –½). Thus the orientation takes on two values, which we interpret as parallel to an arbitrary axis or antiparallel. For spin ½ particles, there are then two well-defined and distinct eigenstates,

$\left|\frac{1}{2}\ \left(+\frac{1}{2}\right)\right\rangle$, which we call spin up, and

$\left|\frac{1}{2}\ \left(-\frac{1}{2}\right)\right\rangle$, which we call spin down.

2.2.2.2 NMR REVIEW

Since all of the example implementations of quantum algorithms are illustrated here using Nuclear Magnetic Resonance (NMR) a quick review of the important concepts is in order.

A spinning charged particle acts like a magnetic dipole. The particle's magnetic dipole moment ($\vec{\mu}$) is proportional to its spin angular momentum,

$$\vec{\mu} = \gamma \vec{S} .$$

The proportionality constant (γ) is called the gyromagnetic ratio. If one considers a particle in a uniform magnetic field $\vec{B} = B_0 \hat{z}$ along the z-axis, the matrix representation of the Hamiltonian of this configuration would be

$$\mathcal{H} = -\gamma B_0 S_z = -\gamma B_0 \frac{\hbar}{2} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

The Hamiltonian is time-independent, so the general solution to the time-dependent Schrödinger equation is,

$$i\hbar \frac{d}{dt} \chi = \mathcal{H} \chi .$$

The solution can be expressed in terms of "stationary states,"

$$\chi\ t\ = a\chi_+ e^{-i\frac{E_+ t}{\hbar}} + b\chi_- e^{-i\frac{E_- t}{\hbar}} .$$

Where

$$\chi_+ = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

indicates the spin up state and

$$\chi_- = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

indicates the spin down state.

Since $\mathcal{H}\left|s\ m\right\rangle = E\left|s\ m\right\rangle$,

$-\gamma B_0 \frac{\hbar}{2} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = -\gamma B_0 \frac{\hbar}{2} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ implies that for χ_+ the energy $E_+ = -\gamma B_0 \frac{\hbar}{2}$, and

$-\gamma B_0 \frac{\hbar}{2} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = +\gamma B_0 \frac{\hbar}{2} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ implies that for χ_- the energy $E_- = +\gamma B_0 \frac{\hbar}{2}$.

Putting this together the general state is,

$$\chi\ t\ = a\chi_+ e^{-i\frac{-\gamma B_0 \frac{\hbar}{2} t}{\hbar}} + b\chi_+ e^{-i\frac{+\gamma B_0 \frac{\hbar}{2} t}{\hbar}}$$, which can be written as $$\chi\ t\ = \begin{pmatrix} ae^{\frac{i\gamma B_0 t}{2}} \\ be^{\frac{-i\gamma B_0 t}{2}} \end{pmatrix}.$$

With a little foreknowledge, one can write $a = \cos\left(\frac{\alpha}{2}\right), \quad b = \sin\left(\frac{\alpha}{2}\right),$ and calculate the expectation values of S_x, S_y, and S_z.

$$\langle S_z \rangle = \left[\cos\left(\frac{\alpha}{2}\right) e^{\frac{-i\gamma B_0 t}{2}} \quad \sin\left(\frac{\alpha}{2}\right) e^{\frac{-i\gamma B_0 t}{2}} \right] \frac{\hbar}{2} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \cos\left(\frac{\alpha}{2}\right) e^{\frac{i\gamma B_0 t}{2}} \\ \sin\left(\frac{\alpha}{2}\right) e^{\frac{-i\gamma B_0 t}{2}} \end{bmatrix} = \frac{\hbar}{2} \sin\ \alpha\ \cos\ \gamma B_0 t$$

$$\langle S_y \rangle = \left[\cos\left(\frac{\alpha}{2}\right) e^{\frac{-i\gamma B_0 t}{2}} \quad \sin\left(\frac{\alpha}{2}\right) e^{\frac{-i\gamma B_0 t}{2}} \right] \frac{\hbar}{2} \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \begin{bmatrix} \cos\left(\frac{\alpha}{2}\right) e^{\frac{i\gamma B_0 t}{2}} \\ \sin\left(\frac{\alpha}{2}\right) e^{\frac{-i\gamma B_0 t}{2}} \end{bmatrix} = -\frac{\hbar}{2} \sin\ \alpha\ \sin\ \gamma B_0 t$$

$$\langle S_z \rangle = \left[\cos\left(\frac{\alpha}{2}\right) e^{\frac{-i\gamma B_0 t}{2}} \quad \sin\left(\frac{\alpha}{2}\right) e^{\frac{-i\gamma B_0 t}{2}} \right] \frac{\hbar}{2} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \cos\left(\frac{\alpha}{2}\right) e^{\frac{i\gamma B_0 t}{2}} \\ \sin\left(\frac{\alpha}{2}\right) e^{\frac{-i\gamma B_0 t}{2}} \end{bmatrix} = \frac{\hbar}{2} \cos\ \alpha$$

From the expression $\langle S_z \rangle = \frac{\hbar}{2} \cos\ \alpha$ one can infer that $\langle S \rangle$ is tilted away from the z-axis at a constant angle α. From the expressions for $\langle S_z \rangle$ and $\langle S_z \rangle$ one can infer that the spin vector precesses around the z-axis at a frequency $\omega = \gamma B_0$. This frequency is called the Larmor frequency and it is dependent on the gyromagnetic ratio of the particle and on the magnetic field.

The energies of the quantum state with measured by S_z with the z-axis taken parallel to the magnetic field are

$$E_+ = -\gamma B_0 \frac{\hbar}{2} \quad \text{and} \quad E_- = +\gamma B_0 \frac{\hbar}{2}.$$

This is an example of the Zeeman Effect. In the same way that an atom can absorb or emit a photon and change energy state, a particular particle can be induced to change its magnetic field alignment if it receives an amount of energy equivalent to $\omega\hbar$ where ω is the Larmor frequency.

It will be seen that this phenomenon forms the basis of NMR and will be used extensively in the example implementations.

2.2.2.3 QUANTUM LOGIC LEVELS

Just as in the classical RTL case described above, the definitions for the implementations of a logical 0 state and a logical 1 state must be defined. The obvious definition is to associate a particular eigenstate of a spin ½ system with a logic level. Typically the following association is made.:

$$|0\rangle \equiv \left|\frac{1}{2}\ \left(+\frac{1}{2}\right)\right\rangle$$
$$|1\rangle \equiv \left|\frac{1}{2}\ \left(-\frac{1}{2}\right)\right\rangle .$$

In this case, the logical 0 is associated with spin up, and logical one is associated with spin down. Although this might seem intuitively backward, it will turn out to be the most sensible definition.

2.2.2.4 QUANTUM GATES USING NMR

Just as it was required to implement a logical NOT operation in a real transistor circuit, one must implement a spin-based quantum logical NOT operator in a real system. As has been repeatedly implied, a common way of doing this is by using Nuclear Magnetic Resonance (NMR) phenomena.

The basic "schematic" for a gate using NMR is shown in Figure 5.

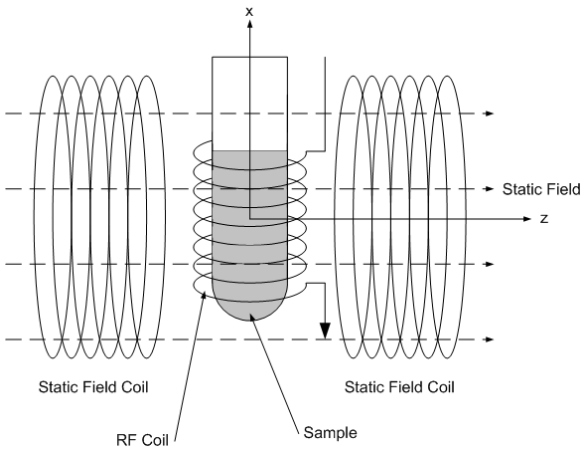


Figure 5: The basic setup for a implementing a quantum mechanical "logic gate" using NMR. The static field coils provide a strong static magnetic field and the RF coil is used to manipulate the spin state of the particles in the sample.

In an NMR apparatus, a sample is immersed in a strong static magnetic field. Recall that there are two states possible for a spin ½ particle: aligned such that the spin is parallel to the field, or aligned such that the spin is antiparallel to the field. There is a trade-off between the tendency of a spin system to remain in the low energy state – aligned with the magnetic field, and the ability of the system to gain energy. Some fraction of the particles in a sample will gain energy from thermal contact with their surroundings and change orientations to the higher energy state – that with the spin aligned antiparallel to the magnetic field. The total number of particles that do change orientation is related to the Boltzmann factor, which is the probability that a given system will be found at a particular energy. This factor is,

$$P(E) = e^{-\frac{E}{kT}}.$$

It can be seen that the ratio of probabilities of a sample of particles to be in a particular state is,

$$\frac{P(E_+)}{P(E_-)} = \frac{e^{\frac{\gamma \hbar B_0}{2kT}}}{e^{-\frac{\gamma \hbar B_0}{2kT}}}.$$

It turns out that there is an almost vanishingly small preference of particles to be in the aligned state. However there are Avogadro numbers of spins in a sample which are summed to make this preference observable. This is called the spin excess and provides a macroscopic magnetization that can be detected. This thermal equilibrium state is often simply called the thermal state of the sample.

If one takes a semiclassical view of a large number of particles immersed in the strong static magnetic field, usually called B_0, one sees a macroscopic magnetic dipole moment $\vec{\mu} = \gamma \vec{J}$ which is a result of the sum of all of the quantum mechanical magnetic dipole moments $\vec{\mu} = \gamma \vec{S}$ of the spin excess. This macroscopic magnetic dipole moment will precess around the field at a frequency $\omega = \gamma B_0$ as discussed above. A second magnetic field (usually called the RF field) designated $\vec{B}_1$ will be caused to rotate around the static field axis at the same frequency as the precession frequency.

This rotating field is actually created using the single RF coil shown in Figure 5. The simplest way to visualize how this happens is shown in Figure 6. Recall that a current passing through the RF coil will create a magnetic field parallel or antiparallel to the $\hat{x}$-

axis depending on the current direction. If this current is varied sinusoidally, the resulting magnetic field vector will oscillate from positive to negative and back again as shown in Figure 6 (top).

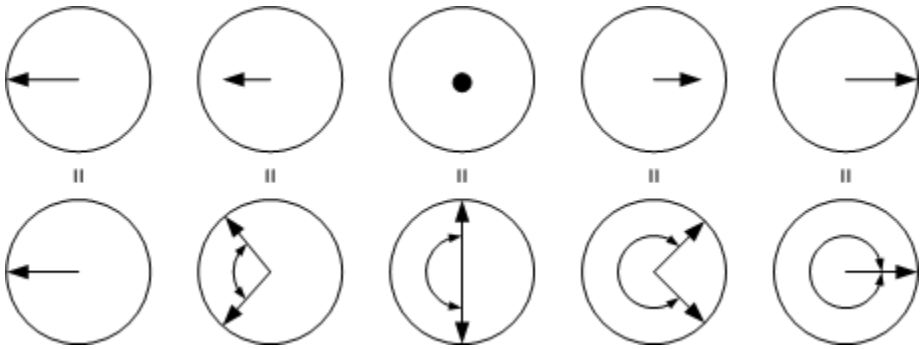


Figure 6: The RF field can be viewed as a sinusoidally varying signal in a single direction (top) as well as the sum of counter-rotating vectors (bottom).

Imagine that this magnetic field vector is actually the sum of two counter-rotating vectors as shown in Figure 6 (bottom). The vector direction of the sum of the two vectors in the bottom row will be the same as that of the vectors in the top row. Now if the rotational frequency of the vectors in the bottom row is chosen to rotate with a particle's Larmor frequency, and the direction of rotation is also chosen to match the direction of precession, the chosen rotating vector will have the characteristics required of the field due to the RF coil – $\vec{B}_1$.

Consider how this situation would appear in a frame of reference where the x-y plane is also rotating at $\omega = \gamma B_0$. Both the precessing spins and $\vec{B}_1$ will be rotating at the same rate and will appear *static* in this reference frame. The second counter-rotating magnetic field vector from Figure 6 will be rotating the wrong way at twice the required speed and can be disregarded in the analysis.

Figure 7 shows the end result. The ensemble dipole moment of the sample will be precessing around the static field at the Larmor frequency. The RF magnetic field will also be rotating around the static field at the Larmor frequency. If the x-y plane is also rotated at the Larmor frequency, the situation depicted in Figure 7 appears static in that rotating frame.

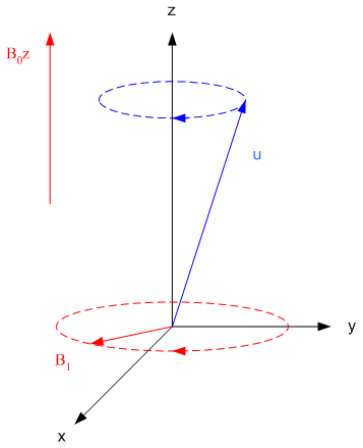


Figure 7: The Static and RF Fields and the Dipole Moment

This rotating reference frame turns out to be very useful for understanding what is happening in NMR systems. The equation of motion for the dipole moment in the laboratory frame is,

$$\frac{d\vec{\mu}}{dt} = \vec{\mu} \times \gamma \left[\vec{B}_0 + \vec{B}_1 \ t \ \right].$$

If $\vec{B}_1$ is taken along the x-axis, and a transform is made into the rotating frame, the equation of motion becomes,

$$\frac{\delta \vec{u}}{\delta t} = \vec{u} \times \gamma \left[\left(B_0 - \frac{\omega}{\gamma} \right) \hat{z} + B_1 \hat{x} \right].$$

An effective magnetic field can then be defined in this frame,

$$B_{eff} = \left(B_0 - \frac{\omega}{\gamma} \right) \hat{z} + B_1 \hat{x} \ ,$$

The cosine of the angle between the center of precession and the effective field is then

$$\cos \ \theta \ = \frac{B_0 - \dfrac{\omega}{\gamma}}{B_{eff}} \ .$$

The angle θ in the macroscopic rotating frame view corresponds to the angle α in the quantum mechanical treatment of spin developed previously. If $\omega = \gamma B_0$, the term in parentheses in B_{eff} above drops out and the magnetic moment sees a torque of,

$$\vec{\mu} \times \gamma \vec{B}_1.$$

Therefore, a magnetic moment that is initially parallel to the static field (in the z-axis) will be caused to rotate in the z-y plane at a frequency of $\omega = \gamma B_1$ as illustrated in Figure 8. (N.B. the orientation of the axes in the figure, with the z-axis vertical).

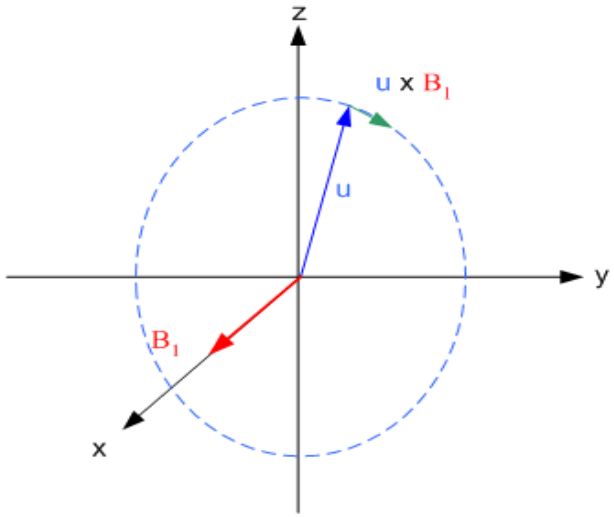


Figure 8: The Magnetic Dipole Moment Rotated by the Torque of the RF Field

In the jargon of NMR systems, if the rotating magnetic field is turned on for a time corresponding to a rotation of $\vec{\mu}$ around a quarter of the circle in Figure 8 and then turned off, it is called a $\pi/2$ pulse. This is because $\vec{\mu}$ will rotate $\pi/2$ radians in that time. Figure 9 shows the effect of a pulse of this type on the macroscopic magnetic dipole moment as viewed in the lab frame. Think of the tip of the vector as initially precessing tightly around the z-axis under the influence of the strong static magnetic field B_0.

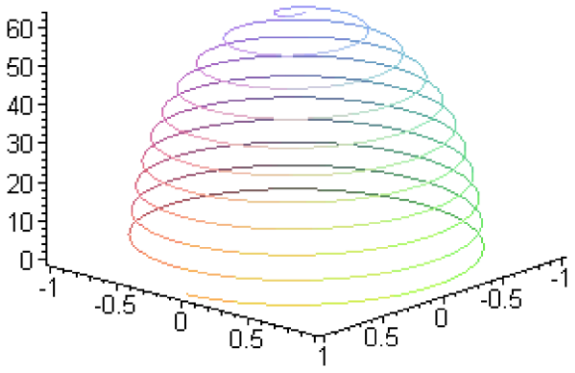


Figure 9: Effect of $\pi/2$ Pulse in Lab Frame

When the rotating magnetic field is turned on, a torque is applied to $\vec{\mu}$ which tends to rotate its precession into the x-z plane. Since $\vec{\mu}$ is precessing, it *spirals* down to the x-z plane.

If the rotating magnetic field is turned on for a time corresponding to a rotation of $\vec{\mu}$ around a half of the circle in Figure 8 and then turned off, it is called a π pulse. This is because $\vec{\mu}$ will rotate π radians in that time and the end result will be to rotate the spins of the spin excess from the parallel to the antiparallel alignment.

In other words, one way to look at the application of a π pulse to an NMR system is to change the state of the spin excess protons from $|0\rangle$ to $|1\rangle$ or vice versa – a NOT gate.

2.3 PREPARING THE INITIAL STATES

Preparation of initial states is, as you might expect a trivial operation in the classical case. As you might also expect, things get more involved in the quantum mechanical case.

2.3.1 PREPARING THE CLASSICAL INITIAL STATE

In electronic implementations of binary logic, a range of current or voltage levels represents a logic level and a bit of information. In Resistor-Transistor logic, a voltage in the range 0-0.6V may represent the logical value "0" while a voltage in the range 4-5V may represent a logical "1." Preparing a classical initial state simply means applying a particular voltage level.

2.3.2 PREPARING THE QUANTUM INITIAL STATE

When dealing with quantum mechanical implementations, there is no magic switch to flip to get the system into an initial state. Instead, the laws of nature are used to cause the system to settle into some form of equilibrium that is defined to be the initial state. For example, in NMR implementations, establishment of an initial state is usually done by waiting for a long enough period of time to allow the sample to return to thermal equilibrium with the spin excess defined as above. Since logical levels may be defined in any way, an initial state of $|0\rangle$ could be defined to be this equilibrium state. One could just as easily define $|1\rangle$ to be this state, however the convention is to use $|0\rangle$.

Preparation of the initial quantum state then simply means allowing enough time to pass to ensure that the sample has relaxed into the thermal equilibrium state as defined by the Boltzmann distribution.

2.4 MEASURING THE FINAL STATES

Measurement of final states is also, as you might expect, a trivial operation in the classical case. As you might also expect, things once again get more involved in the quantum mechanical case.

2.4.1 MEASURING THE CLASSICAL FINAL STATE

There are many ways to measure the output of the RTL NOT gate described above. You can walk down to your local automotive supply store and buy a voltmeter which you can use. You could add a Light Emitting Diode to the circuit and simply watch for light, as another example.

Although the actual operation of that voltmeter or LED is actually quite complex at a low level, most people probably do not appreciate this fact since the devices are so ubiquitous.

2.4.2 MEASURING THE QUANTUM FINAL STATE

When dealing with quantum mechanical implementations, there is no test point to look at in a relatively direct way for a logic level. In a purely quantum mechanical system, you would measure an observable. In a Stern-Gerlach experiment, you might look for the presence or absence of a spot. In the case of an NMR-based experiment you look for what is called the Free Induction Decay (FID).

As shown above, the act of executing a gate in an NMR system consists of applying an external RF signal that causes the macroscopic magnetization in the sample to be rotated. Once the driving force is removed, the magnetization continues to precess and produce its own changing magnetic field. It is the change in this macroscopic field over time which allows us to observe the results of a gate execution.[2]

Imagine a hypothetical bar magnet spinning in a coil. This would just be a simple electrical generator. Now imagine replacing the magnet with the precessing magnetization of an NMR sample. This situation is illustrated in Figure 10. Thie illustration shows a red magnetic field vector, B(t), rotating in the x-y plane representing the precessing magnetization. A (thick black) coil is shown placed in the y-z plane with its center at the origin to detect the changing field. The coil in this case essentially detects the changing magnetic moment in the $\hat{x}$ direction.

[2] The question of whether or not an NMR system can then be classified as a true quantum computation implementation is left for another time.

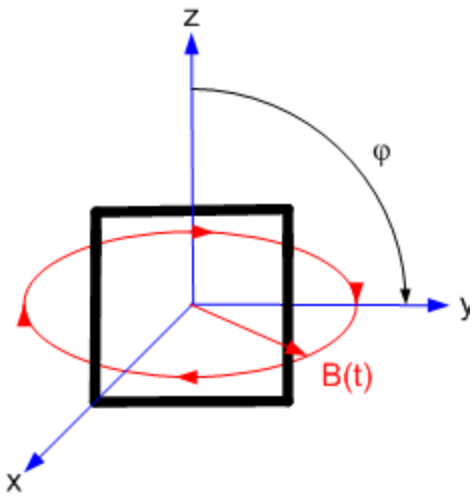


Figure 10: A Stylized Detector

When the driving rotating magnetic field is removed, as mentioned above, the magnetization continues to precess, but the relaxation process also begins and the spins begin to return to their macroscopic equilibrium value. If the same coil used to generate the RF field is used to detect the precession of the spins as they relax, an exponential decay of the signal will be seen with the time constant determined by characteristics of the sample itself. This is called the Free Induction Decay (FID). Figure 11 shows what a response might look like.

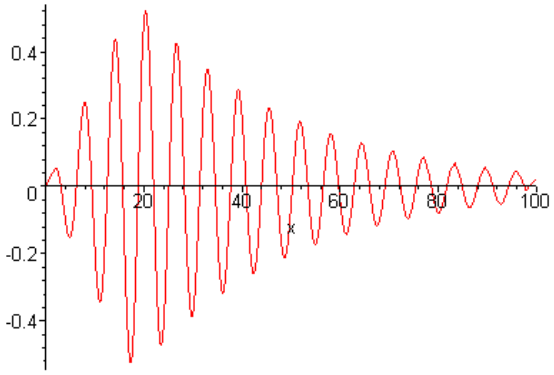


Figure 11: Example of a Free Induction Decay. The increasing part of the signal shown happens during stimulation and the decreasing part during relaxation. The decay time constant is typically on the order of milliseconds.

Detecting the presence or absence of a FID could be as simple as observing a terminal of the RF coil with an oscilloscope. Modern NMR devices, however, will actually sample and process the resulting signal digitally. Additionally a Fourier transform is usually performed and the output of this process will look something like Figure 12.

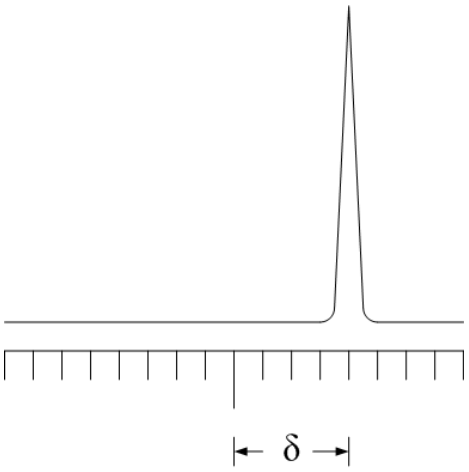


Figure 12: Example of the output of an NMR machine. The single spike in the Fourier transform of the sampled data represents the frequency at which the spin excess of the sample precessed after an RF pulse. Rather than display a huge range of possible frequencies, a delta from a reference resonance frequency is reported in various ways depending on the machine.

Additionally, the typical modern NMR device has a phase-sensitive detector. Conceptually this just adds another orthogonal measurement coil. Just as a coil in the y-z plane effectively measures the component of the magnetization in the $\hat{x}$ direction, a coil in the x-z plane measures the component of the magnetization in the $\hat{y}$ direction. It's not really done that way in a real NMR system (which would use mixers and phase shifters), but this model, shown in Figure 13, serves to illustrate the result conceptually.

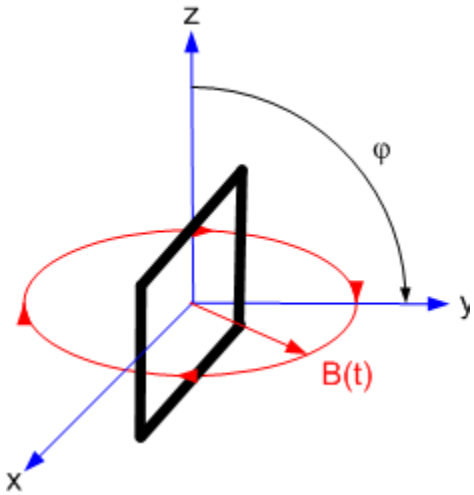


Figure 13: A Phase Sensitive Detector adds a second orthogonal coil in order to determine the phase of the precessing magnetization.

With a phase sensitive detector it is possible to determine the axis on which free precession and relaxation begins. Figure 14 shows a representation of how the amplitude

of the detected FID grows as the magnetization is rotated from the z-axis toward the x-y plane, peaking at $90°$ as the result of a $\pi/2$ pulse and returning back to zero as the magnetization is flipped into the –z axis as the result of a π pulse at $180°$.

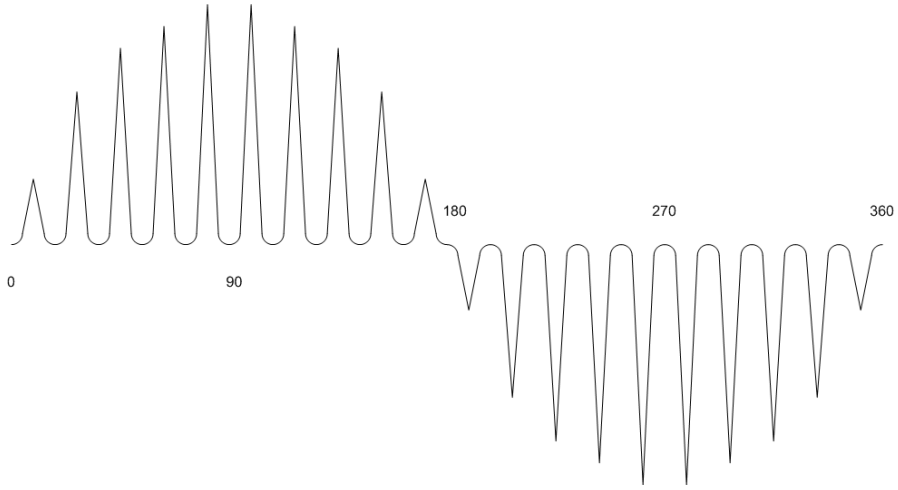


Figure 14: A plot of the angle φ from the z-axis at which a measurement is performed versus the displayed Fourier transformed FID. This illustrates that a Phase Sensitive Detector allows an NMR machine to differentiate between a magnetization precessing initially from the x- or y-axis versus one starting on the –x- or –y-axis.

As the length of the RF pulses is lengthened, the magnetization is rotated on around into the –x plane and a negative FID spike is shown at $270°$ as the result of a $3\pi/2$ pulse. As the length of the RF pulse approaches a 2π pulse the resulting spike is reduced in amplitude back toward zero.

Thus a final state that begins relaxing from along the y-axis can be differentiated from a final state that begins relaxing from along the –y-axis.

3. MORE THAN A CLUMSY, EXPENSIVE TRANSISTOR

In the previous section, it was shown how the equivalent of a classical NOT gate could be constructed using physical manipulation of underlying quantum processes (albeit an ensemble of processes averaged into a macroscopic quantity). If that were all there were to the story, a very expensive and clumsy way of doing what can be done by a microscopic transistor today would have been illustrated. The important quality that quantum mechanical computation brings to the table is superposition of states and this makes all the difference.

Recall the principle of superposition from your elementary quantum mechanics class. If $|\xi\rangle$ and $|\zeta\rangle$ represent possible states of a particle, so does $\alpha|\xi\rangle + \beta|\zeta\rangle$. Consider what this means when speaking about the logic levels $|0\rangle$ and $|1\rangle$.

In a classical system, a gate input can take on the values logic level 0 or logic level 1 – that's it. A logic level somewhere between 0 and 1 indicates a broken machine. In a quantum system, a gate input can also take on the values $|0\rangle$ and $|1\rangle$; but an input can also take on a logical value that is the superposition of states. It turns out that the most general representation of a quantum bit state is $\alpha|0\rangle + \beta|1\rangle$.

Recall that a unitary matrix operator and a set of basis vectors were previously defined as,

$$U_{NOT} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Applying this operator to a general quantum bit state results in,

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}$$

or $U_{NOT} \quad \alpha|0\rangle + \beta|1\rangle \quad = \alpha|1\rangle + \beta|0\rangle$. Notice that the NOT operation has been performed on both terms of the single input state *simultaneously*. This is not possible with a classical system and is the root mechanism of the speedup available in quantum systems.

3.1 MULTIPLE UNARY GATES

The only non-trivial unary (single bit) logic gate for classical computation is the NOT gate. It turns out that there are a number of additional interesting single-bit quantum gates available to designers of quantum computers other than the NOT gate. In order to understand them more fully, it will be worthwhile to review the Bloch Sphere and how it relates to the operators that implement the gates, especially in NMR.

Figure 15 shows a Bloch Sphere. In the examples of NMR-based quantum computing shown so far the basis vectors are $|0\rangle$ which represents spin up in the z-direction; and $|1\rangle$ which represents spin down in the z-direction.

$$|0\rangle$$

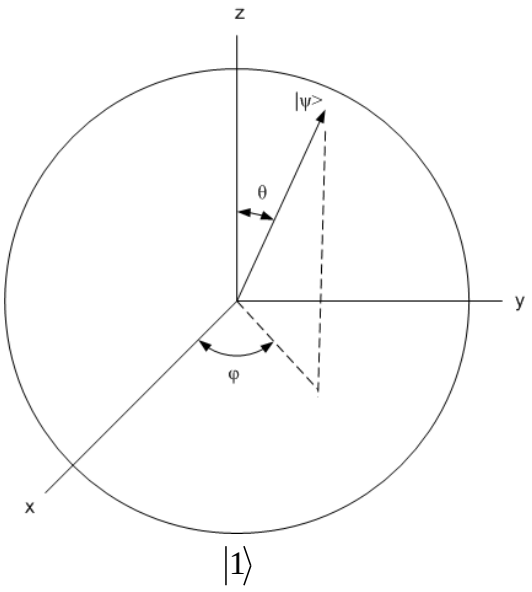


Figure 15: The Bloch Sphere. A Bloch Sphere is a unit sphere on which surface points represent general quantum mechanical states. A Bloch Vector is a vector represented by a point on the sphere.

The NOT operator takes a state and rotates it about the x-axis by π radians. The Pauli X matrix is called that because when it is exponentiated it gives rise to a rotation operator about the x-axis. Recall that the quantum NOT gate is identified with the Pauli X matrix. It can be seen from Figure 15 that a rotation of π radians about the x-axis is indeed a logical NOT operation. Also note that this is an abstraction of the process shown in Figures 8 and 9. In fact the Bloch Sphere was invented in the context of NMR but is used generically when discussing quantum states.

Similarly the Pauli Y matrix represents a rotation about the y-axis and the Pauli Z matrix represents a rotation about the z-axis. These rotation operators are defined as,

$$R_x(\theta) \equiv e^{-i\frac{\theta}{2}X} \quad R_y(\theta) \equiv e^{-i\frac{\theta}{2}Y} \quad R_Z(\theta) \equiv e^{-i\frac{\theta}{2}Z},$$

and correspond to three quantum gates known as X, Y and Z. In another common and handy notation, the action of an on-resonance RF pulse is often described by a pulse propagator,

$$\theta_{\phi}$$

where θ is the rotation angle and ϕ is the axis about which the rotation is performed.

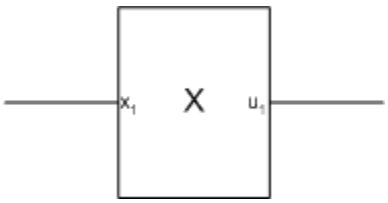
3.1.1 PAULI X GATE

The Pauli X gate is, as previously mentioned, equivalent to a logical NOT operation. In NMR it is implemented as a rotation of π radians about the x-axis. This gate can be viewed as a 2×2 matrix, a rotation operator, an operator on a general state, a graphical depiction or a pulse propagator, depending on the context and its conventions:

$$X \equiv \sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$R_x\ \theta\ \equiv e^{-i\frac{\theta}{2}X}$$

$$X\ \ \alpha|0\rangle + \beta|1\rangle\ = \alpha|1\rangle + \beta|0\rangle .$$



$$\pi_X$$

It can be most readily seen from the pulse propagator notation, π_X, that the Pauli X gate is implemented in NMR using a π pulse about the x-axis. Looking back at Figure 15, it is readily verified that this results in a change from state $|0\rangle$ to state $|1\rangle$ or vice versa.

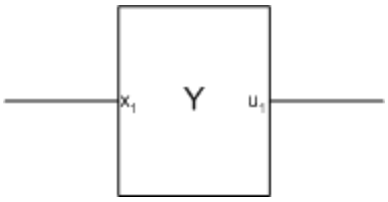
3.1.2 PAULI Y GATE

The Pauli Y gate is a completely new logic gate that doesn't exist classically. This gate can also be viewed as a 2×2 matrix, a rotation operator, an operator on a general state, a graphical depiction or a pulse propagator:

$$Y \equiv \sigma_Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

$$R_y\ \theta\ \equiv e^{-i\frac{\theta}{2}Y}$$

$$Y\ \ \alpha|0\rangle + \beta|1\rangle\ = i\alpha|1\rangle - i\beta|0\rangle .$$



$$\pi_{Y}$$

Again, it can be readily seen from the pulse propagator notation that the Pauli Y gate is implemented in NMR using a π pulse about the y-axis. In real machines this is accomplished by careful manipulation of the phase of the stimulus, but without loss of generality it can be imagined as being done as described above by driving an orthogonal RF coil oriented along the y-axis.

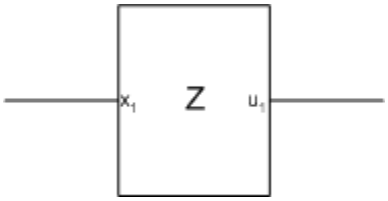
2.1.3 PAULI Z GATE

The Pauli Z gate is also a completely new logic gate that doesn't exist classically. This gate can be viewed as a 2×2 matrix, a rotation operator, an operator on a general state, a graphical depiction or a pulse propagator:

$$Z \equiv \sigma_Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$R_Z(\theta) \equiv e^{-i\frac{\theta}{2}Z}$$

$$Z(\alpha|0\rangle + \beta|1\rangle) = \alpha|0\rangle - \beta|1\rangle$$



$$\pi_{Z}$$

The Pauli Z gate cannot be directly implemented in NMR but a rotation of an angle θ can be implemented as a composition of X and Y gates.[3] In the pulse propagator notation this would be (read right to left),

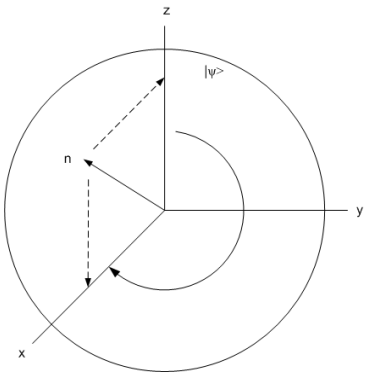
[3] It is worthwhile at this point to note that rotations in NMR have been described in many ways over the years. Different research communities have adopted different notations. Rotation operators and pulse propagators have been shown above. There are several more common notations used. In some of these notations, operators will bind from left to

$$\theta_Z = \left(\frac{\pi}{2}\right)_X \quad \theta_Y \left(\frac{\pi}{2}\right)_{-X}$$

This is fairly difficult to visualize, but can be understood by tracing the trajectory of the vector in question on the surface of a Bloch Sphere. The reader is encouraged to find a handy Bloch Sphere (perhaps a tennis ball), apply a coordinate system and execute a Pauli Z Gate to see how this works.

3.1.4 HADAMARD GATE

Other kinds of unary gates are possible as well. Rotation operators can be defined about an arbitrary axis. If this arbitrary axis is defined as $\left(\frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}}\right)$ a rotation of π radians about this axis is called a Hadamard gate.



This is sometimes called a "square root of NOT" gate, since it turns a $|0\rangle$ into a $\frac{1}{\sqrt{2}}\ |0\rangle + |1\rangle$ which is "halfway" between $|0\rangle$ and $|1\rangle$; and it turns a $|1\rangle$ into $\frac{1}{\sqrt{2}}\ |0\rangle - |1\rangle$ which is also "halfway" between $|0\rangle$ and $|1\rangle$. Another way to look at the Hadamard gate is that it creates a superposition of states, since it takes,

$$|0\rangle \rightarrow \frac{1}{\sqrt{2}}\ |0\rangle + |1\rangle \quad \text{and} \quad |1\rangle \rightarrow \frac{1}{\sqrt{2}}\ |0\rangle - |1\rangle \ .$$

---

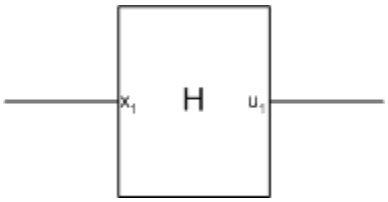
right and in others from right to left. This can be the source of much confusion. In the remainder of this document, notation will be restricted to the pulse propagator notation which is most common in the quantum computing literature. In pulse propagator notation, operators are treated like quantum mechanical operators and are applied from the right to the left.

The Hadamard gate is yet another completely new operator that doesn't exist classically. It can also be viewed as a 2×2 matrix, a rotation operator, an operator on a general state or depicted graphically:

$$H \equiv \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$R_n\ \theta\ \equiv e^{-i\frac{\theta}{2}\hat{n}\bullet\sigma}$$

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



$$\left(\frac{\pi}{2}\right)_Y$$

It can be seen from the pulse propagator notation that the Hadamard gate is implemented in NMR using a $\pi/2$ pulse about the y-axis. This is technically called a Pseudo-Hadamard gate since the single $\left(\frac{\pi}{2}\right)_Y$ pulse is not self-reversible which is required of a true Hadamard gate, however the Pseudo-Hadamard gate is still quite useful and often used.

3.2 MULTIPLE BIT GATES

There are many unary gates available to quantum algorithm "circuit designers." There are also many possible binary gates, or gates with two inputs.

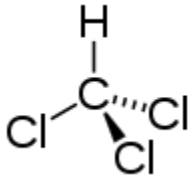
3.2.1 ADDRESSABLE BITS

The first issue that must be resolved when implementing gates with multiple inputs is that there is a need to differentiate between the inputs. So far abstract particles of spin ½ have been discussed, and a large collection of these particles has been imagined to implement a single qubit in an NMR implementation. When considering multiple input gates, individual bits must be "addressable." For example, input A is represented by a collection of spins and input B is represented by some other collection of spins.

In NMR implementations, individual spin ½ entities can be addressable if they belong to a different nuclei. Recall that the Larmor Frequency of a particle is dependent on the

gyromagnetic ratio γ. This is often tabulated in terms of $\gamma/2\pi$ with dimension MHz/T. For example, ^{1}H has a $\gamma/2\pi$ of $42.756\,MHz/T$, while ^{13}C has a $\gamma/2\pi$ of $10.705\,MHz/T$.

In NMR in order to implement a quantum gate with two individually controllable inputs, a sample could be used that contains multiple species of nuclei. For example, chloroform, $CHCl_3$, contains both carbon and hydrogen atoms.



Applying an RF field appropriate to$10.705\ MHz/T$ would manipulate the spins of the carbon atoms, thereby individually addressing the state represented by those atoms' spins. Applying an RF field appropriate to $42.756\,MHz/T$ would individually address the state represented by the hydrogen atom spins. The set of carbon atom spins could then represent "input A" and the set of hydrogen atom spins could represent "input B."

During the detection process, free induction decays would occur at both of these resonant frequencies and a Fourier transform of sample data would yield two peaks, one corresponding to the spins in the carbon atoms and one peak corresponding to the spins in the hydrogen atoms. This is illustrated in Figure 16.

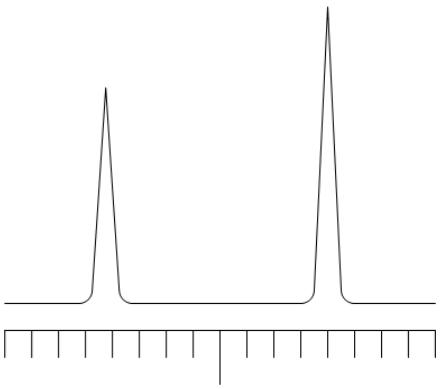


Figure 16: Example of the output of an NMR machine reading two quantum bits. The "spike" to the left of the reference frequency signifies detection of the spins representing one of the bits and the spike to the right represents detection of the other bit.

3.2.1 CONTROLLED NOT GATE

On of the most useful gates used in quantum computing is the controlled-NOT or CNOT gate. This is an extension of the classical exclusive-or (XOR) gate.

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Table 2: Truth table for the logical XOR operation. Inputs to the gate are shown in the column labeled **A** and **B** with outputs in the column **A XOR B**.

There are two important ways to look at the behavior of an XOR gate. First, it behaves as an adder without carry. This binary addition operator is often depicted using the symbol $\oplus$ instead of XOR. The truth table for an XOR gate could be written as,

$$0 \oplus 0 = 0,\ 0 \oplus 1 = 1, 1 \oplus 0 = 1,\ 1 \oplus 1 = 0.$$

Looking at the XOR gate differently, it can be seen as a controllable NOT gate. If one looks at input **A** as a control input, input **B** is passed unchanged to the output if that control input is at logic level 0; and input **B** is passed to the output inverted if the control input is at logic level 1. Another name for logical inversion is the NOT operation.

The graphical depiction of the quantum controlled-NOT gate actually includes references to both interpretations as shown in Figure 17.

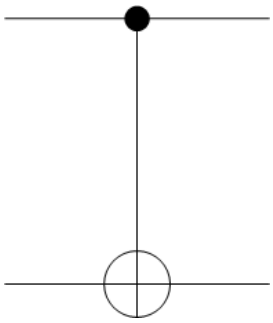


Figure 17: Graphical depiction of the quantum controlled-NOT gate. The upper "wire" represents the control qubit and the bottom "wire" represents the input. The output bit at the lower right represents the binary sum of the control and input bits as shown above and uses the same binary addition symbol.

Notice that the classical XOR gate has two inputs and one output. The quantum gate must have two inputs and two outputs since it is implemented as a unitary transformation – by definition reversible. The classical XOR loses information in its transfer function since the inputs cannot be reconstructed from the outputs and is therefore not reversible.

It turns out that the controlled-NOT gate is essential in quantum computing since all operations necessary for quantum computing can be shown to be achievable using a

controlled-NOT gate and a small set of one-qubit gates, the Hadamard gate, the phase (S) gate and the $\pi/8$ (T) gate. This forms a universal set of gates just as the NOT and AND gates form a universal set of classical gates.

3.2.2 HILBERT SPACE OF A TWO QUBIT SYSTEM

The Hilbert space of a two qubit system needs to be extended since there are more than just the two orthogonal states $|0\rangle$ and $|1\rangle$. There are four product states with associated basis vectors that must be considered,

$$|00\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad |01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \qquad |10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \qquad |11\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

Operators will need to be 4×4 matrices. It turns out that the matrix operator that acts on one qubit of a two qubit system may be determined by calculating the tensor product between the one qubit operator and the 2×2 identity matrix.

$$\hat{O}_a = \hat{O} \otimes I \qquad \hat{O}_b = I \otimes \hat{O}$$

A convention for writing the state as $|ab\rangle$ is used, along with the fact that operator $\hat{O}_a$ acts on the first qubit and operator $\hat{O}_b$ acts on the second. As an example, taking the NOT gate, which is a single qubit operator defined by the matrix,

$$X \equiv \sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

and expanding it into a two qubit system acting on the first qubit, one would make the following calculation,

$$\hat{O}_a = \hat{O} \otimes I = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

The tensor product $\otimes$ is done by replacing the 0 in row 1, column 1 of the operator $\hat{O}$ with 0 times the identity matrix,

$$\begin{bmatrix} 0 & \\ & \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & & \\ 0 & 0 & & \\ & & & \\ & & & \end{bmatrix}.$$

The 1 in row 1, column 2 of the operator is replaced by 1 times the identity matrix, etc.

$$\begin{bmatrix} & 1 \\ & \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} & & 1 & 0 \\ & & 0 & 1 \\ & & & \\ & & & \end{bmatrix}$$

A quick check of this new operator shows the state $|00\rangle$ being changed to $|10\rangle$,

$$\hat{X}|00\rangle = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = |10\rangle$$

and the state $|10\rangle$ being changed to $|00\rangle$,

$$\hat{X}|10\rangle = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = |00\rangle$$

Another interesting way of looking at the matrix operator is as a kind of truth-lookup-table. The possible input states are shown above the first row of the matrix and the possible output states are shown to the right of the last column. A connection is made between input state and output state by placing a single 1 in the column which joins one input to one output.

$\|00\rangle$	$\|01\rangle$	$\|10\rangle$	$\|11\rangle$	
0	0	0	1	$\|00\rangle$
0	0	1	0	$\|01\rangle$
0	1	0	0	$\|10\rangle$
1	0	0	0	$\|11\rangle$

For example, if it is desired that the output state be set to $|11\rangle$ in the case where the input state is $|00\rangle$, a single 1 is placed in column 1, row 4. The same procedure is performed to map all inputs to all outputs exactly once. The matrix above acts as a NOT gate that inverts both inputs simultaneously.

Commonly, the row across the top is elided as redundant; and in this picture the matrix representation of a two input NOT gate would be,

$$NOT_{ab} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} |00\rangle \\ |01\rangle \\ |10\rangle \\ |11\rangle \end{matrix}$$

Using this approach, one can straightforwardly construct the matrix representations of the controlled-NOT gate. There are two representations since the implementation will depend on which qubit is viewed as the control bit.

$$CNOT_a = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{matrix} |00\rangle \\ |01\rangle \\ |10\rangle \\ |11\rangle \end{matrix} \quad CNOT_b = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{matrix} |00\rangle \\ |01\rangle \\ |10\rangle \\ |11\rangle \end{matrix}$$

or, more traditionally,

$$CNOT_a = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad CNOT_b = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

Recall that the controlled-NOT gate is a form of exclusive-OR gate, and so its operation can also be viewed as the binary addition of two qubits,

$$CNOT_a |a,b\rangle = |a, a \oplus b\rangle \qquad CNOT_b |a,b\rangle = |a \oplus b, b\rangle$$

This view will become very important when developing quantum algorithms using superpositions of states.

3.2.3 CONDITIONAL INVERSION OF SPIN IN NMR

It is important to realize that the execution of a quantum gate may involve the normal time evolution of a state. That is, it is not required that a gate be associated with an

active RF pulse. The precession of the spins in the natural course of their time evolution may also be treated as a gate. In fact the most useful operation may be one in which there is no RF pulse.

Of particular interest in the case of controlled-NOT gates is the so-called J-coupling. J-coupling is an interaction between spins which are mediated by electrons shared through a chemical bond. As two spins interact, the local magnetic field of each of the spins is affected by the other. In this case, the magnetic field "felt" by the spins will be different than the applied field by some constant amount. In the case of liquid samples, this is a scalar value called J.

The key observation to make here is that the apparent magnetic field of one spin is affected by the spin state of the other spin. It turns out that a parallel or antiparallel coupling between two spins will increase or decreased the precession frequency of both spins by an amount $\pm\pi J$ depending on the relative orientations of the spins. The difference in precession rate due to the J-coupling is small compared to the Larmor precession rate. Recall that if a spin is precessing at the Larmor rate, and the reference frame is rotating at the same rate, the magnetic moment appears static in that frame. If an additional J-coupling term is present, it will cause the magnetism in the rotating frame to precess in a positive or negative sense in the rotating frame. Typical precession rates due to J-coupling are on the order of a few hundred Hertz.

A pulse propagator sequence used to observe this phenomenon is,

$$\left(\frac{\pi}{2}\right)_Y^1 U\left(\frac{1}{2J}\right)\left(\frac{\pi}{2}\right)_X^1$$

This sequence bgeins with a $\pi/2$ pulse about the x axis tuned to the first spin's Larmor frequency. The second $U\ \tau$ part of the sequence corresponds to a unitary evolution of the system for a time $\tau = 1/2J$ which allows the rotation due to J-coupling to evolve for the given time. The final part is a $\pi/2$ pulse about the y axis, again tuned to the first spin's Larmor frequency.

If the rotating frame is set to the precession frequency a spin unaffected by J-coupling, the result of a J-coupling evolution in that frame will be to see the magnetization angle change with an angular frequency $\pm\pi J$ over the time $1/2J$. The ending angle is determined by the expression,

$$\theta = \omega t = \pm\pi J \times \frac{1}{2J} = \pm\frac{\pi}{2}$$

Figure 18 shows the results of the three operations on a state $|\uparrow\uparrow\rangle$ with only the first vector shown moving on the surface of a Bloch sphere.

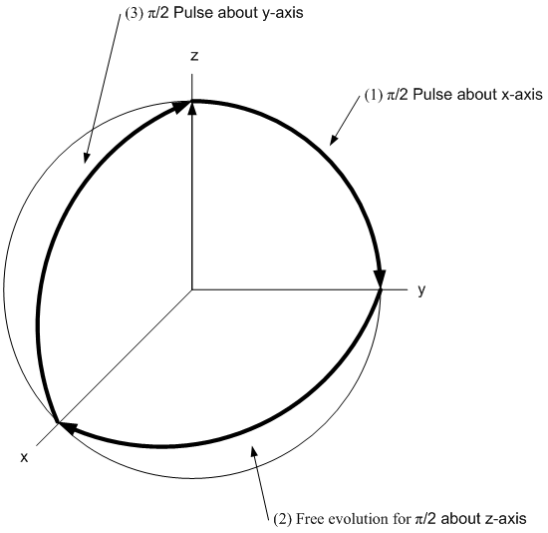


Figure 18: The action of a conditional inversion pulse sequence on a vector representing a $|\uparrow\rangle$ state. A second spin assumed also to be in a $|\uparrow\rangle$ state is not shown.

The other case is shown in Figure 19. In this case the starting state is $|\uparrow\downarrow\rangle$ with the second vector not shown. At the end of the sequence, the first vector has been flipped and the ending state is $|\downarrow\downarrow\rangle$

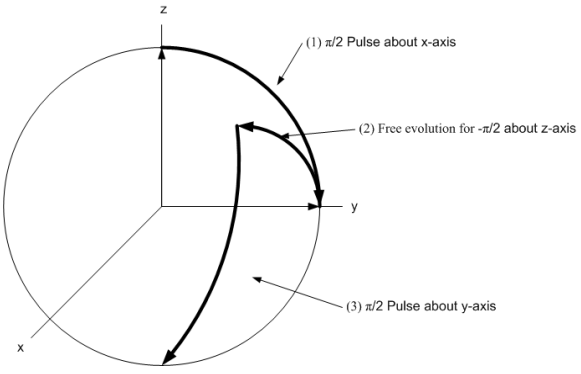


Figure 19: The action of a conditional inversion pulse sequence on a vector representing a $|\uparrow\rangle$ state. A second spin assumed also to be in a $|\downarrow\rangle$ state is not shown.

If one interprets the second spin (not shown above) as a control bit, then the basic operation of a controlled-inversion gate has been demonstrated. Recall that in NMR

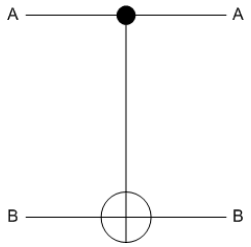
quantum computing implementations, spin up is defined as $|0\rangle$ so the "logic" illustrated above is

$$|00\rangle \rightarrow |00\rangle \quad |01\rangle \rightarrow |11\rangle .$$

The second qubit is interpreted as a control bit causing the first bit to be inverted when true.

3.2.3 IMPLEMENTATION OF CNOT IN NMR

Given this background, the implementation of both species of controlled-NOT gates in NMR is understandable. The first species called $CNOT_A$ uses bit A as the control bit and bit B as the target bit,

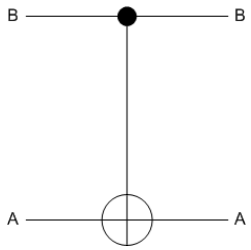


A pulse sequence in pulse propagator notation which will implement $CNOT_A$ could be,

$$CNOT_A = \left(\frac{\pi}{2}\right)_X^B U_J \left(\frac{1}{2J}\right) \left(\frac{\pi}{2}\right)_Y^B$$

First a $\pi/2$ pulse is applied to the target bit about the y-axis (recall the operators are read right to left). Then an evolution period of time $\tau = 1/2J$ under the J-coupling is allowed. Finally, second $\pi/2$ pulse is applied to the target bit about the x-axis.[4]

The second species called $CNOT_B$ uses bit B as the control bit and bit A as the target bit,



[4] In real NMR systems, a composite-z pulse would be applied to both bits at the end of the sequence, but this is not required to understand the operation and so is deleted from this example.

A pulse sequence in pulse propagator notation which will implement $CNOT_B$ could be,

$$CNOT_B = \left(\frac{\pi}{2}\right)_X^A U_J \left(\frac{1}{2J}\right) \left(\frac{\pi}{2}\right)_Y^A$$

The $CNOT_B$ implementation simply reverses the sense of the control and target bits in the graphical depiction and the pulse sequence implementation.

4. QUANTUM CIRCUITS

The implementations of several basic quantum gates have been discussed. Just as classical computer logic gates are composed into circuits, quantum computer gates are composed into quantum circuits. A few symbols need to be introduced before proceeding.

4.1 QUANTUM CIRCUIT NOTATION

The symbol for a made measurement is a graphical depiction of an analog meter as shown in Figure 20. Generally, this measurement symbol implies a quantum mechanical measurement in the sense of an operator and a projection onto a state, just as in any quantum mechanics problem. In an NMR implementation a measurement implies the detection of the precessing magnetization of the sample via the FID as described above.

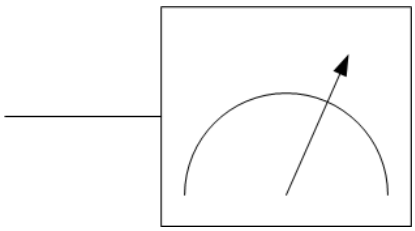


Figure 20: Graphical depiction of a measurement process in a quantum circuit.

The single line coming in to the left side of the meter of Figure 20 signifies a "wire" carrying a single qubit. A double line corresponds to a wire carrying a single classical bit. A circuit "bus" is indicated by a slash through the wire with an annotation indicating the number of wires in the bus. There may be classical and quantum bus depictions. Time is assumed to progress from left to right across these wires. Samples of these depictions are shown in Figure 21.

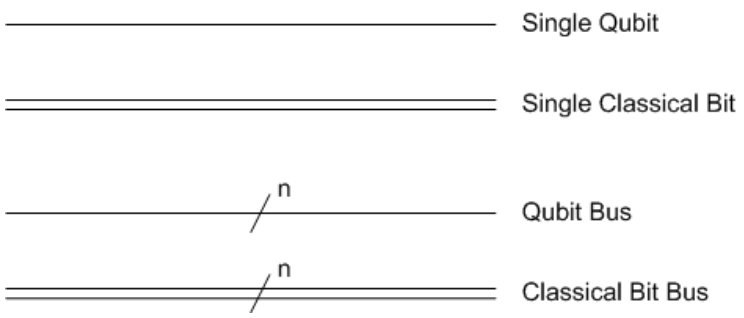


Figure 21: Graphical depiction of signals in a quantum computation.

With this information it is now possible to design and depict a complete quantum circuit

4.1 A FIRST QUANTUM CIRCUIT

Figure 22 shows a simple quantum circuit. One can also look at it as an algorithm. The circuit is read from left to right. At the left side, notice that a control qubit and a data qubit are inputs. This is therefore a two qubit system and the underlying NMR implementation needs to have a sample in which there are two spins that can be individually manipulated. As described above, chloroform, $CHCl_3$, contains both carbon and hydrogen atoms and could provide the necessary addressable bits.

Applying an RF field appropriate to10.705 MHz/T would manipulate the spins of the carbon atoms, thereby individually addressing the state represented by those atoms' spins. Applying an RF field appropriate to 42.756 MHz/T would individually address the state represented by the hydrogen atom spins. The set of carbon atom spins could then represent "Control" and the set of hydrogen atom spins could represent "Data."

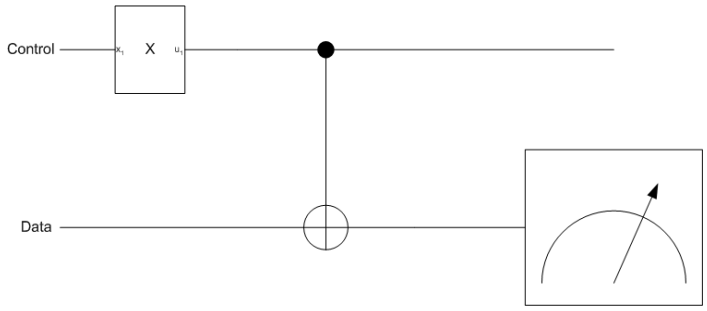


Figure 22: Graphical depiction of a simple quantum computation circuit and algorithm.

In order to implement this circuit, the first step is to ensure that the sample is in thermal equilibrium. Therefore step one of the quantum algorithm for this circuit is to wait for a time sufficient to allow the sample to relax to thermal equilibrium.

Looking from left to right, the next step in the algorithm is the NOT gate. Recall that the NOT gate is implemented by a π pulse about the x-axis. The pulse must be applied at the resonance frequency of the Control qubit, which are the carbon atoms' Larmor frequency.

$$\pi_X^{Control}$$

This has the effect of switching the control bit from $|0\rangle$ to $|1\rangle$. The next step in the algorithm is the controlled-NOT gate.

$$CNOT_A = \left(\frac{\pi}{2}\right)_X^{Data} U_J \left(\frac{1}{2J}\right)\left(\frac{\pi}{2}\right)_Y^{Data}$$

In a somewhat confusing notational conflict, the CNOT gate is usually specified in a pulse propagator notation which is read right to left. This operation is executed in a quantum circuit diagram that is read from left to right.

Therefore the next step in the algorithm is to apply a $\pi/2$ pulse tuned to the Data bit frequency (the hydrogen spins) about the y-axis. The system is then allowed to evolve under the J-coupling for a time corresponding to $1/2J$ and then a $\pi/2$ pulse tuned to the Data bit frequency (again, the hydrogen spins) is applied about the x-axis.

If the algorithm is performed correctly, one should measure the state of the output as expected from the truth table of the circuit. The initial state is $|00\rangle$ using the convention $|Control, Data\rangle$. The control bit is inverted and passed to the controlled-NOT as $|10\rangle$. Recall from the function of the controlled NOT that this implies the data qubit will be inverted and the output should be found to be $|11\rangle$.

The final step in the algorithm is measurement of the Data qubit. At the end of the sequence, this qubit is expected to be in the $|1\rangle$ state, or antiparallel to the z-axis. If the system is left to relax, the macroscopic magnetization of the spins of the hydrogen atoms will return to their thermal equilibrium state and produce a free induction decay (FID) signal. This signal will be digitized and Fourier transformed by the NMR system as described above to produce an output signal as stylized in Figure 23. Since both bits are in the logical one state, there will be two FID frequencies detected.

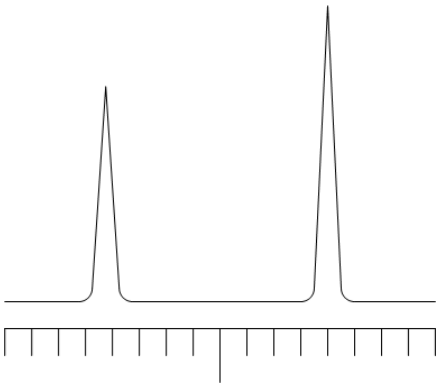


Figure 23: Output signal of a quantum circuit.

In this case, the δ from the reference frequency will include the J-coupling effect since a chloroform molecule is used.

4.2 STILL A CLUMSY, EXPENSIVE TRANSISTOR

What has been implemented so far is still a quantum implementation of a classical system of gates. In the next section, it will be shown that quantum circuits and algorithms can provide a speedup with respect to classical algorithms.

5. QUANTUM ALGORITHMS

The key to realizing a speedup in quantum computing is the ability to extend the available states from the logical 0 and logical 1 of binary logic to superpositions of these states and entangled states. The use of such states leads to a so-called "quantum parallelism," and allows quantum algorithms to evaluate a given function $f(x)$ for many different values of x *simultaneously*.

5.1 DEUTSCH'S ALGORITHM

The simplest quantum algorithm that demonstrates such a speedup is called Deutsch's Algorithm. This is a somewhat contrived example, is relatively simple to explain, but also demonstrates that quantum computing can, in fact, be faster than classical computing.

5.2 DEUTSCH'S ALGORITHM IN THE ABSTRACT

A function $f(x)$ is given with domain $\{0,1\}$ and range $\{0,1\}$ .The function can be one of the constant functions $f(x)=0$ or $f(x)=1$; the identity function $f(x)=x$; or the inverse function $f(x)=x\oplus 1$ where $\oplus$ indicates binary addition.

Although it may seem like a brute force approach, it will be helpful to exhaustively enumerate the possible functions. Thus for the constant function $f(x)=0$, the possibilities are

$$f(0) = 0 \qquad f(1) = 0.$$

For the constant function $f(x) = 1$, the possibilities are

$$f(0) = 1 \qquad f(1) = 1.$$

If the identity function is chosen,

$$f(0) = 0 \qquad f(1) = 1,$$

and the results if the inverse function is chosen are

$$f(0) = 1 \qquad f(1) = 0.$$

The problem to be solved is, while ignoring the definition of the function itself, to determine if the function is "balanced" or not. Balanced is defined by the condition $f(0) = f(1)$ and not balanced is defined by $f(0) \neq f(1)$. From the list above, it is easily seen that the balanced case is possible iff $f(x)$ is a constant function.

5.3 CLASSICAL VERSION OF DEUTSCH'S ALGORITHM

The classical Deutsch algorithm is trivial, but serves the purpose of providing a baseline against which to compare the quantum algorithm. The function $f(x)$ is given as a "black box" and the algorithm goes as follows:

1. Evaluate the unknown function $f(0)$ and save the result;
2. Evaluate the unknown function $f(1)$ and save the result;
3. Compare the result of $f(0)$ with that of $f(1)$ and if equal, indicate Balanced, otherwise Not-Balanced.

The criterion used for evaluating the algorithm is the number of evaluations of $f(x)$. The classical version of the problem requires two evaluations of $f(x)$ as shown above. Deutsch discovered that if qubits were used, the evaluations of the functions could be done in parallel and the algorithmic complexity halved (cf. $O(2)$ vs. $O(1)$ in "big-O notation").

5.4 QUANTUM VERSION OF DEUTSCH'S ALGORITHM

The idea in Deutsch's algorithm is to start with two qubits A and B, one in state $|0\rangle$ representing the input to $f\ 0$, and one in state $|1\rangle$ representing the input to $f\ 1$. Both inputs are fed into $f\ x$ at the same time in superposition.

5.4.1 HADAMARD GATES ON THE WAY IN

Recall that the Hadamard gate is used to create superposition states,

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

The two Hadamard gates use to create the required superposition are typically represented graphically as shown in Figure 24.

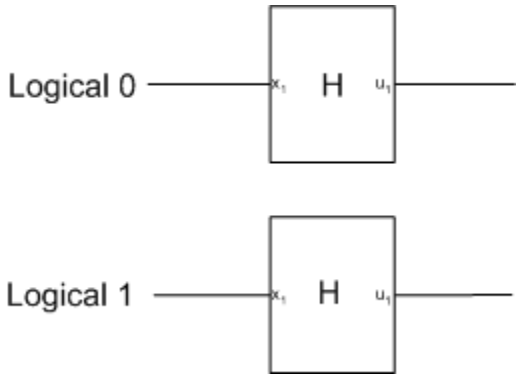


Figure 24: The input "circuitry" for Deutsch's Algorithm.

The outputs of the Hadamard gates are passed on to the inputs of the next step in the algorithm.

5.4.2 CALCULATE THE FUNCTION

The gate that will calculate $f\ x$ has two inputs. One receives the superposition created by $H|0\rangle$ and the other receives the superposition created by $H|1\rangle$. These two superposition states can be viewed as a product state and can be represented in the product basis.

$$H|0\rangle H|1\rangle = \frac{1}{\sqrt{2}}\ |0\rangle + |1\rangle\ \frac{1}{\sqrt{2}}\ |0\rangle - |1\rangle\ = = \frac{1}{2}\ |00\rangle - |01\rangle + |10\rangle - |11\rangle \quad .$$

It is this state that will be fed into the actual computation of $f\ x$. The gate implementing this computation is defined by the map,

$$\left|x_1, x_2\right\rangle \rightarrow \left|x_1, x_2 \oplus f\ \ x_1\ \right\rangle .$$

The simplest thing to do at this point is simply to work out what happens in all of the cases described above. Again, this may seem like a brute force approach, but will pay off shortly. First, assume that $f\ x$ is the constant function $f\ \ x\ = 0$. The results of applying $f\ \ x\ = 0$ to all four possible input states are,

$$\begin{aligned}
\left|00\right\rangle \equiv \left|0,0\right\rangle &\rightarrow \left|0,0 \oplus 0\right\rangle = \left|0,0\right\rangle \equiv \left|00\right\rangle \\
\left|01\right\rangle \equiv \left|0,1\right\rangle &\rightarrow \left|0,1 \oplus 0\right\rangle = \left|0,1\right\rangle \equiv \left|01\right\rangle \\
\left|10\right\rangle \equiv \left|1,0\right\rangle &\rightarrow \left|1,0 \oplus 0\right\rangle = \left|1,0\right\rangle \equiv \left|10\right\rangle \\
\left|11\right\rangle \equiv \left|1,1\right\rangle &\rightarrow \left|1,1 \oplus 0\right\rangle = \left|1,1\right\rangle \equiv \left|11\right\rangle
\end{aligned}$$

The output of $f\ \ x\ $, given the product basis state above as input, is easily seen to be,

$$f\left(\frac{1}{2}\ \left|00\right\rangle - \left|01\right\rangle + \left|10\right\rangle - \left|11\right\rangle\ \right) = \frac{1}{2}\ \left|00\right\rangle - \left|01\right\rangle + \left|10\right\rangle - \left|11\right\rangle ,$$

Observe that the input is passed to the output unchanged.

Now, assume that $f\ \ x\ $ is the constant function $f\ \ x\ = 1$. The results of applying $f\ \ x\ = 1$ to all four possible input states are,

$$\begin{aligned}
\left|00\right\rangle \equiv \left|0,0\right\rangle &\rightarrow \left|0,0 \oplus 1\right\rangle = \left|0,1\right\rangle \equiv \left|01\right\rangle \\
\left|01\right\rangle \equiv \left|0,1\right\rangle &\rightarrow \left|0,1 \oplus 1\right\rangle = \left|0,0\right\rangle \equiv \left|00\right\rangle \\
\left|10\right\rangle \equiv \left|1,0\right\rangle &\rightarrow \left|1,0 \oplus 1\right\rangle = \left|1,1\right\rangle \equiv \left|11\right\rangle \\
\left|11\right\rangle \equiv \left|1,1\right\rangle &\rightarrow \left|1,1 \oplus 1\right\rangle = \left|1,0\right\rangle \equiv \left|10\right\rangle
\end{aligned}$$

The output of $f\ \ x\ $, given the product basis state above as input, is then,

$$f\left(\frac{1}{2}\ \left|00\right\rangle - \left|01\right\rangle + \left|10\right\rangle - \left|11\right\rangle\ \right) = \frac{1}{2}\ \left|01\right\rangle - \left|00\right\rangle + \left|11\right\rangle - \left|10\right\rangle ,$$

Observe that the output is the negative of the input.

If the identity function $f(x) = x$ is assumed, the results are,

$$\begin{aligned}
|00\rangle \equiv |0,0\rangle &\rightarrow |0,0 \oplus 0\rangle = |0,0\rangle \equiv |00\rangle \\
|01\rangle \equiv |0,1\rangle &\rightarrow |0,1 \oplus 0\rangle = |0,1\rangle \equiv |01\rangle \\
|10\rangle \equiv |1,0\rangle &\rightarrow |1,0 \oplus 1\rangle = |1,1\rangle \equiv |11\rangle \\
|11\rangle \equiv |1,1\rangle &\rightarrow |1,1 \oplus 1\rangle = |1,0\rangle \equiv |10\rangle
\end{aligned}$$

The output of $f(x)$, given the product basis state above as input, can be found to be,

$$f\left(\frac{1}{2}\ |00\rangle - |01\rangle + |10\rangle - |11\rangle\ \right) = \frac{1}{2}\ |00\rangle - |01\rangle + |11\rangle - |10\rangle\ ,$$

Observe that the signs of the last two states are reversed and the signs of the first two states are unchanged.

Finally, if the inverse function $f(x) = x \oplus 1$ is assumed, the results are,

$$\begin{aligned}
|00\rangle \equiv |0,0\rangle &\rightarrow |0,0 \oplus 1\rangle = |0,1\rangle \equiv |01\rangle \\
|01\rangle \equiv |0,1\rangle &\rightarrow |0,1 \oplus 1\rangle = |0,0\rangle \equiv |00\rangle \\
|10\rangle \equiv |1,0\rangle &\rightarrow |1,0 \oplus 0\rangle = |1,0\rangle \equiv |10\rangle \\
|11\rangle \equiv |1,1\rangle &\rightarrow |1,1 \oplus 0\rangle = |1,1\rangle \equiv |11\rangle
\end{aligned}$$

The output of $f(x)$, given the product basis state above as input, is seen to be,

$$f\left(\frac{1}{2}\ |00\rangle - |01\rangle + |10\rangle - |11\rangle\ \right) = \frac{1}{2}\ |01\rangle - |00\rangle + |10\rangle - |11\rangle\ ,$$

Observe that the signs of the first two states of the superposition are reversed and the signs of the last two states are unchanged.

Just as the two input states were taken as a product state and viewed in the product basis, the resulting product basis representation can be viewed as a product composed of two output states by factoring the product state.

The representation of the results of the constant function 0 computation are then seen to be,

$$\frac{1}{2}\left|00\right\rangle-\left|01\right\rangle+\left|10\right\rangle-\left|11\right\rangle=\frac{1}{\sqrt{2}}\left|0\right\rangle+\left|1\right\rangle\ \frac{1}{\sqrt{2}}\left|0\right\rangle-\left|1\right\rangle .$$

The representation of the results of the constant function 1 computation are found to be,

$$\frac{1}{2}-\left|00\right\rangle+\left|01\right\rangle-\left|10\right\rangle+\left|11\right\rangle=-\frac{1}{\sqrt{2}}\left|0\right\rangle+\left|1\right\rangle\ \frac{1}{\sqrt{2}}\left|0\right\rangle-\left|1\right\rangle .$$

The identity function representation becomes,

$$\frac{1}{2}\left|00\right\rangle-\left|01\right\rangle-\left|10\right\rangle+\left|11\right\rangle=\frac{1}{\sqrt{2}}\left|0\right\rangle-\left|1\right\rangle\ \frac{1}{\sqrt{2}}\left|0\right\rangle-\left|1\right\rangle ,$$

and the inverse function

$$\frac{1}{2}-\left|00\right\rangle+\left|01\right\rangle+\left|10\right\rangle-\left|11\right\rangle=\frac{1}{\sqrt{2}}-\left|0\right\rangle+\left|1\right\rangle\ \frac{1}{\sqrt{2}}\left|0\right\rangle-\left|1\right\rangle .$$

Observe that the second state term in each of the four cases is identical,

$$\frac{1}{\sqrt{2}}\left|0\right\rangle-\left|1\right\rangle .$$

Therefore, the useful information is found only in the first state. This is not at all obvious from the definition of the function.

5.4.3 HADAMARD GATE ON THE WAY OUT

Recall that the Hadamard gate is defined by the matrix,

$$H\equiv\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}.$$

Continuing the so-far successful approach of brute force calculation, the four states identified above as carrying the useful information can operated upon by a Hadamard gate. First, the state that resulted from choosing the constant function 0,

$$H\frac{1}{\sqrt{2}}\left|0\right\rangle+\left|1\right\rangle=H\frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}=\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}=\frac{1}{2}\begin{bmatrix}2\\0\end{bmatrix}=\begin{bmatrix}1\\0\end{bmatrix}=\left|0\right\rangle .$$

The state that resulted from choosing the constant function 1 is transformed into,

$$H\frac{1}{\sqrt{2}}\ -|0\rangle-|1\rangle\ = H\frac{1}{\sqrt{2}}\begin{bmatrix}-1\\-1\end{bmatrix}=\frac{1}{\sqrt{2}}\begin{bmatrix}1&1\\1&-1\end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix}-1\\-1\end{bmatrix}=\frac{1}{2}\begin{bmatrix}-2\\0\end{bmatrix}=-\begin{bmatrix}1\\0\end{bmatrix}=-|0\rangle.$$

The state that resulted from choosing the identity function is transformed into,

$$H\frac{1}{\sqrt{2}}\ |0\rangle-|1\rangle\ = H\frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}=\frac{1}{\sqrt{2}}\begin{bmatrix}1&1\\1&-1\end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}=\frac{1}{2}\begin{bmatrix}0\\2\end{bmatrix}=\begin{bmatrix}0\\1\end{bmatrix}=|1\rangle.$$

Finally, the state that resulted from choosing the inverse function is transformed into,

$$H\frac{1}{\sqrt{2}}\ -|0\rangle+|1\rangle\ = H\frac{1}{\sqrt{2}}\begin{bmatrix}-1\\1\end{bmatrix}=\frac{1}{\sqrt{2}}\begin{bmatrix}1&1\\1&-1\end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix}-1\\1\end{bmatrix}=\frac{1}{2}\begin{bmatrix}0\\-2\end{bmatrix}=-\begin{bmatrix}0\\1\end{bmatrix}=-|1\rangle$$

Recall that it was determined at the start of this section, that the balanced case is possible only if $f\ x$ is a constant function. Notice that it has just been shown that the output of the final Hadamard gate is in state $\pm|0\rangle$ if the function $f\ x$ is one of the constant functions and is in state $\pm|1\rangle$ otherwise.

Therefore if the final state is measured to be in state $|0\rangle$ the functions are balanced and if the final state is measured to be in state $|1\rangle$ the functions are not-balanced. This completes the quantum version of Deutsch's algorithm.

5.5 IMPLEMENTATION OF QUANTUM VERSION

A quantum circuit for an implementation of Deutsch's Algorithm is shown in Figure 25.

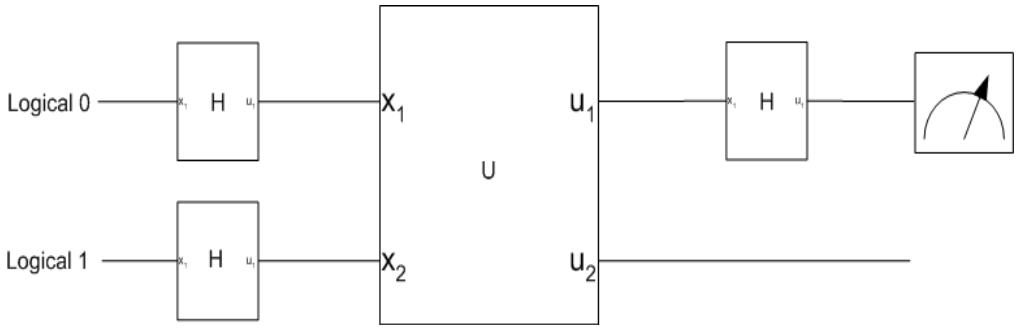


Figure 25: Quantum circuit for Deutsch's Algorithm.

As in the simple circuit described above, this circuit is read from left to right. At the left side, notice that a "Logical 0" qubit and a "Logical 1" qubit are inputs. This is therefore a two qubit system and the underlying NMR implementation needs to have a sample in which there are two spins that can be individually manipulated. Also as described above,

chloroform, $CHCl_3$, contains both carbon and hydrogen atoms and could provide the necessary addressable bits.

Since the initial state will be defined as the thermal equilibrium state, the spin excess will be in the spin up state. Recall the previously made definition:

$$|0\rangle \equiv \left|\frac{1}{2}\ \left(+\frac{1}{2}\right)\right\rangle$$
$$|1\rangle \equiv \left|\frac{1}{2}\ \left(-\frac{1}{2}\right)\right\rangle .$$

Using this definition, both qubits will be in state $|0\rangle$ at thermal equilibrium. In order to resolve the bit initialization discrepancy, one can either assume an additional NOT operation performed on the initial state of the spin representing Logical 1, or one can reverse the sense of the logic for that bit. This will be taken as resolved in the following discussion since the representation of the quantum circuit for Deutsch's algorithm is typically rendered as in Figure 25.

In order to implement this circuit, then, the first step in the algorithm is then to ensure that the sample is in thermal equilibrium by waiting for a time sufficient to allow the sample to relax to equilibrium.

The next step in the algorithm is then to apply the Hadamard gates to the input bits using two $\pi/2$ pulses tuned to the individual Larmor frequencies of the atoms representing the two bits. The upper input is defined to be bit A, and the lower input is defined to be bit B. The application of Hadamard gates to multiple bits is so common in quantum computing that it is given a specal symbol $H^{\otimes N}$ where N indicates the number of bits. For the purposes of this circuit,

$$H^{\otimes 2} = \left(\frac{\pi}{2}\right)_Y^A \left(\frac{\pi}{2}\right)_Y^B .$$

Recall the definition of the next gate, the gate used to calculate $f\ \ x$, was given by a map,

$$|x_1, x_2\rangle \rightarrow |x_1, x_2 \oplus f\ \ x_1\ \rangle .$$

Notice that the controlled-NOT gate was also defined quite similarly,

$$CNOT_a\,|a,b\rangle = |a, a \oplus b\rangle .$$

This can be rewritten in a suggestive way,

$$CNOT_a \left| x_1, x_2 \right\rangle = \left| x_1, f\ \ x_1\ \oplus x_2 \right\rangle .$$

Since Deutsch's algorithm explicitly says that the function $f\ \ x$ is not known to the algorithm, it is reasonable to choose the identify function $f\ \ x\ = x$ and make this substitution.

$$CNOT_a \left| x_1, x_2 \right\rangle = \left| x_1, x_1 \oplus x_2 \right\rangle .$$

It can now be seen that the perhaps odd-looking map seen above as defining $f\ \ x$ is simply a controlled-NOT gate. The behavior of a controlled-NOT gate obviously becomes much more interesting when supplied with superpositions of bits. It was shown above that the NMR sequence for a controlled-NOT gate with input A as the control qubit is,

$$CNOT_A = \left(\frac{\pi}{2} \right)_X^B U_J \left(\frac{1}{2J} \right) \left(\frac{\pi}{2} \right)_Y^B$$

The last step in the Deutsch algorithm is the final Hadamard gate applied to the upper qubit to resolve the superposition,

$$H^{\otimes 1} = \left(\frac{\pi}{2} \right)_Y^A .$$

The NMR pulse sequence to implement Deutsch's algorithm is then,

$$\left(\frac{\pi}{2} \right)_Y^A \left(\frac{\pi}{2} \right)_X^B U_J \left(\frac{1}{2J} \right) \left(\frac{\pi}{2} \right)_Y^B \left(\frac{\pi}{2} \right)_Y^A \left(\frac{\pi}{2} \right)_Y^B$$

If a measurement is made on qubit A at the end of this pulse sequence, and a free induction decay is observed, the output of the quantum circuit will have been observed to have been in state $|1\rangle$ which is the indication that the functions are "not balanced." This would be the expected result since $f\ \ x$ was defined to be the identity function.

Notice that although this may seem fiendishly complicated, there was indeed only one evaluation of $f\ \ x$ required and it was done in parallel for both $f\ \ 0$ and $f\ \ 1$.

6 CONCLUSION

At the start of this paper it was observed that quantum computing is really in the very beginning stages of both research and development. A comparison was made between the original point contact transistor of Bardeen and Brattain and a modern Intel microprocessor. At the time at which the transistor was being developed there were only hints of what could be done. You will probably agree that putting together an NMR implementation of the Deutsch algorithm feels a lot more like soldering together wires and "rocks" to make something that may be useful in the future than it does working with a modern Linux computing cluster.

Figure 26: Wires and rocks versus modern computers.

Just as there were hints at the future possibilities of solid state electronic devices in 1947, there are hints at the possibilities of quantum computers today. Nobody knows if the various implementations of quantum computers today will evolve into the quantum equivalent of the modern computing cluster, but perhaps the future equivalent of a Fairchild Semiconductor will appear soon to begin the process.

References

[1] Cohen-Tannoudji C., et al. Quantum Mechanics, Hermann, Paris (1977)

[2] Griffiths D., Introduction to Quantum Mechanics, Prentice Hall, Upper Saddle River (2005)

[3] Neilsen M. et al.,Quantum Computation and Quantum Information, Cambridge, Cambridge (2000)

[4] Oliveira I. et al., NMR Quantum Information Processing, Elsevier, Amsterdam (2007)

[5] Sakurai J., Modern Quantum Mechanics, Addison-Wesley, Reading (1994)

[6] Shankar R., Principles of Quantum Mechanics, Springer, New York (1994)