Chapter 7

PROCESSORS

INTRODUCTION

Theoretical investigations concerning quantum algorithms, automata, complexity, information theory and in cryptography are of great interest and importance. However, progress in the experimental efforts to design quantum information-processing systems is crucial for seeing properly the overall perspectives of the future designs of real and powerful quantum computers, and for isolating and solving the problems that need to be dealt with if powerful quantum computers are ever to be built.

It has been realized, from the very early days of research in quantum computing, at least by some, that powerful evolution of isolated quantum systems is hard to utilize in real quantum processors, because of their interaction with the environment that can destroy very large but fragile quantum superpositions; and because of the natural imperfections of (inherently analogue) quantum devices. In addition, quantum error correction was considered impossible.

Fortunately, several developments brought the vision of quantum computers closer to reality. Quantum computation stabilization methods and quantum error correction codes have turned out to be possible and efficient. Techniques for fault-tolerant quantum computing have been developed. Finally, some promising technologies to design quantum gates, circuits, and processors have been identified and are being experimentally tested.

LEARNING OBJECTIVES

The aim of the chapter is to learn:

1. the early proposals for quantum computers;
2. the impacts of imprecision, dissipation and decoherence on quantum computing;
3. the methods of designing and using quantum error-correcting codes;
4. the basics of quantum fault-tolerant techniques;
5. the basics of the main current technologies used, or considered, to develop experimental quantum processors.

In theory there is no difference between theory and practice, but in practice, there is.

Theoretical developments concerning quantum algorithms, automata, complexity, information and cryptography have been accompanied by an intensive theoretical as well as experimental effort to identify and overcome problems that need to be dealt with in order to design successfully (experimental) quantum processors.

A real quantum computer is just a physical system whose evolution can be interpreted as performing some specific quantum computation. In order to design such systems, frontiers of current technology have to be explored and insights have to be developed into the essence of such crucial problems as imprecisions during quantum computations and decoherence, which occurs when quantum information is sent, in time or space, through a noisy quantum channel. These problems are either specific for quantum computing or have in this case a very different nature than in the classical computing. Finally, methods have to be developed to deal with these problems. Some of such methods can be seen as quantum generalizations of classical ones, but also methods had to be developed that have no classical analogue.

Progress has been achieved in studying the main obstacle to practical quantum computing — decoherence — and formidable successes have been achieved in developing quantum computing and storage-stabilizing techniques: quantum error-correcting codes, entanglement purification (or distillation) techniques, and so on. In addition, fault-tolerant techniques have already been developed for quantum computing. It is clear today that arbitrarily long quantum computations can be performed reliably, in principle, provided that the average probability of error per quantum gate is less than a certain threshold. Therefore, internal imprecisions and external decoherence do not have to be any longer considered as an obstacle to quantum computation we would not be able to cope with. Theoretical results indicate that it might even be possible to build inherently fault-tolerant quantum hardware. All that makes the vision of real quantum computers much closer than expected a few years ago.

Significant progress has been achieved in the experimental development and testing of several technologies that seem to have a potential use in the design of small experimental quantum processors.

The main theoretical developments behind the design of experimental quantum processors as well as the main principles of technologies being currently explored for this purpose are discussed in this chapter.

7.1 Early Quantum Computers Ideas

The first four historically important attempts to describe how a quantum computer could function are briefly summarized in this section. They also illustrate a shift from the first half-classical/half-quantum Turing machine model of Benioff, through Feynman's universal quantum simulator—a quantum version of Boolean circuits—to a fully quantum Turing machine model of Deutsch.

7.1.1 Benioff's quantum computer

Benioff (1982, 1982a) made the first attempts to develop "quantumized" Turing machines. In his models the tape of a QTM consists of a sequence of qubits (spin states), each one being in one of the basis states $|0\rangle$ or $|1\rangle$ after each elementary computation step. This way a binary input and intermediate results were encoded. The "head" of a TM had the form of a device that could change the values of qubits. Finite control, with the program to be performed, was implicitly contained in the Hamiltonian of the carefully designed Schrödinger equation.

During each computational step Benioff's computer could be in a superposition of quantum states, but at the end of each step the tape had to be in a basis state representing a sequence of classical states and no characteristically quantum property of the model (superposition of basis states, parallelism, entanglement) could be detected. His model was developed within the framework of quantum kinematics and dynamics, but a classical Turing machine could simulate it effectively and easily.

There have been several variants of Benioff's model. A major problem with the early models was that in order to design a Hamiltonian to mimic a particular Benioff's computer, not only the program but also the complete set of computation paths had to be given. In other words, "one would need to know the answer to the problem one had to compute in order to design a Benioff's computer to solve it".

The main implementation problem of Benioff's computer was how to realize an interaction between the head and qubits that could be far apart.

7.1.2 Feynman's quantum computer

Feynman (1982, 1986) took at first glance a very simplified approach to quantum computing. His approach was based on the understanding, due to the results of Fredkin and Toffoli (1982), that there are universal reversible Boolean gates. His main contribution was a general method to design the Hamiltonian for a quantum circuit in which each gate performs a unitary operation.

At first it seemed (see Deutsch, 1985) that Feynman's approach is a restrictive one. "Programming" of his simulator actually means designing a new quantum circuit and then setting its inputs. However, due to Yao's (1993) result, it is now clear that from the computational power point of view the quantum circuits model has the same power as quantum Turing machines.

Let us assume that we have a quantum circuit consisting of a serial connection of k quantum gates, each performing a unitary operation, say $U_1, \ldots, U_k$. Feynman dealt with such a case in its full generality and showed the form a Hamiltonian H in the Schrödinger equation

$$U(t) = e^{-\frac{i}{\hbar}Ht}$$

should have for such a circuit. Feynman thereby found a systematic, though not very efficient, way to transform a quantum circuit description of a quantum computer to the dynamical Schrödinger equation that simulates computation steps of the circuit.

Feynman used extra k qubits, for so-called "program counter sites" as well as the "creation" operators c_i $(i = 1, \ldots, k)$ and the "annihilation operators" a_i $(i = 0, \ldots, k-1)$. Each creation operator c_i "sets the ith counter qubit to 1", the annihilation operator to 0. The overall Hamiltonian has then the form

$$H = \sum_{i=0}^{k-1} (c_{i+1} \cdot a_i \cdot U_{i+1} + (c_{i+1} \cdot a_i \cdot U_{i+1})^*)$$

where "·" denotes the product of matrices. The first terms take care of a sequential execution of all gates. (One needs to add the conjugate terms because the resulting Hamiltonian has to be Hermitian.) In total, Feynman's approach needs a register with $m + k - 1$ qubits to deal with k counters and m input qubits.

More exactly,

$$c = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \qquad a = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

and c_i (a_i) is just c (a) applied to the ith counter qubit. (Observe that c maps $|0\rangle \to |1\rangle$ and $|1\rangle$ to a "null state"; similarly a maps $|1\rangle \to |0\rangle$ and $|0\rangle$ to a "null state".)

Computation on such a circuit begins by putting the input bits into the input register and the pointer to occupy the site 0. One then checks, at site k, that site k is empty, or that the site has a pointer. Once the cursor is found, it is removed so that it cannot return down the program line. At that moment the register contains all outputs that just need to be measured. Termination is not taken care of by such a quantum computer itself. It has to be decided from the outside when a measurement is to be performed.

In Feynman's model all of the quantum uncertainty of the computation is concentrated in the time needed for computation to be completed, and not at all in the correctness of the outcomes. Namely, if a computation is done, and a certain bit indicates it, the result obtained is always correct.

7.1.3 Peres' quantum computer

Peres (1985) improved some of Benioff's and Feynman's ideas on how a quantum computer should be designed. For example, he improved Feynman's model in such a way that the result of the calculation shows up at the precise time and place. In spite of the fact that his model, similarly to Benioff's, is effectively classical—inherently quantum phenomena (as superposition) occur only in the dynamics and not in the logic of the computer—Peres has already brought up the idea of having a computer "continuous quantum logic"—with states being a superposition of basis states.

Like all previous attempts to describe a quantum computer, Peres concentrates on Hamiltonians: how to write a Hamiltonian H for a quantum system such that the time evolution $e^{-\frac{iHt}{\hbar}}$ represents the execution of a computation. The main new contribution is an analysis of possible errors (in the Hamiltonian and in measurements) and especially an idea of how to use quantum error-correcting codes and how to incorporate into Hamiltonians quantum error correction procedures.

Peres has considered the case that one qubit is encoded by three ($|0\rangle \to |000\rangle$, $|1\rangle \to |111\rangle$) and the potential error states have the form

$$\alpha_0|000\rangle + \beta_0|100\rangle + \gamma_0|010\rangle + \delta_0|001\rangle$$

or

$$\alpha_1|111\rangle + \beta_1|011\rangle + \gamma_1|101\rangle + \delta_1|110\rangle,$$

for suitable amplitudes $\alpha_i, \beta_i, \gamma_i, \delta_i$, $i = 0, 1$.

Peres also considered a way of using Stern–Gerlach magnets for error detection and error correction. In addition, he realized that errors can be corrected by unitary operations.

Finally, he has considered ways error correction can be incorporated in the Hamiltonian, so that the probability of error can be made arbitrarily small.

In spite of the fact that Peres' error-correcting code is not good enough, see page 278, because it does not use entanglement to protect quantum information, it was the first attempt to consider quantum error-correcting codes.

7.1.4 Deutsch's quantum computer

Deutsch's basic philosophical standpoint was that underlying the Church–Turing principle there is an implicit physical assertion. On this base he developed the so-called physical version of the Church–Turing principle:

> Every finitely realizable[1] physical system can be perfectly simulated by a universal model computing machine operating by finite means.

Deutsch (1985) presented a general, fundamentally new, and fully quantum model of quantum computation. The tape ($\mathbf{t}$) of Deutsch's Turing machine $\mathcal{U}$ consists of an infinite sequence of qubits and its finite control consists of a finite sequence of qubits ($\mathbf{m}$). In addition there is an observable $\mathbf{x}$, which has any integer from $\mathbf{Z}$ as its potential value—a pointer to the currently scanned tape cell. Deutsch deals with the problem of infinitely long tape by assuming that tape is not rigid and there is a mechanism that can move the tape according to signals transmitted at finite speed between adjacent segments. The state of the quantum computer $\mathcal{U}$ is therefore a unit vector in the space spanned by basis vectors $|\mathbf{x}, \mathbf{t}, \mathbf{m}\rangle$.

The dynamics of $\mathcal{U}$ is given by a constant unitary operator U and for the evolution of the state $|\psi(t)\rangle$ it holds

$$|\psi(t)\rangle = U^t|\psi(0)\rangle \quad \text{where}|\psi(0)\rangle = \sum_n \lambda_n|0, 0, \mathbf{t}\rangle,$$

and only finitely many λ_i are non-zero if an infinite number of elements in $\mathbf{t}$ are non-zero. U has to satisfy a special condition in order to perform operations "by finite means".

Deutsch was fully aware of such features of quantum computing as quantum parallelism and entanglement. To explain these features he used Everett's many-world interpretation, Section 9.1.7, because, as he explains, "the intuitive explanation of these properties places an intolerable strain on all interpretations of quantum theory other than Everett's".

Deutsch also described the universal quantum computer capable of simulating every finitely realisable physical system, therefore also any other quantum computer, with arbitrarily high precision. To design the universal quantum computer Deutsch made use of the fact that if α is any irrational multiple of π, then the four transformations

$$\begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}, \begin{pmatrix} \cos\alpha & i\sin\alpha \\ i\sin\alpha & \cos\alpha \end{pmatrix}, \begin{pmatrix} e^{i\alpha} & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & e^{i\alpha} \end{pmatrix}.$$

form a group dense in the group of all unitary transformations in H_2.

[1] By "finitely realizable physical system" any physical object is meant upon which experimentation is possible.

7.2 Impacts of Imperfections

There are two main sources of imperfections that may influence computations on real quantum processors and which one may ignore when ideal quantum computation is considered: internal and external.

Internal imperfections have to be seen more as a rule than an exception: they are imperfections in the setting of the initial state (input data), imperfections of the quantum evolution itself (of the gate operations), and, finally, imperfections of the final measurement. Internal imperfections can be seen as "hardware imperfections".

As we shall see, these imperfections do not influence quantum computing dramatically. Actually, with respect to the imperfections of this type quantum processors are much more robust than classical ones, in which impacts of internal imperfections grow exponentially.

On the other hand, the external imperfections, due to decay (dissipation), and, especially, decoherence, are much more serious and they keep threatening the very basis of existence of the successful and powerful quantum computers.

7.2.1 Internal imperfections

At first we explore impacts of errors that one can expect to occur, with high probability, in the preparation and execution of quantum computations on real quantum computers.

There is a variety of reasons why such imperfections are practically inevitable, directly opposite to the situation in classical computing. The main reason is that in all three main stages: setting of the initial state; quantum evolution; and final measurements; one has to set up complicated, sensitive and in the essence analogue devices about which there is practically no likelihood of absolute precision. Fortunately, these imprecisions are not fatal for quantum computing.

Imprecisions in the initial state

The initial state $|\psi_0\rangle$ for the quantum evolution $|\psi(t)\rangle = U^t|\psi_0\rangle$, may be one of the basis states, or a quantum superposition of such states. To set up such a state various sensitive devices (i.e. polarizers, magnets, devices for timing of laser pulses in the ion trap computers) have to be used. Their setting and performance may be a source of errors. The question is now how much such errors propagate and accumulate during a quantum evolution—how much is quantum evolution stable with respect to the initial uncertainties. In other words, how much can we trust the results of quantum evolution in the case of errors in the initial state?

Fortunately, due to the linearity of quantum evolution, such errors do not grow with time. To illustrate this case let us consider the situation that instead of the initial state $|\psi_0\rangle = |0\rangle$ we have the state $|\psi_{err}\rangle = \sqrt{1-\varepsilon^2}|0\rangle + \varepsilon|1\rangle$. After t computation steps we have

$$U^t(\sqrt{1-\varepsilon^2}|0\rangle + \varepsilon|1\rangle) = \sqrt{1-\varepsilon^2}U^t(|0\rangle) + U^t(|1\rangle) = \sqrt{1-\varepsilon^2}|\psi(t)\rangle + \varepsilon U^t(|1\rangle)$$

For small ε the first term is almost the state $|\psi(t)\rangle$ and the second one can be negligible. (On the other hand, many quantum algorithms are so sensitive to small changes in the amplitudes of superpositions that the above result should not be overestimated.)

Operational imperfections

Another source of imperfections, that results in the errors in the Hamiltonian of the system and consequently in the quantum evolution, is an unavoidable inaccuracy of quantum computer components. This is basically due to the fact that quantum computer components are mainly analogue type devices. As a consequence, the state of a quantum superposition depends on several continuous parameters. For example, gates very often used are those performing a rotation by an angle θ. If such a gate is applied there is naturally some inaccuracy in θ. Errors of this type are caused by unitary transformations, actually by over-rotations or under-rotations.

There are several ways to estimate impacts of operational imperfections.

In Section 4.2.3 we have seen, by analyzing imprecisions of Turing machine computations, Theorem 4.2.21, that imprecisions during computations only add and do not grow exponentially. In addition, it was shown that $\mathcal{O}(\lg t)$ bits of precision in transition amplitudes are sufficient to support t steps of a QTM with required precision.

Another way to approach the problem is to consider the case that instead of correct Hamiltonian H there is slightly different Hamiltonian $H' = H + H_{err}$. An analysis by Williams and Clearwater (1997) shows that errors grow at most quadratically in time.

Imperfections in the measurement

Finally, let us discuss the last case, namely that we do not know exactly when the result of a quantum computation is ready (as at the Grover's search algorithms), and we have to perform n measurements at the equally spaced time intervals.

An analysis by Williams and Clearwater (1997) implies that more measurements will not make computation less secure. This is, however, only a partial result. A deeper and more general analysis of imperfections due to measurements is still to be done.

7.2.2 Decoherence

Two essential properties of quantum information processing that efficient quantum algorithms essentially exploit are the existence of quantum superpositions and entanglement—non-local correlations between different parts of physical systems. It is an elementary but fundamentally important fact that both of them are ultrasensitive to interactions with the environment. The enormous fragility of quantum states used to process information is the main problem for any design of quantum processors.

In order to perform a successful quantum computation one has to maintain a coherent unitary evolution until the completion of the computation. However, technologically it is not possible to ensure that a quantum register is completely isolated from the environment.

There are several reasons why it is practically not possible to avoid interactions of a real quantum computer with the environment. Quite sophisticated technology has to be used to create quantum registers, and this infrastructure technology cannot be fully isolated from the register. Secondly, there is an unavoidable coupling of quantum systems with the thermal environment. (Even such phenomena as cosmic rays may have their impacts.)

There are several ways interactions of quantum systems with their environment materialize. For example, the **decay** (**dissipation**) is a process by which a quantum system dissipates energy into the environment. For example, if an excited (high energy) state is to represent $|1\rangle$ in a qubit and the lower energy state represents $|0\rangle$, then the system (qubit) can, spontaneously, make a transit from $|1\rangle$ to $|0\rangle$, emitting a photon in the process. A qubit flip (from $|1\rangle$ to $|0\rangle$ only), is then the net result of such a spontaneous transition.

The **decoherence** is a general term for process of coupling of a quantum system (processor) with its environment, if it is not perfectly isolated. As a consequence, quantum state of the system is modified due to the interactions of the system with the environment. Such interactions mean that the quantum dynamics of the environment is also relevant to the operations of the quantum computer and its states become entangled with the states of the environment. The effect of such an entanglement with the environment can be seen as if the environment applied a measurement to the system. This has destructive impacts on superpositions and interference. Decoherence tends to destroy irreversibly information in a superposition of states in a quantum computer in a way we cannot control. It spoils both constructive and destructive interferences that are essential for quantum computing and long computations seem to be impossible. As the result, quantum information initially encoded in a register becomes encoded instead in the correlations between the quantum computer and its environment and we can no longer access the information by observing only the computing device. Decoherence can therefore be seen as a physical process, in which quantum systems lose, due to their interaction with environment, some of their key quantum properties of importance for being able to have quantum information processes more efficient than classical ones.

Especially delicate are entangled states. Making a measurement on an entangled state will usually cause a collapse of it to a less entangled state. Small interactions with the environment provide a sort of continuous measurement of the system. As a quantum system gets larger this is harder and harder to ignore. System starts to decohere more and more and starts to look and behave more and more as a classical system. Decoherence is why quantum world looks classical at the human level (Gottesman, 1997).

Decoherence is the most fundamental obstacle so far preventing the design of real quantum computers. For example (see Barenco, 1996), the effects of decoherence in time on the qubit state $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$, represented by the density matrix

$$\rho_{|\psi\rangle} = |\phi\rangle\langle\phi| = \begin{pmatrix} |\alpha|^2 & \alpha\beta^* \\ \alpha^*\beta & |\beta|^2\rangle \end{pmatrix},$$

can be discussed by the time-dependent density matrix

$$\rho = \begin{pmatrix} |\alpha|^2 & e^{-\frac{t}{\tau}}\alpha\beta^* \\ e^{-\frac{t}{\tau}}\alpha^*\beta & |\beta|^2 \end{pmatrix},$$

in which τ is so-called decoherence time. Matrix ρ_τ converges to the diagonal matrix

$$\begin{pmatrix} |\alpha|^2 & 0 \\ 0 & |\beta|^2 \end{pmatrix}.$$

Example 7.2.1 *(Barenco, 1996) Decoherence can affect the probability distribution of the possible outcomes of computations. In order to demonstrate that let us consider the situation that the Hadamard matrix H is applied to the state $|0\rangle$ of a qubit twice. We get*

$$|0\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \xrightarrow{H} |0\rangle. \tag{7.1}$$

If (7.1) is reformulated in terms of the corresponding density matrices we see this evolution as follows:

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \xrightarrow{H} \frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \xrightarrow{H} \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}. \tag{7.2}$$

If there is no decoherence, the measurement of the final state yields the outcome $|0\rangle$ *with probability* 1. *Let us now assume that decoherence occurs between the two applications of the operator* H *and annihilates completely off-diagonal elements of the second matrix in (7.2).*[2] *Instead of (7.2) we get the evolution*

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \xrightarrow{H} \frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \xrightarrow{decoherence} \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \xrightarrow{H} \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

At the measurement we get both $|0\rangle$ *and* $|1\rangle$ *with probability* $\frac{1}{2}$.

It is clear that decoherence is a major problem. It is less clear how really big a problem it is and whether we can successfully deal with it.

The first estimations of decoherence impacts were very pessimistic. The basic message was that decoherence is so large that the probability of getting a correct result decreases exponentially. Indeed, the error rate due to decoherence is time-dependent and approximately modeled by the function

$$1 - e^{-\frac{t}{\tau_{dec}}},$$

where τ_{dec} is the so-called decoherence time. Therefore, by Barenco (1996), if τ_{dec} is the typical decoherence time for a single qubit, the probability of getting the correct result for a quantum computation with the input of size n is

$$P \approx P_0 e^{-is(n)t(n)/\tau_{dec}},$$

where P_0 is the probability of the result with no errors, $s(n)$ is the total number of qubits necessary to perform the computation and $t(n)$ is the time needed to perform the computation. This implies, for example, that the decoherence problem cannot be efficiently dealt with by simply increasing the number of runs.

It has slowly turned out that a more detailed analysis of the decoherence problem can bring a different and less pessimistic view of decoherence, in some situations at least. For example, the decoherence problem is usually considered under the assumption that each of the qubits interacts with a different environment. However, if a different assumption is taken, for example that all qubits interact with the same environment, then it can be shown (Duan and Guo, 1996), that for some entangled initial states no decoherence occurs at all.

Table 7.1, due to DiVincenzo (1995), displays estimates for gate switching time t_s, decoherence time τ_{dec}, as well as the number of steps that can be performed without losing coherence, for several technologies of potential interest.

Experiments indicate (see DiVincenzo and Terhal, 1998), that using a single trapped beryllium ion, decoherence time is about 1 ms and with NMR technology around 1 s.

How to fight decoherence? Firstly, one should try to use technologies with low decoherence time. It is believed that there is still much to discover along these lines.

Secondly, it is well known that under certain circumstances decoherence is smaller. For example, some systems need more time to decohere at very low temperatures.

Ingenuity of experimental physicists is expected to bring significant improvements concerning decoherence. However, the main way to fight decoherence seems to be indirect one —a "software approach"—to try to avoid damage caused by decoherence by undoing its bad effects using, similarly as in the classical case, quantum error correcting codes, and other methods presented in Section 7.4.

[2]In general, decoherence is often a process that eliminates off-diagonal elements of the density matrix of mixed states.

Quantum system	t_s	τ_{dec}	comput. steps
Mössbauer nucleus	10^{-19}	10^{-10}	10^9
GaAs electrons	10^{-13}	10^{-10}	10^3
Au electrons	10^{-14}	10^{-8}	10^6
Trapped indium ions	10^{-14}	10^{-1}	10^{13}
Optical microcavity	10^{-14}	10^{-5}	10^9
Electron spin	10^{-7}	10^{-3}	10^4
Electron quantum dot	10^{-6}	10^{-3}	10^3
Nuclear spin	10^{-3}	10^4	10^7

Table 7.1: Switching time t_s, decoherence τ_{dec}, both in seconds, and the number of computation steps performed before decoherence impacts occur

The third main source of environmental impacts is in some sense inverse to the first one. They are impacts of external forces such as cosmic rays, or residual gas molecules, that can hit a qubit and change its state. These impacts are much out of our control.

7.3 Quantum Computation and Memory Stabilization

Efficient methods for the stabilization of quantum computations and memory are of crucial importance for computing and communication.

Redundancy is the main idea behind all methods of stabilization. In classical computing redundancy is utilized through redundant computations combined with the majority voting stabilization or by redundant coding using error detection and correction codes. Unfortunately, none of these methods can be used directly in the quantum setting.

The facts that quantum bits cannot be measured, without risking the collapse of their states, and cannot be copied, bring a completely new situation, compared with the classical case, into the problem of quantum error detection and correction. Even where we know what the state of a quantum bit should be we cannot test whether this is the case.

There is also no direct way to use the majority voting method. Indeed, even it is the case the states $|\phi_1\rangle, \dots, |\phi_n\rangle$ are all equal, any measurement of them, with respect to an observable for which they are not eigenvectors, can each time give a different result. A majority voting method in quantum setting cannot therefore be used either to determine the majority state or to reset computations to the correct state. Because of that the idea of quantum error correcting codes looked hopeless. However, by utilizing in a novel way quantum entanglement, the field of quantum error correcting codes developed excellently.

In this section we present a sort of quantum analogue of the majority voting method to make use of the redundancy for quantum computation stabilization in a way that has no parallel in classical computing.

Before presenting the method itself there is still one point to discuss concerning stabilization: efficiency. Only such computation/memory stabilization methods are of interest that are efficient enough. Computations which require an exponentially increasing precision, or exponential amount of time, space or energy or of some other physical resource are considered as unfeasible. The same is true for polynomial resource-bounded computations, provided they do not have polynomial resource-ounded stabilization methods. From a quantum computing/memory stabilization technique it is therefore required that it makes polynomial resource-bounded computations out of any polynomial time-bounded computations.

Another fact of importance in this connection is that in some cases, for example for decoherence, error probability grows exponentially with the size of input. It is therefore of vital importance to determine whether we can have quantum stabilization that would make polynomial algorithms stay polynomial even under such unfavourable circumstances.

The majority voting method, brought to classical computing by von Neumann, is very successful in classical computing for a very simple reason: it is extremely good. If r computations are performed and each has probability $1-\theta$ of being successful, then by Chernoff bound the probability that the method fails is less than $e^{-\theta^2 r/6}$ and therefore the probability decreases exponentially with the number r of redundant computations.

Such a performance we can hardly expect from a quantum computation stabilization method. Fortunately, as will be shown, even much more modest stabilization methods may still be useful. Indeed, let us have a polynomial time algorithm that performs each step of the computation correctly with probability $1-\varepsilon$. After t steps the probability of successful computation is $(1-\varepsilon)^t \approx e^{-\varepsilon t}$. Suppose now that we have a computation stabilisation method available which, using redundancy r, reduces the error in each step modestly, by the factor $\frac{1}{r}$. After t steps the probability of success is $e^{-\frac{\varepsilon t}{r}} \approx 1-\delta$ if $r = \frac{\varepsilon t}{-\lg(1-\delta)}$, for any $\delta > 0$. In addition, r is polynomial in t and therefore also in the size of input.

We can therefore see that even with a moderately successful stabilization method, as the one presented in this section, which is due to Barenco et al. (1997), we can stabilize exponentially growing errors using only polynomial time computations.

7.3.1 The symmetric space

The very basic idea behind stabilization of computation by symmetrization is simple. In the case of r redundant computations in a Hilbert space H the resulting composed state will be, in the ideal error-free case, $\bigotimes_{i=1}^r |\psi\rangle$, i.e. a symmetric state that surely belongs to the smallest subspace of $H^{(r)} = \bigotimes_{i=1}^r H$ containing all such states. In the case of any erroneous computation, the state after r computations can be $\bigotimes_{i=1}^r |\psi_i\rangle$, with all $|\psi_i\rangle$ different, that is not symmetric. However, after adding additional "software redundancy" and making a projection to the smallest subspace containing all symmetric states[3] we can expect to get rid of erroneous computations.

Definition 7.3.1 *If H is an n-dimensional Hilbert space, then $\mathcal{SYM}_H^{(r)}$ is the smallest subspace of $H^{(r)} = \bigotimes_{i=1}^r H$ containing all states of the form $\bigotimes_{i=1}^r |\psi\rangle$, where $|\psi\rangle \in H$.*

The above definition does not bring a sufficient insight into the space $\mathcal{SYM}_H^{(r)}$—one does not see immediately that states $|\psi_1\rangle|\psi_2\rangle + |\psi_2\rangle|\psi_1\rangle$ are in $\mathcal{SYM}_H^{(2)}$ if $|\psi_1\rangle, |\psi_2\rangle \in H$. The following theorem offers an alternative and equivalent definition.

Theorem 7.3.2 *$\mathcal{SYM}_H^{(r)}$ is the subspace of all states in $H^{(r)}$ which are symmetric in the sense that they are unchanged under the interchange of sites for any pair of positions in the tensor product of their basic states.*

It is important that the subspace $\mathcal{SYM}_H^{(r)}$ is small. Its dimension is $\mathcal{O}(r^{n-1})$.

[3]Projection always has to be done to a subspace, and states of the form $\bigotimes_{i=1}^r |\psi\rangle$, $|\psi\rangle \in H$ do not form by themselves a subspace of $H^{(r)}$.

7.3.2 Stabilization by projection into the symmetric subspace

The basic idea of the stabilization method discussed below is that error-free components of states from $H^{(r)}$ are always in $\mathcal{SYM}_H^{(r)}$ and therefore by a projection into this subspace the error-free components remain to be included and the components with errors will be removed. This way not all errors will be removed because $\mathcal{SYM}_H^{(r)}$ contains also states not of the form $|\psi\rangle|\psi\rangle\ldots|\psi\rangle$. Nevertheless the error probability will be decreased by the factor of r.

The projection algorithm presented below is efficient: the number of operations performed is polynomial with respect to r and $\lg n$.

In the algorithm we assume that the set $\{\sigma_i\}_{i=1}^{r!}$ contains all permutations of r elements. Let $|\phi_1\rangle,\ldots,|\phi_r\rangle \in H^{(r)}$.

Algorithm 7.3.3 (Projection into $\mathcal{SYM}_H^{(r)}$)

1. *Apply to the first $\lceil \lg r! \rceil$ qubits of the register $|0^{(\lceil \lg r! \rceil)}, \phi_1,\ldots,\phi_r\rangle$. the transformation*

$$U : |0^{(\lceil \lg r! \rceil)}\rangle = \frac{1}{\sqrt{r!}}\sum_{i=0}^{r!-1}|i\rangle$$

2. *Apply, to the state $|\psi\rangle = \frac{1}{\sqrt{r!}}\sum_{i=0}^{r!-1}|i,\phi_1,\ldots,\phi_r\rangle$, the unitary transformation which transforms $|i\rangle|\phi_1\rangle\ldots|\phi_r\rangle$ into $|i,\phi_{\sigma_i(1)},\ldots,\phi_{\sigma_i(r)}\rangle$' to get*

$$|\psi_1\rangle = \frac{1}{\sqrt{r!}}\sum_{i=0}^{r!-1}|i,\psi_{\sigma_i(1)},\ldots,\phi_{\sigma_i(r)}\rangle.$$

3. *Apply U^{-1} to the first $\lceil \lg r! \rceil$ qubits of $|\psi_1\rangle$. The resulting state is then*

$$|\psi_2\rangle = \frac{1}{\sqrt{r!}}\sum_{i=0}^{r!-1}|i,\xi_i\rangle,$$

   where $|\xi_i\rangle \in H^{(r)}$. Since U transforms $|0^{(\lceil \lg r! \rceil)}\rangle$ into an equal-amplitude superposition of all $|i\rangle$, U^{-1} transforms each $|i\rangle$ back to $|0^{(\lceil \lg r! \rceil)}\rangle$ with equal-amplitude. The coefficient at $|0^{(\lceil \lg r! \rceil)}\rangle$ will therefore be an equal amplitude superposition of all permutations of $|\phi_1\rangle,\ldots,|\phi_r\rangle$, i.e., the required symmetrized state.

4. *Measure the first $\lceil \lg r! \rceil$ qubits with respect to the standard basis. If the result is $0^{(\lceil \log r! \rceil)}$, then the state $|\psi_0\rangle = |\phi_1\rangle\ldots|\phi_r\rangle$ has been successfully projected into $\mathcal{SYM}_n^{(r)}$; otherwise the symmetrization failed.*

Due to the linearity of all processes the algorithm can be applied to the general state of $H^{(r)}$ and not only to a basis state as illustrated above.

Exercise 7.3.4 *Show that for the algorithm 7.3.3 the total number of operations is $\mathcal{O}(r^2 \lg n + (r \lg r)^2)$ and therefore the algorithm can be considered as efficient — in spite of the fact that we need to consider a creation of superpositions with $r!$ members.*

Exercise 7.3.5 *Design a symmetrization network of the size $\mathcal{O}(r^2)$ for r redundant computations.*

It has been shown by Barenco et al. (1997), by a detailed analysis, that both for unitary errors and for errors due to decoherence, the stabilization by symmetrization of r-redundant computations reduces the error in each step of stabilization by the factor $\frac{1}{r}$.

7.4 Quantum Error-Correcting Codes

No really significant progress in classical computing and communication would have been possible without error-correcting codes. They help to deal successfully and efficiently with the noise and disruptions one encounters in classical computing and communication.

Practically till 1995 it has been believed that error-correcting codes cannot be used in quantum computing, and this had been one of the main arguments of those with serious doubts concerning the overall feasibility of quantum computing. The reasoning against quantum error-correcting codes went as follows.

All states of the form

$$\alpha|0\rangle + \beta|1\rangle \quad \text{with} \quad |\alpha|^2 + |\beta|^2 = 1$$

are legal states. There are uncountably many such states. Since quantum evolution is basically an analogue and therefore a continuous process, and since all the above states are legal, there is no way to distinguish a state $|\phi\rangle$ from the one obtained from $|\phi\rangle$ by adding some noise. Moreover, the number of possible errors seemed to be infinite and it seemed that an error destroys a state in such a way that the original state cannot be recovered. In addition, each attempt to make a restriction to a discrete set of states seemed to bring an essential restriction to quantum computing.

However, it was shown first by Shor (1995) and Steane (1996), and soon by many others, that good quantum error-correcting codes exist and they can protect qubits against general types of error (which may be caused by imperfections or interactions with the environment). Quantum error-correcting codes have to be based on different principles than classical error-correcting codes, but they do exist. Since then the progress in the development of quantum error-correcting codes has been remarkable—this has been one of the most successfully developing areas of quantum computing and it is to a large extent due to these results, and results discussed in the next section, on fault-tolerant quantum computing, that the vision of real quantum computing is much closer.

The very basic idea of quantum computation with error-correcting codes goes as follows. The evolution of the quantum computer is restricted to a subspace of the Hilbert space carefully chosen in such a way that if quantum states are encoded using states of the chosen subspace, then all departures from this subspace, due to errors, lead to mutually orthogonal subspaces. After a quantum state is entangled with the environment and an "error" occurs, one can determine, by a measurement, but without destroying the erroneous state, into which of the error subspaces the erroneous state has fell, and the error can be undone using a unitary transformation.

However, it is far from trivial how to implement such an idea and how to utilize redundancy for that. (It is well known that redundancy is not very useful in analogue computing.) The ingenious idea of Shor and Steane was to use quantum entanglement for the design of quantum error-correcting codes (QECC). The discovery of QECC caused much excitement because it converted large-scale quantum computation from an impossibility to a possibility.

There are three new types of problem concerning quantum error-correcting codes, compared to the classical situation:

1. The variety of possible errors seems to be much broader.
2. The assumption that encoding and decoding are error-free is much less realistic.
3. Since decoherence decreases the probability of success of quantum computation exponentially, quantum error-correction codes need to have some "exponential efficiency". They have to be able "to fight exponential decoherence in polynomial time".

A desirable error correction process can be seen as having the following form: the sender, Alice, encodes a to-be-sent quantum state into a new quantum state which is then sent through a noisy channel on which an error/noise operator acts and changes the transmited state. Encodding has to be such that even if the error operator changes the state being trasmitted, it cannot entangle it with the environment and, as a consequence, the receiver Bob, who can act on the state he receives, but not on the environment, is able first to detect which error operator was applied and then he can undo its effect and to receive the original state.

As discussed in more detail in Chapter 8, in order Bob is able to recover the state Alice has sent, no information about her state should leak into the environment. Quantum codes have to be therefore such that they hide information from the environment. The idea is to use encodings of such types that encoded quantum information of k qubits is spread out over n qubits in a non-local way through an entangled state in such a way that environment which can access only a small number of qubits can gain no information about the overall state being transmited and this way transmited quantum information is protected.

Quantum error-correcting theory is a crucial part of quantum information theory. A variety of quantum error-correction methods have their analogues in classical error-correcting codes and rely heavily on their properties. That is why we start with a short summary of the very basic concepts and methods of the classical error-correcting codes codes. For more see Hill (1986), Hoffman et al. (1991), van Lint (1995) and MacWilliams and Sloane (1977).

7.4.1 Classical error-detecting and -correcting codes

The aim of the error-detecting and -correcting codes is to help to transmit safely messages through noisy channels using words of a q-ary alphabet $\Sigma_q = \{0, 1, \ldots, q-1\}$.

The **Hamming distance** of two words u and v, notation $hd(u, v)$, is the number of symbols in which u and v differ. The **(Hamming) weight**, $hw(w)$, of a codeword w is the number of symbols in w different from 0.

A **code** C is a subset of Σ_q^n for some n; its elements are called **codewords**. For error detection and correction the minimal distance $d(C)$ of a code C is of importance.

$$d(C) = \min\{hd(u, v) \mid u, v \in C, u \neq v\}.$$

The distance $d(C_1, C_2)$ of two codes C_1 and C_2 is defined by

$$d(C_1, C_2) = \min\{hd(w_1, w_2) \mid w_1 \in C_1, w_2 \in C_2\}.$$

This allows us to formulate one of the most basic results of the error-detecting and -correcting codes.

Theorem 7.4.1 *(1) A code C can detect up to s errors in any codeword if and only if $d(C) \geq s+1$; (ii) A code C can correct up to t errors if and only if $d(C) \geq 2t+1$.*

Definition 7.4.2 *An (n, M, d)-**code** is a code of M words of length n and minimal distance d. $A_q(n,d)$ denotes the largest M such that there exists a q-nary (n, M, d)-code.*

Exercise 7.4.3 *Show that $A_q(n,1) = q^n$, $A_q(n,n) = q$.*

The equivalence of codes is defined as follows:

Definition 7.4.4 *Two q-nary codes are called* **equivalent** *if one can be obtained from the other by a combination of the following operations:*

- *permutation of the positions of the codewords;*
- *permutation of the symbols appearing in a fixed position in the codewords.*

One of the aims of the coding theory is to find perfect codes. In order to define them let, for $u \in \Sigma_q^n$ and $r \geq 0$, $S(u,r) = \{v \in \Sigma_q^n \mid hd(u,v) \leq r\}$ be the sphere of the radius r.

Exercise 7.4.5 *Show that a sphere of radius $0 \leq r \leq n$ in Σ_q^n contains exactly*

$$\binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \ldots + \binom{n}{r}(q-1)^r$$

words.

Exercise 7.4.6 *Show that a q-nary $(n, M, 2t+1)$-code satisfies the following* **space packing** *or* **Hamming inequality (bound)***;*

$$M\{\binom{n}{0} + \binom{n}{1}(q-1) + \ldots + \binom{n}{t}(q-1)^t\} \leq q^n.$$

Definition 7.4.7 *An $(n, M, 2t+1)$-code C is called* **perfect**[4] *if the equality holds in 7.4.6.*

Exercise 7.4.8 *Explain why is perfect a perfect code.*

Extension of a code is an important technique to design new codes. A code C is extended by adding one bit to each codeword. For example, this bit, the parity check bit, is 0 if the weight of the code is even and 1 otherwise.

Several classes of codes are of special basic importance. The simplest of them is the $(n, 2, \lceil (n-1)/2 \rceil)$ **repetition code** $\{0^{(n)}, 1^{(n)}\}$. **Linear codes**, which we deal with next, are the most often used family of codes.

[4]it is already pretty well known which codes are perfect.

Linear codes

In order to formulate elegantly basic concepts and results of linear codes it is useful to consider words $u_1u_2\dots u_n$ of Σ_q^n as vectors $(u_1,u_2,\dots,u_n)$ of length n with elements from $\mathbf{Z}_q$. The set of all such vectors is denoted $V(n,q)$. Component-wise addition of two vectors and scalar/vector multiplications in $V(n,q)$ are done in $\mathbf{Z}_q$.

Definition 7.4.9 *A code C over $V(n,q)$ is called* **linear** *if C is a subspace of $V(n,q)$,qprime.*

Exercise 7.4.10 *Show that a subset $C \subseteq V(n,q)$ is a linear code if and only if: (1) $u+v \in C$ for all u,v in C; (2) $au \in C$ for all $u \in C$, $a \in \mathbf{Z}_q$.*

If the dimension $dim(C)$ of a linear code C in $V(n,q)$, as that of the subspace C, is k then C is said to be an $[n,k]$-code. In addition if C is of distance d, then it is said to be $[n,k,d]$**-code**. In other words, an $[n,k,d]$-code is a code by which n bits can store k bits of information in such a way that correction of up to $\lfloor (d-1)/2 \rfloor$ errors is always possible.

The **rate** of an $[n,k,d]$-linear code C is $\frac{k}{n}$. This is the ratio of the information content of a codeword to the information content of an arbitrary string of length n.

If C is a linear code, then $C^{\perp} = \{w \mid u \cdot w = 0 \text{ if } u \in C\}$ is called the **dual code** to C. A code C is **self-dual** if $C^{\perp} = C$.

Exercise 7.4.11 *Show that all codewords of a binary self-dual code have an even number of ones.*

Exercise 7.4.12 *Show that if C is an $[n,k]$ code over $\mathbf{Z}_q$, then $C^{\perp}$ is a $[n,n-k]$-code over $\mathbf{Z}_q$.*

Exercise 7.4.13 *Show that two $k \times n$ matrices generate equivalent linear $[n,k]$-codes over $\mathbf{Z}_q$ if one matrix can be obtained from the other by a sequence of operations of the following type: (1) permutation of rows; (2) multiplication of a row by a non-zero scalar; (3) addition of a scalar multiple of one row to another; (4) permutation of columns; (5) multiplication of any column by a non-zero scalar.*

A matrix G whose rows are all vectors of a basis of a linear code C (as a subspace) is said to be a **generator** of C. A generator matrix H of the dual code $C^{\perp}$ is called the **parity-check** matrix of C.

If G is a generator matrix of an $[n,k]$-code C, then $C = \bigcup_{v\in\{0,1\}^k} vG$. The name "parity-check matrix" is derived from the fact that a parity matrix H of a code C can be used to test whether a given word w is in C. Indeed, $w \in C$ if and only if $Hw^T = \mathbf{0}$.

The following theorem, easy to show, provides a simple way to construct a parity-check matrix of a linear code with a given generator matrix and vice versa.

Theorem 7.4.14 *If G is the generator matrix of an $[n,k]$-code C written in the form $G = [I_k|A]$, where I_k is the $k\times k$ unit matrix, then a parity-check matrix for C is $H = [-A^T|I_{n-k}]$.*

It follows from the definition of the parity-check matrix H of a code C that for each $w \in C$

$$wH^T = \mathbf{0^T}, \quad \mathbf{0} = Hw^T$$

and if $w \notin C$, $w = w_1 + w_e$, $w_1 \in C$, then $w_1 H^T = \mathbf{0}^T$, $\quad w_1^T H = 0$. This means that the row space of H is orthogonal to C. In addition, $GH^T = \mathbf{0}^T$, $\ \mathbf{0} = HG^T$.

Exercise 7.4.15 *Denote by $w(u)$ the* **(Hamming) weight** *of a word u. For a code C let $w(C) = \min\{w(u) \,|\, u \in C - \{0\}\}$. Show that $d(C) = w(C)$ for any linear code C.*

Encoding with linear codes. If C is an $[n, k]$-code over $\mathbf{Z}_q$ with a generator matrix G, then C contains q^n codewords and therefore it can be used to communicate up to q^k distinct messages. Let us identify messages with words in $V(n, q)$. Encoding of a message u is done by the matrix multiplication uG.

Syndrome decoding with linear codes is also easy, but several new concepts are needed to formulate an efficient algorithm.

Definition 7.4.16 *If C is an $[n, k]$-code over $\mathbf{Z}_q$ and a is any vector in $V(n, q)$, then the set $a + C = \{a + x \,|\, x \in C\}$ is called the* **coset** *of C. A vector of a coset with the minimum weight is its* **leader** *(which does not have to be unique).*

Exercise 7.4.17 *Suppose C is an $[n, k]$-code over $\mathbf{Z}_q$. Show: (a) every vector of $V(n, q)$ is in some coset of C; (b) any coset contains exactly q^k vectors; (c) two cosets of C are either identical or disjoint.*

Suppose H is a parity-check matrix of an $[n, k]$-code C. For any $y \in V(n, q)$ the row vector $S(y) = yH^T$ is called the **syndrome** of y (with respect to C).

As discussed above, if $w = w_1 + w_e$ with $w_1 \in C$, then $S(w) = S(w_e)$ and therefore the syndrome only depends on the word (vector) w_e. That means that a syndrome specifies an error without revealing anything about the codeword w_1 itself. This is an important property of the syndromes of linear codes that plays the key role in several quantum error correcting codes.

Exercise 7.4.18 *Show that two vectors are in the same coset if and only if they have the same syndrome.*

Decoding is now easy once we have constructed the so-called **standard array** for an $[n, k]$-code C. It is a $q^{n-k} \times q^k$ array of all vectors in $V(n, q)$. The first row contains all codewords of C starting with the word $0^{(n)}$. The first column contains leaders of all cosets. All other entries in the array are sums of elements in the first row of the corresponding column and in the first column of the corresponding row. In addition, one column is added with ith element being the syndromes for the cosets of the ith row.

Algorithm 7.4.19 (Syndrome decoding for linear codes) *Given a word y to decode do the following;*

1. *compute $S(y) = yH^T$;*
2. *Decode y as $y - l_y$, where l_y is the coset leader in the coset with the syndrome $S(y)$.*

In order to make decoding, or error correction, one needs to flip the erroneous bits, i.e., to apply NOT operation to them.

Exercise 7.4.20 *(Singleton bound) Show that if C is an (n,k,d) linear code, then $d \leq n-k+1$.*

The Hamming bound, page 271, and Exercise 7.4.20 provide upper bounds on the size of linear (n,k,d)-codes with the given distance. The **Gilbert–Varshamov bound**

$$\sum_{j=0}^{d-2} \binom{n-1}{j} < 2^{n-k}$$

provides an achievable lower bound—for binary codes.

Hamming codes

The Hamming codes are interesting and important examples of the single error-correcting linear codes with easy encoding and decoding.

Definition 7.4.21 *Let $r \in \mathbf{N}^+$ and H be an $r \times (2^r - 1)$ matrix whose columns are distinct non-zero vectors of $V(r,2)$. The code having H as its parity-check matrix is called a* **binary Hamming code** *and denoted by $Ham(r,2)$.*

Example 7.4.22 *$Ham(3,2)$ is the Hamming code with the parity-check matrices*

$$H = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \quad or \quad H' = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

By Theorem 7.4.14 a generator matrix for $Ham(3,2)$ is

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

The main theoretical results on Hamming codes are summarized in the following theorem:

Theorem 7.4.23 *The Hamming code $Ham(r,2)$ has the following properties: (1) it is a $[2^r-1, 2^r-1-r]$-code; (2) it has minimum distance 3; (3) is a perfect code.*

Hence $Ham(3,2)$-code is an $[7,4,3]$-code. The dual to a Hamming code is called **simplex code** or **maximal-length feedback shift register code**.

Exercise 7.4.24 *Show that dual code to the Hamming $[7,4,3]$-code consists of Hamming code codewords of even weight.*

Dual of the extended Hamming code is also an important code: **first-order Reed–Muller code**. Since each Hamming code is linear, encoding with it is easy as described above. To describe decoding observe first that all coset leaders are exactly all vectors of weight ≤ 1. The syndrome of each such coset leader $(0, \ldots, 1, \ldots, 0)$ with 1 in the jth position is just the transpose of the jth column of H. Therefore, if the columns of H are arranged in order of the increasing binary numbers they represent, as in the example above, we have the following decoding method:

Algorithm 7.4.25 (Decoding algorithm for Hamming code)

Input: *a vector* y.

1. *Calculate the syndrome* $S(y) = yH^T$.

2. *If* $S(y) = 0$ **then** y *is assumed to be the codeword itself*;
else $S(y)$ *gives the binary representation of the error position.*

Example 7.4.26 *If* $y = 1110011$, *then* $S(y) = 001$ *and therefore* y *is decoded as* 0110011.

Cyclic codes

Definition 7.4.27 C *is a* **cyclic code** *if it is a linear code and if* $a_{n-1}a_{n-2}\ldots a_0 \in C$ *implies that* $a_0a_{n-1}\ldots a_1 \in C$.

Cyclic codes have an interesting algebraic structure. To see it let us identify a polynomial $a_0 + a_1x + a_2x^2 + \ldots + a_{n-1}x^{n-1}$ with each codeword $a_{n-1}a_{n-2}\ldots a_0$.

Moreover, denote by R_n^p the set of all polynomials of one variable over $\mathbf{Z}_p$, with p a prime, taken modulo the polynomial $x^n - 1$. For a polynomial $f(x) \in R_n^p$ denote $\langle f(x)\rangle = \{r(x)f(x) \mid r(x) \in R_n^p\}$. It holds:

Theorem 7.4.28 *(1) For any* $f(x) \in R_n^p$, *the set* $\langle f(x)\rangle$ *is a cyclic code (generated by* $f(x)$*). (2) If* C *is a non-zero cyclic code in* R_n^p, *then there is a polynomial* $g(x)$ *such that* $C = \langle g(x)\rangle$ *and* g *is a factor of* $x^n - 1$.

Two special types of cyclic codes are of importance.

Definition 7.4.29 *A cyclic code of codewords of length* n *over* $\mathbf{Z}_q$, $q = p^r$ *is called* **BCH code** *of distance* d *if its generator* $g(x)$ *is the least common multiple of the minimal polynomials of* ω^l, $\omega^{l+1}, \ldots, \omega^{l+d-2}$ *for some* l, *where* ω *is the primitive* n*th root of unity.*[5] *If* $n = q^m - 1$ *for some* m, *then the BCH code is called primitive. A* **Reed–Solomon code** *is a primitive BCH code with* $n = q - 1$. *Reed-Solomon codes are self-dual.*

Classical codes as quantum states

A binary code $C = \{c_1, \ldots, c_m\}$ of codewords of length n specifies a subspace H_C of Hilbert space H_{2^n} spanned by the vectors $\{|c_1\rangle, \ldots, |c_m\rangle\}$ and can be represented by the quantum state

$$|\phi_C\rangle = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} |c_i\rangle.$$

[5] A polynomial p is said to be minimal for a complex number $x \in \mathbf{Z}_q$ if $p(x) = 0$ and p is irreducible over $\mathbf{Z}_q$.

If $|\phi_C\rangle$ is represented in the dual basis, then this representation can be seen as defining a new code—C_{dual}. Let C_{dual} have m' codewords and distance d' and let C have distance d. Clearly $m \leq A(n,d)$, $m' \leq A(n,d')$. According to the inequality 2.5, page 73, we have

$$A(n,d)A(n,d') \geq 2^n.$$

If C is a $[n,k]$-linear code, then C_{dual} has a neat form, which is easy to show:

Theorem 7.4.30 *If C is a linear code, then $C_{dual} = C^{\perp}$.*

According to Exercise 7.4.12, C_{dual} is a $[n, n-k]$-code with 2^{n-k} codewords. Linear codes are therefore codes for which the equality holds in the inequality (2.5).

Classical error-correction techniques cannot be directly applied to quantum information processing for two main reasons: (1) it is not possible, in general, to copy or measure qubits without causing undesirable effects; (2) it is not sufficient to correct 0/1 values of qubits—also amplitudes need to be preserved and this is a completely new feature with which quantum error-correcting techniques have to be able to deal.

7.4.2 Framework for quantum error-correcting codes

In order to be able to deal with the problem of correcting the effects of noise (errors) at the communication through noisy channels we need a formal description of the noise or error process (see Steane, 1998, and Knill and Laflamme, 1997).

Error model

Inaccuracies, noise and decoherence can be described in terms of the most general quantum operators—superoperators—or, equivalently, in terms of the unitary operators on the system and its environment.

There is a large variety of possible quantum errors. [6] However, to consider QECC successfully it is quite sufficient to make several simple, but (quasi-)realistic assumptions concerning the character, frequency and types of errors. [7]

We shall assume that errors, due to decoherence and inaccuracies, on different qubits or on the same qubit in different times are random and statistically uncorrelated. Namely, that they are **locally independent** (errors in different qubits or gates are not correlated) and **sequentially independent** (subsequent errors on the same qubit or in the same gate are not correlated). [8] (In other words, it is assumed that there are no interactions between environments of different qubits and also between environments of the same qubit at different time steps.) No knowledge about their nature will be assumed. [9] (As a consequence, an error operator on n qubits can be written at each time step as a tensor product of errors

[6] The term "error" is used here in a special way. As pointed out by Peres (1996) "A computer is a physical system, subject to the laws of nature. No errors occur in the application of these laws. What we call an error is a mismatch between what the computer does and what we wanted it to do."

[7] Without assumptions on how error occurs it is not possible to prove nontrivial results on error correction.

[8] If additional information is available about the error process, more efficient quantum error-correcting procedures can be developed to deal with errors of such processes. See, for example, Plenio, Vedral and Knight (1996), for the case that the error process is a spontaneous emission.

[9] Sometimes the so-called **no leakage** assumption is made: a physical system which implements a qubit has access only to the two-dimensional Hilbert space defined by the qubit. A photon with two basis states represented by the horizontal and vertical polarizations is an example of a system which, without modifications, does not satisfy this assumption (photons have a tendency to be scattered or absorbed and in this way lost for the computation).

on particular qubits.) If these conditions are satisfied, then it is believed that errors are correctable provided the error rate is below 10^{-5} per qubit and clock cycle (DiVincenzo and Terhal, 1998). As a consequence, the above error model implies that a correlation between errors on different qubits can exists only in the case of qubits interacting through a quantum gate.

Error types

A general interaction between a qubit $\alpha|0\rangle + \beta|1\rangle$ and its environment leads to the evolution of the form:

$$|e\rangle(\alpha|0\rangle + \beta|1\rangle) \rightarrow \alpha(|e_{00}\rangle|0\rangle + |e_{01}\rangle|1\rangle) + \beta(|e_{11}\rangle|1\rangle + |e_{10}\rangle|0\rangle), \tag{7.3}$$

where $|e\rangle$, $|e_{00}\rangle$, $|e_{01}\rangle$, $|e_{10}\rangle$ and $|e_{11}\rangle$ are states of the environment.

The right-hand side of (7.3) can now be written in the form

$$(|e_{0+}\rangle I + |e_{0-}\rangle\sigma_z + |e_{1+}\rangle\sigma_x - |e_{1-}\rangle i\sigma_y)(\alpha|0\rangle + \beta|1\rangle), \tag{7.4}$$

where

$$|e_{0+}\rangle = \frac{1}{2}(|e_{00}\rangle + |e_{10}\rangle) \qquad |e_{0-}\rangle = \frac{1}{2}(|e_{00}\rangle - |e_{10}\rangle), \tag{7.5}$$

$$|e_{1+}\rangle = \frac{1}{2}(|e_{01}\rangle + |e_{11}\rangle) \qquad |e_{1-}\rangle = \frac{1}{2}(|e_{01}\rangle - |e_{11}\rangle), \tag{7.6}$$

and $\sigma_x, \sigma_y, \sigma_z$ are Pauli matrices.

The key ideas behind quantum error-correction codes can be seen looking carefully on the state (7.4): (1) any error can be seen as being composed of four basic errors and therefore if we are able to correct any of these four types of error, we can correct any error; (2) error model resembles more a discrete one than a continuous one; (3) the resulting state of the environment is independent of the state on which an error process acts and depends only on the type of error operator being applied. This also suggest the following error-detection and -recovery process: (1) To compute which type of error has occurred (error "syndrome" is computed); (2) to undo errors.

The impact of Pauli matrices on a qubit $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$ is shown in Figure 7.1.

$\|\phi\rangle$	$\sigma_x\|\phi\rangle$	$\sigma_y\|\phi\rangle$	$\sigma_z\|\phi\rangle$	$i\sigma_x\sigma_y\|\phi\rangle$
$\alpha\|0\rangle + \beta\|1\rangle$	$\alpha\|1\rangle + \beta\|0\rangle$	$i\alpha\|1\rangle - i\beta\|0\rangle$	$\alpha\|0\rangle - \beta\|1\rangle$	$\alpha\|1\rangle - \beta\|0\rangle$

Figure 7.1: Impacts of quantum errors

σ_x therefore stands for the **bit (flip) error** (or "amplitude error"), σ_z for the **sign (flip) error** (or "phase (shift)" error). σ_y and $\sigma_x\sigma_z$ for a **bit-sign (flip) error** (or "bit-phase" error)—a combination of the bit error and the phase error.

Observe that $\sigma_y = i\sigma_x\sigma_z$. That is why sometimes a slightly different error model is used, with three types of errors represented by matrices $X = \sigma_x$, $Z = \sigma_z$ and $Y = \sigma_x\sigma_z$.

In the case of an n-qubit register the general type of error is therefore represented by the matrix

$$M = \bigotimes_{i=1}^{n} M_i,$$

where $M_i \in \{X, Y, Z, I\}$ is an error matrix for the ith qubit.

Example 7.4.31 *Let us explore perhaps the simplest idea for a quantum error-correcting code (Peres, 1985, Aharonov, 1998), namely encoding of the basis states*

$$|0\rangle \rightarrow |000\rangle \qquad |1\rangle \rightarrow |111\rangle,$$

which results in the encoding of the general one-qubit state $\alpha|0\rangle + \beta|1\rangle \rightarrow \alpha|000\rangle + \beta|111\rangle$. *Unfortunately, such an encoding is not good enough because it does not protect the quantum state even against one error. Indeed, let us assume that a noise operator operates on the first qubit and the environment* $|e\rangle$ *in such a simple way that it does not changes the first qubit, but it changes the environment depending on the value of the qubit:*

$$|e\rangle(\alpha|000\rangle + \beta|111\rangle) \rightarrow \alpha|e_0\rangle|000\rangle + \beta|e_1\rangle|111\rangle.$$

The resulting state is entangled with the environment and Bob cannot disentangled it by a local action and cannot recover the original state.

Basic ideas of quantum error correction

Suppose we want to encode uniquely k qubits into $n > k$ qubits. This means we want to map, using a unitary transformation, any state $|\phi\rangle$ over k qubits into a state over n qubits. This can be done if we introduce $n-k$ new qubits, initially in the state $|0^{(n-k)}\rangle$, for example, and then apply a unitary encoding transformation E as follows.

$$E(|\phi\rangle|0^{(n-k)}\rangle) \rightarrow |\phi_E\rangle,$$

where $|\phi_E\rangle$ is said to be the **quantum code** of $|\phi\rangle$ defined by E. The encodings, or of the basis states of k qubits are called **codewords** and they form an orthonormal basis of a 2^k-dimensional subspace of H_{2^n}.

If an error occurs in a codeword $|\phi_E\rangle$, then $|\phi_E\rangle$ is altered by some linear transformation, superoperator, $\mathcal{E}$ and

$$|\phi_E\rangle \xrightarrow{\mathcal{E}} |\mathcal{E}\phi_E\rangle.$$

($\mathcal{E}$ is not required to be unitary; there is a need to correct also non-unitary errors.)

An error-correction process (ECP) can now be modeled by unitary transformations that first entangle the erroneous state $|\mathcal{E}\phi_E\rangle$ with an **ancilla** (an auxiliary state of auxiliary qubits), and then transform the resulting entangled state into a tensor product of the original state $|\phi_E\rangle$ and a new state $|A_{\mathcal{E}}\rangle$ of the ancilla:

$$|\mathcal{E}\phi_E\rangle|A\rangle \xrightarrow{ECP} |\phi_E\rangle|A_{\mathcal{E}}\rangle.$$

Since the state $|\phi_E\rangle|A_{\mathcal{E}}\rangle$ is not entangled we can measure $|A_{\mathcal{E}}\rangle$ without disturbing $|\phi_E\rangle$ and this way we can determine a transformation which has to be applied to $\mathcal{E}|\phi_E\rangle$ to get $|\phi_E\rangle$.

Let us now look into the error-creation and -correction process in more detail for the important case where erroneous states have the form

$$\sum_{s=1}^{l} M_s|\phi_E\rangle \quad \text{or} \quad \sum_{s=1}^{l} |\psi_{env}^s\rangle M_s|\phi_E\rangle, \tag{7.7}$$

where each M_s is a tensor product of n error matrices from the set $\{X, Y, Z, I\}$ (and it is called an an "**error operator**" or an "**error**" and $|\psi_{env}^s\rangle$ are states of the environment. (As discussed more in Section 7.4.5, such error operators generate a group which will be denoted by $\mathcal{G}_n$.)

The basic task is to determine, without disturbing $|\phi_E\rangle$ in an irreversible way, an operation that has to be performed in order to get $|\phi_E\rangle$ out of $\mathcal{E}|\phi_E\rangle$. The basic idea is to compute, as in the case of linear codes, syndromes of errors without disturbing $|\phi_E\rangle$. In order to do that additional qubits of ancilla are introduced in a special initial state, for example in the state $|0^{(n-k)}\rangle$. In order to compute syndromes, a syndrom-extraction operator S is applied with the effect

$$S(M_s|\phi_E\rangle|0^{(n-k)}\rangle) = M_s|\phi_E\rangle|s\rangle$$

for all M_s from a set of "correctable errors" (which depends on the encoding). Here s is just a number indicating with which error operator we are dealing. (States $|s\rangle$ have to be orthogonal!) Where the noisy state (7.7) contains only correctable errors, after the syndrome extraction operator is applied we get as the resulting state

$$\sum_{s=1}^{l} |\psi^s_{env}\rangle(M_s|\phi_E\rangle|s\rangle). \tag{7.8}$$

Since the states $|s\rangle$ are orthogonal we can measure the ancilla qubits in the basis $\{|s\rangle\}$ to get:

$$|\psi^{s_0}_{env}\rangle(M_{s_0}|\phi_e\rangle|s_0\rangle)$$

for a single, randomly chosen, s_0. This is excellent: instead of a complicated erroneous state we have now only one error operator M_{s_0} and by applying $M^{-1}_{s_0}$ we get as the result the state $|\psi^{s_0}_{env}\rangle|\phi_E\rangle|s_0\rangle$. Therefore, the state $|\phi_E\rangle$ has been reconstructed—it is no longer entangled.

As shown above, the quantum error-correcting processes processes are, surprising, discrete processes, not continuous ones. The key discretization step is projection measurement.

Remark 7.4.32 Actually, it is not necessary to measure ancilla qubits to get the syndrome. Indeed, after the syndrome extraction one can apply a unitary operator C such that $C(|x\rangle|s\rangle) = M_s^{-1}|x\rangle|s\rangle$ to the sum in (7.8). The final state then would be $|\phi_E\rangle\sum_{s=1}^{l}|\psi^s_{env}\rangle|s\rangle$—entanglement between the state and the environment is transferred into the entanglement between the environment and the ancilla.

.

A quantum t error-correcting code code is a (unitary) mapping of k qubits into a subspace of a quantum space of $n > k$ qubits such that errors in any of the t qubits can be corrected; i.e. the original quantum state can be perfectly recovered from the remaining $n - t$ qubits.

The simplest case to consider is $k = 1$, even it is quite clear that "more efficient" QECC are expected to exist for $k > 1$.

Sufficient and necessary conditions for quantum error-correcting codes

Let $\mathcal{S}_\mathcal{E}$ be a set of errors and we are interested to find necessary and sufficient conditions for an error-correcting code to be able to correct all errors from $\mathcal{S}_\mathcal{E}$.

In order to be able to correct any two errors E_a and E_b from $\mathcal{S}_\mathcal{E}$ we must be able to distinguish the case E_a is acting on a basis codeword $|\psi_i\rangle$ from the case E_b is acting on another basic codeword $|\psi_j\rangle$, $i \neq j$. This can be done for sure if and only if the states $E_a|\psi_i\rangle$ and $E_b|\psi_j\rangle$ are orthogonal. Thus, it has to hold

$$\langle\psi_i|E_a^*E_b|\psi_j\rangle = 0 \tag{7.9}$$

if $i \neq j$ and E_a, E_b are any errors from $\mathcal{S}_{\mathcal{E}}$. In other words, errors on different basis codewords have to result in orthogonal states.

How about different errors on the same basis codeword? We could require the same condition to hold. Namely, that (7.9) holds also for $i = j$ and all E_a, E_b from $\mathcal{S}_{\mathcal{E}}$. However, the condition (7.9) is too strong.

What is needed for an error-correcting code is that when we make a measurement to find out about an error, we must learn nothing about the actual state of the coding space on which the error was made. Would we learn something we could damage the superposition of the basis states and we would not be able to correct an arbitrary codeword. How we learn information about a codeword? By measuring $\langle\psi_i|E_a^* E_b|\psi_i\rangle$. This value has therefore to be the same for all basis codewords. In other word, for any correctable errors (i.e. from $\mathcal{S}_{\mathcal{E}}$) E_a and E_b and any $i \neq j$ it has to hold:

$$\langle\psi_i|E_a^* E_b|\psi_i\rangle = \langle\psi_j|E_a^* E_b|\psi_j\rangle. \tag{7.10}$$

Conditions (7.9) and (7.10) are therefore necessary for a code to be able to correct a given set $\mathcal{S}_{\mathcal{E}}$ of errors. Actually, they are also sufficient conditions as shown by Knill and Laflamme (1997) and Bennett et al. (1996a). In this case the proof is more subtle and we sketch here only its basic ideas.

Conditions (7.9) and (7.10) can be combined into a single condition

$$\langle\psi_i|E_a^* E_b|\psi_j\rangle = c_{a,b}\delta_{ij}, \tag{7.11}$$

where c_{ab} is a constant independent on i and j. Let M be a matrix such that $M[a,b] = c_{ab}$. The matrix is Hermitian and therefore it can be diagonalized. If this is done and errors $\{E_a\}$ are properly transformed to a get a new basis $\{F_a\}$ for the space of possible errors, we get that

$$\langle\psi_i|F_a^* F_b|\psi_j\rangle = \delta_{ab}\delta_{ij} \tag{7.12}$$

or

$$\langle\psi_i|F_a^* F_b|\psi_j\rangle = 0, \tag{7.13}$$

depending on a and b. (The new basis does not have to contain only errors that are tensor products of errors from the set $\{X, Y, Z, I\}$). Errors of the type (7.13) annihilate any codeword and therefore the probability they occur is 0. The other errors always produce orthogonal states and therefore we can detect them by a measurement and then they can be corrected. Hence the code satisfies the condition (7.11) if and only if it can correct all correctable errors from $\mathcal{S}_{\mathcal{E}}$.

If for all errors E_a and E_b and any basis states $|\psi_i\rangle, |\psi_j\rangle$,

$$\langle\psi_i|E_a^* E_b|\psi_j\rangle = 0 \tag{7.14}$$

then the code is called **orthogonal** or **non-degenerate**, otherwise it is called **degenerate**.

Remark 7.4.33 Observe that whether a code is degenerate or not depends on the set of errors that is expected to get corrected. For example, a degenerate two error-correcting code can be nondegenerate when considering it as a one error-correcting code.

Exercise 7.4.34 *How should condition (7.11) be changed would we want from a code only the capability to detect an error or a set of errors?*

Bounds on quantum error-correcting codes

As for linear codes, one can derive a quantum version of the Hamming bound on parameters n, k, t for quantum non-degenerate codes mapping k qubits into n and correcting up to t errors (see Steane, 1996, and Ekert and Macchiavello, 1996).

There are 2^k basis states of k qubits. Since there are 3 possible errors (X or Y or Z) the number of possibilities for having i errors on the codeword of n qubits is $3^i\binom{n}{i}$ and for all basis states and for i from 0 to t there are $2^k\sum_{i=0}^{t}3^i\binom{n}{i}$ possible error states. If the code is non-degenerate all error states obtained from the original codeword have to be orthogonal. Since the overall dimension of the space of n qubits is 2^n we have the basic bound:

$$2^k \sum_{i=0}^{t} 3^i \binom{n}{i} \leq 2^n. \tag{7.15}$$

and for large n and t

$$\frac{k}{n} \leq 1 - \frac{t}{n}\lg_2 3 - H(\frac{t}{n}),$$

where H is Shannon entropy, and therefore $\frac{k}{n}$ is close to 1 if $\frac{t}{n} \approx 0.1892$.

For the case $k = 1$ and $t = 1$ we have, as a special case, the bound

$$2(3n+1) \leq 2^n \tag{7.16}$$

and 5 is the minimal n satisfying (7.16). Three questions now immediately come up:

1. Is the bound 7.15 satisfied also by non-degenerate (non-orthogonal) codes?
2. Are there infinitely many codes that can reach the bound (i.e., codes for which equality holds in (7.15))?
3. Can we get bound (7.15) into a more transparent form?

The first problem is still open. Concerning the second problem, Gottesman (1996) has constructed infinitely many codes reaching the bound. Concerning the third problem, attention has focused on the task to determine the best possible rate $\frac{k}{n}$.

Calderbank and Shor (1995) have shown quantum error-correcting codes with asymptotic rate $\frac{k}{n} = 1 - H(\frac{2t}{n})$, and have derived, on the basis of the results of Bennett et al. (1996a), concerning the bounds on the capacity of quantum channels, the upper bound

$$\min\left[1 - H_2\left(\frac{2t}{3n}\right), H_2\left(\frac{1}{2} + \sqrt{(1-\frac{t}{n})\frac{t}{n}}\right)\right],$$

for $\frac{t}{n} < \frac{1}{2}$ and 0 for $\frac{t}{n} \geq \frac{1}{2}$.

Bit versus sign errors

There is a simple relation between bit errors, represented by the matrix $X = \sigma_x$ and the phase error, represented by the matrix $Z = \sigma_z$. Namely,

$$Z = HXH \qquad \text{and} \qquad X = HZH \tag{7.17}$$

where H is the Hadamard matrix, an application of which transforms the states expressed in the standard basis to the dual basis and vice versa. In other words a sign error in the standard basis is the bit error in the dual basis and vice versa.

Together with the fact that the classical error-correcting codes are used to correct bit errors, the identities (7.17) suggest a simple approach to quantum error correction for the case that only bit and sign errors are expected.

A classical error-correcting code C_1 is used to correct bit errors. In order to correct sign errors the erroneous state is first transformed to the dual base, sign errors are then corrected, if by the basis transformation we get codewords with support from some error-correcting code C_2. The resulting state is then transformed back to the standard basis. This will be described later.

Notation

$\{n, k, d_1, d_2\}$ is used to denote quantum codes C that with n qubits encode k qubits and $\lceil (d_1 - 1)/2 \rceil$ ($\lceil (d_2 - 1)/2 \rceil$) bit (sign) errors can be corrected by the code. Notation $\{n, k, d\}$, or $[[n, k, d]]$, is used for $\{n, k, d, d\}$ codes. Finally, notation $\{n, k, d\}^+$ denotes codes codewords of which are superpositions all basis states of which have as amplitude 1.

Remark 7.4.35 Interestingly enough, quantum error-correcting codes are able to fight decoherence caused by entanglement of quantum systems with the environment, using again entanglement as the main tool. As pointed out by Preskill (1998), "we can fight entanglement with entanglement".

7.4.3 Case studies

Design and use of quantum error-correcting codes brings a variety of new problems and methods. This will now be illustrated using some of the first good quantum error-correcting codes—see table in Figure 7.2.

Code	$\|0_E\rangle$	$\|1_E\rangle$
Shor's 9 qb code	$(X)(X)(X)$ $X = \|000\rangle + \|111\rangle$	$(Y)(Y)(Y)$ $Y = \|000\rangle - \|111\rangle$
Steane's 7 qb code	$\|0000000\rangle + \|1010101\rangle + \|0110011\rangle$ $+\|1100110\rangle + \|0001111\rangle + \|1011010\rangle$ $\|0111100\rangle + \|1101001\rangle$	$\|1111111\rangle + \|0101010\rangle + \|1001100\rangle$ $+\|0011001\rangle + \|1110000\rangle + \|0100101\rangle$ $+\|1000011\rangle + \|0010110\rangle$
LMPZ's 5 qb code	$+\|00000\rangle + \|11100\rangle - \|10011\rangle - \|01111\rangle$ $+\|11010\rangle + \|00110\rangle + \|01001\rangle + \|10101\rangle$	$-\|00011\rangle + \|11111\rangle - \|10000\rangle + \|01100\rangle$ $+\|11001\rangle - \|00101\rangle - \|01010\rangle + \|10110\rangle$
Barenco's 3 qb code	$\|000\rangle + \|011\rangle + \|101\rangle + \|110\rangle$	$\|111\rangle + \|100\rangle + \|010\rangle + \|001\rangle$

Figure 7.2: Examples of 1-qubit quantum error-correcting codes; all superpositions are equally weighted, but amplitudes are omitted in the table

Quantum error-correcting code design

The first problem is how to design a good QECC. Several general methods of designing quantum error-correcting codes are presented in the Sections 7.4.4 and 7.4.5. Here we discuss only how codes presented in Figure 7.2 were (or it could be) discovered. Shor's 9-qubit code was historically first (see Shor, 1995), Steane's 7-qubit code next (Steane, 1996) and LMPZ's code (see Laflamme, Miquel, Paz and Zurek (1996)[10] is the best possible code, from the point of view of the minimal number of qubits and also codewords of the support, see inequality (7.15), which can correct all three types of errors when one qubit is encoded. A simple 3-qubit code shown in Figure 7.2 is good for those case where only sign errors occur.

Exercise 7.4.36 *(Shor, 1995) (a) Show that if p is the probability of one qubit error, then the probability that at least 2 qubits out of 9 are erroneous is approximately $36p^2$; (b) Show that if k qubits are encoded using Shor's 9-qubit code, then the probability that $9k$-qubits can be decoded is $(1-36p^2)^k$; (c) determine probabilities as in the cases (a) and (b) for LMPZ's code.*

Shor's code was obtained from the repetition code $\{|000\rangle, |111\rangle\}$ by replacing $|0\rangle$ with $\frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$ and $|1\rangle$ with $\frac{1}{\sqrt{2}}(|000\rangle - |111\rangle)$, two states of the Bell's basis in H_8:

$$\frac{1}{\sqrt{2}}(|000\rangle \pm |111\rangle), \frac{1}{\sqrt{2}}(|001\rangle \pm |100\rangle), \frac{1}{\sqrt{2}}(|010\rangle \pm |101\rangle), \frac{1}{\sqrt{2}}(|010\rangle \pm |101\rangle)$$

The starting classical code for Steane's code was the simplex $(7, 8, 4)$-code

$$C = \{0000000, 1010101, 0110011, 1100110, 0001111, 1011010, 0111100, 1101001\}.$$

Using the standard-to-dual basis transformation we get $C_{dual} = C^{\perp} = Ham(3,2)$—page 274. The resulting Steane's code is now obtained as

$$|0_E\rangle = \sum_{\substack{c \in Ham(3,2) \\ C \text{ even}}} |c\rangle, \qquad |1_E\rangle = \sum_{\substack{c \in Ham(3,2) \\ C \text{ odd}}} |c\rangle,$$

The point is that the code C has distance 4 and $C^{\perp}$ has still large enough distance, 3, to correct one bit error. Observe also that in both codes, $|0_E\rangle$ and $|1_E\rangle$, the last three bits of any codeword uniquely determine the first four. (They are actually the parity bits.)

LMPZ's code was obtained experimentally, by analysis of the orthogonality conditions, as discussed in the next subsection, and by a computer search (Laflamme et al. 1996).

Finally, Barenco's 3-qubit code can be obtained from the repetition code $\{000, 111\}$, using the Hadamard rotation on all three qubits providing:

$$\begin{aligned}
|0\rangle \to |000\rangle &\to \frac{1}{\sqrt{8}}(|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + |110\rangle + |111\rangle) \\
&= \frac{1}{\sqrt{2}}(|0_E\rangle + |1_E\rangle), \\
|1\rangle \to |111\rangle &\to \frac{1}{\sqrt{8}}(|000\rangle - |001\rangle - |010\rangle + |011\rangle - |100\rangle + |101\rangle + |110\rangle - |111\rangle) \\
&= \frac{1}{\sqrt{2}}(|0_E\rangle - |1_E\rangle).
\end{aligned}$$

[10] LMPZ code is equivalent up to a change of basis of individual qubits to the code independently discovered by Bennett et al. (1996a).

Exercise 7.4.37 *Determine which bit, sign and bit–sign errors map the following codes into mutually orthogonal states: (a) Shor's* 9*-qubit code; (b) Steane's* 7*-qubit code; (c) LMPZ's* 5*-qubit code; (d) Barenco's* 3*-qubits code.*

Encoding circuits—encoders

To use a quantum code with mappings $|0\rangle \to |0_E\rangle$, $|1\rangle \to |1_E\rangle$, a quantum circuit is needed to transform an arbitrary quantum state $\alpha|0\rangle + \beta|1\rangle$ into the state $\alpha|0_E\rangle + \beta|1_E\rangle$. Encoding circuits for Steane's and LMPZ's codes are shown in Figures 7.3a,b.

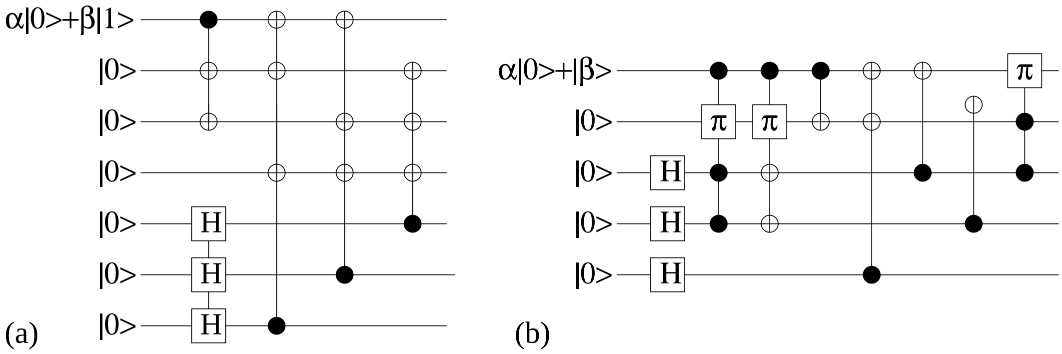

Figure 7.3: Encoding circuits for the Steane's and LMPZ's codes; π-gate realizes π-rotation

The circuit to produce Steane's code is simple (Preskill, 1998) and makes use of the fact that codewords of the support of $|1_E\rangle$ can be obtained from the codewords of the support for $|0_E\rangle$ by flipping the bits. The first two XORs produce the state $\alpha|0000000\rangle+\beta|1110000\rangle$. The three Hadamard gates produce equally weighted superposition of all eight possible values for the last three bits. These three bits uniquely determine the first four bits and this is taken care by the remaining multiple XOR gates.

Encoding circuit for LMPZ's code is more tricky (Laflamme et al. 1996). The first three bits of each codeword uniquely determine the last two in each of the codewords. This is then easy to implement using multiple XOR gates. The problem is with the signs, and they are taken care by π-gates that represent rotation by π (i.e. multiplication by $e^{i\pi}$).

Exercise 7.4.38 *Design encoding circuits for: (a) Shor's* 9 *qubit code; (b) Barenco's* 3*-qubit code.*

Error syndrome computation

Efficient syndrome computation is the key problem in using quantum error-correcting codes.

In the case of Shor's code, Shor (1995), it is instructive to see the impact of the decoherence process

$$|e_0\rangle|0\rangle \to |a_0\rangle|0\rangle + |a_1\rangle|1\rangle, \qquad |e_1\rangle|1\rangle \to |b_0\rangle|0\rangle + |b_1\rangle|1\rangle,$$

where $|a_0\rangle$. $|a_1\rangle$, $|b_0\rangle$, $b_1\rangle$ are states of the environment, on the first qubit of the states $\frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$ and $\frac{1}{\sqrt{2}}(|000\rangle - |111\rangle)$. The resulting states are shown in Figure 7.4a,b.

$$\begin{array}{r} \frac{1}{\sqrt{8}}(|a_0\rangle + |b_1\rangle)(|000\rangle + |111\rangle) \\ +\frac{1}{\sqrt{8}}(|a_0\rangle - |b_1\rangle)(|000\rangle - |111\rangle) \\ +\frac{1}{\sqrt{8}}(|a_1\rangle + |b_0\rangle)(|100\rangle + |011\rangle) \\ +\frac{1}{\sqrt{8}}(|a_1\rangle - |b_0\rangle)(|100\rangle - |011\rangle) \end{array} \qquad \begin{array}{r} \frac{1}{\sqrt{8}}(|a_0\rangle + |b_1\rangle)(|000\rangle - |111\rangle) \\ +\frac{1}{\sqrt{8}}(|a_0\rangle - |b_1\rangle)(|000\rangle + |111\rangle) \\ +\frac{1}{\sqrt{8}}(|a_1\rangle + |b_0\rangle)(|100\rangle - |011\rangle) \\ +\frac{1}{\sqrt{8}}(|a_1\rangle - |b_0\rangle)(|100\rangle + |011\rangle) \end{array}$$

(a) (b)

Figure 7.4: Decoherence impact on Shor's code

Observe an important fact that the states of the environment $(|a_0\rangle + |b_1\rangle, \ldots)$, are the same for both error states arising from encodings of $|0_E\rangle$ and $|1_E\rangle$.

It is now clear that using XOR gates with several ancillary qubits and comparing three triplets one can easily determine whether an error occurred and in which qubit. By a measurement with respect to the Bell basis the type of error can be determined. (By such a measurement the overall state collapses but what remains is good enough to restore the initial state.)

In the case of Steane's code one can make use of the way syndrome is computed for Hamming codes, page 275. Namely, the matrix H, page 274, shows that the error syndrome for the bit error is uniquely determined by three bits: $(b_4 \oplus b_5 \oplus b_6 \oplus b_7, b_2 \oplus b_3 \oplus b_6 \oplus b_7, b_1 \oplus b_3 \oplus b_5 \oplus b_7)$, where $(b_1, \ldots, b_7)$ denote the seven bits of the codewords.

To correct also sign errors the code has to be transformed first from the standard to the dual basis. The key point is now that the resulting classical code of codewords is $Ham(3,2)$ or $(7,4,3)$-code. Now

$$|0'_E\rangle = \frac{1}{\sqrt{2}}(|0_E\rangle + |1_E\rangle), \qquad |1'_E\rangle = \frac{1}{\sqrt{2}}(|0_E\rangle - |1_E\rangle),$$

and the same technique as above can be used to detect sign errors (and consequently also bit–sign errors). Once this is done the code is transformed to the standard basis.

Exercise 7.4.39 *Design error syndrome computation network for Steane's code.*

Syndromes for LMPZ's code can be computed with the same circuit as for code generation; it is only necessary to run this circuit backward. A relation between syndromes and errors is shown in Figure 7.5a (Laflamme et al. 1996).

Finally, in the case of the Barenco's 3-qubit code let us denote by $|0^j_E\rangle$ and $|1^j_E\rangle$, $1 \leq j \leq 3$ states obtained from $|0_E\rangle$ and $|1_E\rangle$ by a sign error on the jth qubit. Error symbols can be computed using two projection operators $\mathcal{P}_1$ on the subspace $\{|0_E\rangle, |1_E\rangle, |0^1_E\rangle, |1^1_E\rangle\}$ and $\mathcal{P}_2$ on the subspace $\{|0_E\rangle, |1_E\rangle, |0^2_E\rangle, |1^2_E\rangle\}$. The syndrome table is shown in Figure 7.5b.

Correcting circuits

Once error symbols are computed, the error correction is done by applying the appropriate unitary transformation on the erroneous state. Let us discuss it in some detail only for Shor's 9-qubit code.

error type	syndrome s_1, s_2, s_3, s_4	resulting state
no	0000	$\alpha\|0\rangle + \beta\|1\rangle$
BS3	1011	$-\alpha\|1\rangle + \beta\|0\rangle$
BS5	1111	$-\alpha\|0\rangle + \beta\|1\rangle$
B2	1000	$\alpha\|0\rangle - \beta 11\rangle$
S3	0101	
S5	0011	
BS2	1010	
B5	1100	$-\alpha\|0\rangle - \beta\|1\rangle$
S1	0001	
S2	0010	
S4	0100	
B1	0110	$-\alpha\|1\rangle + \beta\|0\rangle$
B3	1110	
B4	1101	
BS1	1110	
BS4	1001	

(a)

$\mathcal{P}_1$	$\mathcal{P}_2$	error
true	**true**	no
true	**false**	S1
false	**true**	S2
false	**false**	S3

(b)

Figure 7.5: Syndrome tables for the LMPZ's code. (B (S) stands for bit (sign) error and the number specifies the qubit, and 3-qubit code.

At the measurement during the syndrome computation, with respect to Bell basis, the erroneous state collapses into one of the states

$$\begin{array}{l} (|a_0\rangle + |b_1\rangle)((\alpha+\beta)|000\rangle + (\alpha-\beta)|111\rangle), \\ (|a_0\rangle - |b_1\rangle)((\alpha+\beta)|000\rangle - (\alpha-\beta)|111\rangle), \\ (|a_1\rangle + |b_0\rangle)((\alpha+\beta)|000\rangle + (\alpha-\beta)|011\rangle), \\ (|a_1\rangle - |b_0\rangle)((\alpha+\beta)|000\rangle - (\alpha-\beta)|011\rangle), \end{array} \tag{7.18}$$

In order to make the correction to each of such resulting states a unitary transformation has to be applied that produces the original state

$$\alpha(|000\rangle + |111\rangle) + \beta(|000\rangle - |111\rangle).$$

Exercise 7.4.40 *Determine unitary transformations to get from the states in (7.18) to the original state* $\alpha(|000\rangle + |111\rangle) + \beta(|000\rangle - |111\rangle)$

Technology tailored quantum error correction

As first illustrated by van Enk et al. (1997), specific technologies can allow to use more efficient error-correction methods than those illustrated above. They prepared a quantum optic scheme to transfer perfectly qubits from one ion trap to another. This techniques allows to use two qubits to encode one and uses both quantum teleportation and sending of classical bits to achieve transmissions. (This example indicates that potentials of quantum mechanics for fighting quantum mechanics (decoherence) are still far from being explored.)

Experiments with quantum error correction

The first experimental implementation of QECC, a simple 3 qubit one phase error-correcting code, using NMR technology was reported by Cory et al. (1998).

7.4.4 Basic methods to design quantum error-correcting codes

There are two basic approaches to the design of quantum error-correcting codes: an adaptation of some classical error-correcting codes and a design of completely new, inherently quantum, methods.

Base changes

The first method we are going to discuss is very simple but surprisingly useful and an important component of some more general methods discussed later.

Each quantum code mapping k qubits into n qubits specifies a subspace of dimension 2^k in H_{2^n}. A particular code is usually given by providing a particular basis of the code subspace. By changing the basis a (potentially infinite) number of different code representations can be obtained.

Of special importance is Hadamard rotation, which can switch a bit-correcting code to a sign-correcting code and vice versa.. Sometimes a variety of useful code modifications can be obtained by applying Hadamard rotation to a few carefully chosen qubits.

Example 7.4.41 (DiVincenzo and Shor, 1996)) *Let us start with the codewords*

$$\begin{aligned} |0_E\rangle &= \frac{1}{4}(|00000\rangle \\ &+ |11000\rangle + |01100\rangle + |00110\rangle + |00011\rangle + |10001\rangle \\ &- |10100\rangle - |01010\rangle - |00101\rangle - |10010\rangle - |01001\rangle \\ &- |11110\rangle - |01111\rangle - |10111\rangle - |11011\rangle - |11101\rangle) \end{aligned}$$

and $|1_E\rangle$ obtained from $|0_E\rangle$ by flipping all bits in the basis states. (Observe that basis states of both superpositions are classical cyclic codes.) By changing the basis for all qubits using the Hadamard rotation we get the codewords

$$|0'_E\rangle = \frac{1}{\sqrt{2}}(|0_E\rangle + |1_E\rangle) \qquad |1'_E\rangle = \frac{1}{\sqrt{2}}(|0_E\rangle - |1_E\rangle), \tag{7.19}$$

and if the Hadamard rotation is again applied to the first two qubits of the codewords $|0_E\rangle, |1_E\rangle$, then exactly LMPZ's code from Figure 7.2 is obtained.

Exercise 7.4.42 *Verify, in detail, the above claim how one can get LMPZ's code.*

Changing a quantum code by changing the basis with Hadamard rotation is also an important tool to design error syndrome computing circuits. For example, Figure 7.6 shows such a syndrome computation circuit (due to DiVincenzo and Shor, 1996), for the code with

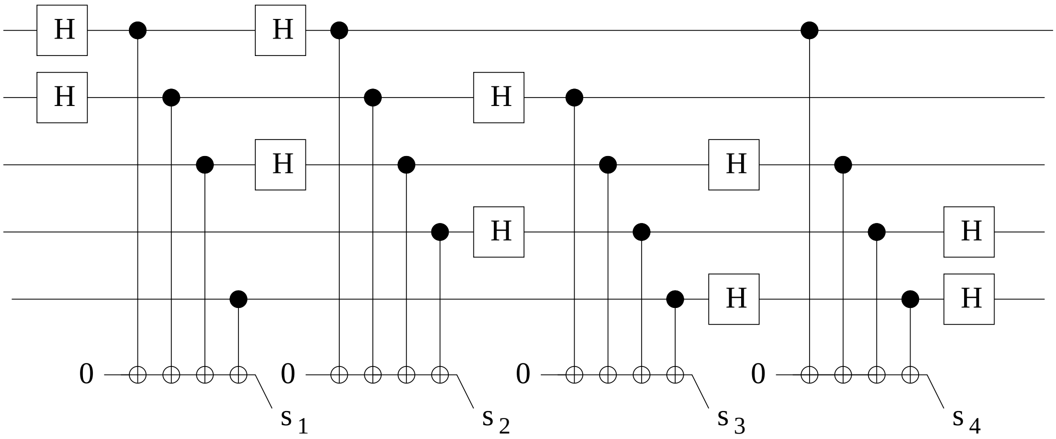

Figure 7.6: A circuit to compute syndromes for the code (7.19)

$|0'_E\rangle$ and $|1'_E\rangle$ given above as codewords. A systematic structure of this network indicates that once the first two qubit rotations have been found useful, one could deduce the rest of the circuit by a careful inspection of the codes obtained each time additional qubits bases were changed (and then transformed back).

Exercise 7.4.43 *Design an error-correcting table showing how syndromes s_1, s_2, s_3, s_4, provided by the network in Figure 7.6, determine errors for the code (7.19).*

CSS-codes—converting classical codes into quantum codes

Another simple and powerful method of designing one quantum error-correcting code out of two, or out of one (dual), classical error-correcting linear codes, was developed by Calderbank and Shor (1995) and Steane (1996a).

Let C_1 be an $[n,k]$ linear binary code with the generator matrix G. Denote by H_{C_1} the subspace of H_{2^n} spanned by the set of vectors $\{|c\rangle \,|\, c \in C_1\}$. In this subspace we construct a new basis as follows. To each $w \in \{0,1\}^n$ we associate a quantum state:

$$|c_w\rangle = \frac{1}{\sqrt{2^k}} \sum_{v\in\{0,1\}^k} (-1)^{vG\cdot w}|vG\rangle \tag{7.20}$$

$$= \frac{1}{\sqrt{2^{n-k}}} \sum_{u\in C^\perp} (-1)^{u\cdot w}|v+w\rangle. \tag{7.21}$$

The last equality, which expresses $|c_w\rangle$ in the dual basis follows from the properties of the superpositions $|c_w\rangle$ summarized in the following exercises.

Exercise 7.4.44 *Show the following properties of the superpositions* $|c_w\rangle$. *(a)* $\sum_{v\in\{0,1\}^k}(-1)^{vG\cdot w} = 0$ *unless* $vG \cdot w = 0$ *for all* $v \in \{0,1\}^k$*; (b) if* $w_1 + w_2 \in C_1^\perp$*, then* $|c_{w_1}\rangle = |c_{w_2}\rangle$*; (c) if* $w_1 + w_2 \notin C_1^\perp$*, then* $c_{w_1} \cdot c_{w_2} = 0$*; (d)* $\{|c_w\rangle \,|\, w \in \{0,1\}^k\}$ *contains exactly* 2^k *different states and all are mutually orthogonal (and therefore form a basis for* H_{C_1}*); (e) cosets of* $C_1^\perp$ *are natural elements to index states* $\{|c_w\rangle \,|\, w \in \{0,1\}^k\}$.

If $C_2 \subseteq C_1$ is another code, then we define a quantum code $Q_{C_1,C_2} = \{|c_w\rangle \,|\, w \in C_2^\perp\}$. Clearly, $dim(Q_{C_1,C_2}) = dim(C_1) - dim(C_2)$. If C is a self-dual (n,k,d) code, then $Q_C = Q_{C,C^\perp} = \{|c_w\rangle \,|\, w \in C^\perp\}$ and $dim(Q_C) = k - (n-k) = 2k - n$.

Exercise 7.4.45 *Design* Q_{C_1,C_2} *code for the case* $C_1 = Ham(3,2), C_2 = C_1^\perp$.

Exercise 7.4.46 *Show that if* C *is a self-dual code then in the dual bases* $|c_w\rangle$ *has the form*

$$|d_w\rangle = \frac{1}{\sqrt{2^{n-k}}} \sum_{c\in C^\perp} |w + c\rangle. \tag{7.22}$$

The following theorem (see Calderbank and Shor, 1996), provides a justification for an introduction of the codes Q_{C_1,C_2}.

Theorem 7.4.47 (Quantum Correction Theorem, 1996)) *If* C_1*,* C_2 *are linear* $[n,k,d]$*-codes and* $C_2^\perp \subset C_1$*, then* $Q_{C_1,C_2^\perp}$ *is a* t *error-correcting quantum code for* $t = \lceil (d-1)/2 \rceil$.

It is easy to see that Q_{C_1,C_2} codes can be used to decode bit errors in one basis and sign errors in the dual basis, and that these two error correction steps do not interfere—Q_{C_1,C_2} can be therefore used also to correct bit–sign errors.

Using this approach $\{13,5,3\}^+$-code, $\{14,6,3\}^+$-code, $\{17,7,3\}^+$-code and $\{20,9,3\}^+$-code were designed by Steane (1997c). He has also found, using a systematic change of the signs of the basis states of codewords, the following codes: $\{8,3,3\}$-code, $\{10,4,3\}$-code and $\{11,5,3\}$-code.

An interesting/important example of CSS codes are quantum Reed-Muller Codes, see Section 7.4.1 for classical versions, due to Steane (1996b) and Zhang and Fuss (1997).

Steane (1998a) has also developed a method how to convert certain CSS codes into quantum codes with better parameters.

Analysis of orthogonality conditions and search for signs of the basis states

This method, introduced by Laflamme et al. (1996), is simple in principle, but elaborate, and was used to get the first “perfect” encoding of one qubit by 5 qubits and 8 basis states for each of the codewords —see Table 7.2.

Exercise 7.4.48 *(Laflamme et al. 1996) Assume that an encoding has the form*

$$|0\rangle \to \sum_{i=0}^{2^n-1} \alpha_i|i\rangle, \qquad |1\rangle \to \sum_{i=0}^{2^n-1} \beta_i|i\rangle$$

and that all states obtained by one qubit error are orthogonal. Show that in such a case the following equalities have to hold:

$$\sum_{i_k=0, i_l=0} |\alpha_i|^2 = \sum_{i_k=0, i_l=1} |\alpha_i|^2 = \sum_{i_k=1, i_l=0} |\alpha_i|^2 = \sum_{i_k=1, i_l=1} |\alpha_i|^2. \tag{7.23}$$

for all $0 \leq k \neq l < n$ *(where* j_s *denotes sth bit of* j *expressed in binary).*

Encodings of $|0_E\rangle$ and $|1_E\rangle$ should be such that all states obtained from them by one qubit error should be orthogonal. This gives rise to a bunch of conditions the amplitudes have to satisfy, such as one in (7.23). These conditions then have to be analysed to determine (to guess) the support for the code. As reported in Laflamme et al. (1996), this way they found the support for the code shown in Figure 7.7.

If one then takes a natural assumption of simplicity, namely that all amplitudes of the basis states are either $+1$ or -1, then a computer search can find out quantum codes with a given support and amplitudes ± 1. Laflamme et al. (1996) discovered that all such minimal encodings of one qubit have to have two -1 as amplitudes in one codeword and four in other. An example of a modified version of their code (see Williams and Clearwater, 1998), has the form shown in Table 7.7:

Code	$\|0_E\rangle$	$\|1_E\rangle$
Modified LMPZ's code	$+\|00000\rangle - \|11100\rangle + \|10011\rangle - \|01111\rangle$ $+\|11010\rangle + \|00110\rangle + \|01001\rangle + \|10101\rangle$	$-\|00011\rangle + \|11111\rangle - \|10000\rangle + \|01100\rangle$ $+\|11001\rangle - \|00101\rangle - \|01010\rangle + \|10110\rangle$

Figure 7.7: Modified version of LMPZ code

Exercise 7.4.49 *Determine whether each code with the same string-support as LMPZ code has to have four amplitudes* -1 *at basis states in one of codewords and two at the other.*

7.4.5 Stabilizer codes

A general class of quantum error-correcting codes was introduced by Gottesman (1996) and Calderbank and Shor (1997) and included all QECC known till that time. See also Gottesman (1997) for a detailed treatment of the subject.

Basic concepts—code stabilizer

The basic elements to work with at this approach are error operators

$$E_1 \otimes E_2 \otimes \ldots \otimes E_n, \tag{7.24}$$

where each $E_i \in \{I, X, Y, Z\}$ and $X = \sigma_x$, $Z = \sigma_z$ and $Y = XZ = i\sigma_y$, where σ_x, σ_y and σ_z are Pauli matrices. An error operator (7.24) is said to have **weight** t if it has t component matrices different from I.

Denote by X_i, Y_i and Z_i operator/matrices X, Y and Z acting on the ith qubit and let $\mathcal{G}_n$ be the group generated by $3n$ of such matrices.

Properties of $\mathcal{G}_n$: $X_i^2 = Z_i^2 = I = -Y_i^2$ and $Y_i = -X_iZ_i = Z_iX_i$, for $1 \leq i \leq n$. $\mathcal{G}_n$ has order $2 \cdot 4^n = 2^{2n+1}$ (there are four possible elements for each i and the sign). In addition, each element of $\mathcal{G}_n$ squares to ± 1 and if $A, B \in \mathcal{G}_n$, then A and B either commute or anticommute, i.e., either $[A, B] = 0$ or $\{A, M\} = 0$, see page 369.

Let C be a quantum error-correcting code of H_{2^n}. C spans a subspace of H_{2^n}. The group $\mathcal{G}_n$ can be seen as acting on states of C. A **stabilizer** $\mathcal{S}_C$ of C is the set

$$\mathcal{S}_C = \{M \in \mathcal{G}_n \mid M|\phi\rangle = |\phi\rangle \ \text{ if } \ |\phi\rangle \in C\}.$$

The following property is of crucial importance for the "stabilizer codes" to be defined later:
If $M \in \mathcal{G}_n$ and $S \in S_C$ are such that $\{M, S\} = 0$, then for any $|\phi\rangle$, $|\psi\rangle \in C$,

$$\langle\phi|M|\psi\rangle = \langle\phi|MS|\psi\rangle = -\langle\phi|SM|\psi\rangle = -\langle\phi|M|\psi\rangle$$

and therefore $\langle\phi|M|\psi\rangle = 0$. The code herefore satisfies the condition 7.11 whenever $M_a{}^*M_b$ anticommute with some element of S.

M_1	Z	Z	I	I	I	I	I	I	I
M_2	Z	I	Z	I	I	I	I	I	I
M_3	I	I	I	Z	Z	I	I	I	I
M_4	I	I	I	Z	I	Z	I	I	I
M_5	I	I	I	I	I	I	Z	Z	I
M_6	I	I	I	I	I	I	Z	I	Z
M_7	X	X	X	X	X	X	I	I	I
M_8	X	X	X	I	I	I	X	X	X

(a)

M_1	X	X	X	X	I	I	I
M_2	X	X	I	I	X	X	I
M_3	X	I	X	I	X	I	X
M_4	Z	Z	Z	Z	X	X	X
M_5	Z	Z	I	I	Z	Z	I
M_6	Z	I	Z	I	Z	I	Z

(b)

M_1	X	Z	Z	X	I
M_2	I	X	Z	Z	X
M_3	X	I	X	Z	Z
M_4	Z	X	I	X	Z

(c)

M_1	X	X	X	X	X	X	X	X
M_2	Z	Z	Z	Z	Z	Z	Z	Z
M_3	I	X	I	X	Y	Z	Y	Z
M_4	I	X	Z	Y	I	X	Z	Y
M_5	I	Y	X	Z	X	Z	I	Y

(d)

Figure 7.8: Stabilizers

Example 7.4.50 *Figures 7.8a,b,c (Gottesman, 1997), shows generators of the stabilizers for Shor's 9-qubit code, Steane's 7-qubit code and LMPZ's 5-qubit code. Figure 7.8d show generators of the stabilizer for an $[[8, 3, 3]]$-code due to Gottesman (1996). Let us discuss design and use of the stabilizer for Steane's code. Error vectors in Figure 7.8b can be discovered in a straightforward way from how one detects a single bit or sign error for this code. Indeed, to detect a bit error in a state $|\psi\rangle$ on one of the first three qubits it is sufficient* *to compare the first qubit with second and then the first qubit with the third in $|\psi\rangle$. One way of doing that is to measure $|\psi\rangle$ with respect to M_1 and M_2 as observables. A similar role play error vectors M_3 to M_6. The last two error vectors can be used to detect sign errors.*

Exercise 7.4.51 *Show that if $\{M_1, \dots, M_s\}$ is a stabilizer for a code C, then so is $\{M_1, \dots, M_{i-1}, M_iM_j, M_{i+1}, \dots, M_s\}$, for any $1 \leq i, j \leq s$.*

Exercise 7.4.52 *Show that if a code encodes k qubits using n qubits, then its stabilizer has dimension 2^{n-k}.*

Implications? Let E and F be error operators, both of weight t or less. If the operator F^*E anticommutes with any operator in $\mathcal{S}$, then vectors $E|\phi\rangle$ and $F|\psi\rangle$ are orthogonal for any $|\phi\rangle, |\psi\rangle \in C$. However, this is exactly the requirement a non-degenerate quantum error-correcting code should satisfy. In other words, to get a non-degenerate code we just need to find a code C and the corresponding stabilizer $\mathcal{S}_C$ such that any non-identity error operator of $\mathcal{G}_n$ of weigth at most $2t$ anticommutes with some element from $\mathcal{S}_C$.

The above result suggests two methodologies to design non-degenerate codes: To choose C and to look for $\mathcal{S}_C$ or vice versa.

In order to facilitate the second methodology let us look for some properties $\mathcal{S}_C$ should have to be a stabilizer.

$\mathcal{S}_C$ is clearly a subgroup of $\mathcal{G}_n$. In addition, for any $|\psi\rangle \in C$, $S \in \mathcal{S}_C$, $S^2|\psi\rangle = S|\psi\rangle = |\psi\rangle$ and therefore $S^2 = I$. Moreover, for any $|\phi\rangle \in C$ and any $M, N \in \mathcal{S}_C$, $MN|\psi\rangle = NM|\psi\rangle = |\psi\rangle$ and therefore $[M,N]|\psi\rangle = 0$. We have therefore that either $[M,N] = 0$ or $\{M,N\} = 0$. In the case $\{M,N\} = 0$ we would have $[M,N] = 2MN$, but this contradicts the property $[M,N]|\psi\rangle = 0$ for any $|\psi\rangle \in C$. Hence $[M,N] = 0$. $\mathcal{S}_C$ has therefore to be an Abelian group. $S^2 = I$ has to hold for any of its elements S. It can be shown that these conditions are sufficient for the existence of a non trivial code C for which $\mathcal{S}_C$ is a stabilizer (provided S_c is not too big).

Let C be a code of codewords of length n and $\mathcal{S}_C$ be its stabilizer. The **centralizer** of $\mathcal{S}_C$, $\mathcal{C}(\mathcal{S}_C)$ is the set of elements of $\mathcal{G}_n$ that commute with all elements of $\mathcal{S}_C$, Clearly $\mathcal{S}_C \subseteq \mathcal{C}(\mathcal{S}_C)$. Define $\mathcal{F}_\mathcal{S} = \mathcal{S}_C \cup (\mathcal{G}_n - \mathcal{C}(\mathcal{S}_C))$. The code C corrects any set $\mathcal{E}_0$ of errors such that for any $M_a, M_b \in \mathcal{S}_\mathcal{E}$, $M_a^* M_b \in \mathcal{S}_C \cup (\mathcal{G}_n - \mathcal{C}(\mathcal{S}_C))$.

For the stabilizer codes there are straightforward ways to design networks for encoding—see Gottesman (1997)—and also for syndrom computation, see DiVincenzo and Shor (1995).

Gottesman (1996) has developed a methodology for choosing $\mathcal{S}$ so that error operators of length at most $2t$ will anticommute with some member of $\mathcal{S}$. On this base he designed an infinite class of codes saturating the quantum Hamming bound. One of them is $\{8,3,3\}$-code whose stabilizer is shown in Figure 7.8d.

The concept of the stabilizer has much simplified the design of quantum error correcting codes—see Gottesman (1996), Steane (1996) and Calderbank et al. (1996, 1997).

Special notation

In order to describe stabilizers also a different notation is used which is especially handy in the case of the quantum error-correcting codes derived from the classical codes. An error operator $M_1 \otimes \ldots \otimes M_n$ is written in the form $X(x_1, \dots, x_n)Z(z_1, \dots, z_n)$, where $x_i = 1$ if $M_i \in \{X, Y\}$, and 0 otherwise, and $z_i = 1$ if $M_i \in \{Y, Z\}$, and 0 otherwise. Moreover, in such a case also the notation $(X_x|Z_z)$ is used, where x_i and z_i are defined as above.

A set of error operators forming generators of a stabilizer can then be written in the matrix form

$$(X|Z), \tag{7.25}$$

For example, stabilizers for the LMPZ code and for the {8,3,3 }code mentioned above are in this form described in Figure 7.9a,b.

If X and Z are $(n-k)\times n$ matrices for some n and k then $(X|Z)$ forms a stabilizer if $XZ^T + ZX^T = 0$ (Calderbank et al. 1997).

Stabilizers for CSS-codes

Let us consider a CSS code created out of two classical linear codes C_1 and C_2 and let P_{C_1} and P_{C_2} be their parity-check matrices.

A quantum code to correct just bit errors can be designed using the set of generators S_{C_1} obtained from vectors of P_{C_1} by replacing each 1 with Z. To that we add the set of generators obtained from the parity-check matrix P_{C_2} for C_2 with X replacing 1's this time. These generators can identify the sign errors. Together they identify also bit/sign errors. In the case $C_2^\perp \subseteq C_1$, $C_1^\perp \subseteq C_2$ these two generators can be combined into a single set of generators for the code.

$$\begin{pmatrix} 1&0&0&1&0&|&0&1&1&0&0\\ 0&1&0&0&1&|&0&0&1&1&0\\ 1&0&1&0&0&|&0&0&0&1&1\\ 0&1&0&1&0&|&1&0&0&0&1 \end{pmatrix} \qquad \begin{pmatrix} 1&1&1&1&1&1&1&1&|&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&|&1&1&1&1&1&1&1&1\\ 0&0&0&0&1&1&1&1&|&0&0&1&1&0&0&1&1\\ 0&0&1&1&0&0&1&1&|&0&1&0&1&0&1&0&1\\ 0&1&0&1&0&1&0&1&|&0&0&1&1&1&1&0&0 \end{pmatrix}$$

(a) (b)

Figure 7.9: Generators of two CSS-codes

Beyond stabilizer codes

The first example of a non-stabilizer code that is better than any other stabilizer code of the same type is due to Rains et al. (1997). It is a {5,6,2}code that is defined (as a subspace) by the following projection operators.

$$\begin{aligned} P \;=\; & \frac{1}{16}[3(I\otimes I\otimes I\otimes I\otimes I) + (I\otimes\sigma_z\otimes\sigma_y\otimes\sigma_y\otimes\sigma_z)_{cyc} + (I\otimes\sigma_x\otimes\sigma_z\otimes\sigma_z\otimes\sigma_x)_{cyc} \\ & -(I\otimes\sigma_y\otimes\sigma_x\otimes\sigma_x\otimes\sigma_y)_{cyc} + 2(\sigma_z\otimes\sigma_x\otimes\sigma_y\otimes\sigma_y\otimes\sigma_x)_{cyc} - 2\sigma_z\otimes\sigma_z\otimes\sigma_z\otimes\sigma_x\otimes\sigma_z] \end{aligned}$$

where the subscript "*cyc*" denotes that all five cyclic shifts of the corresponding error operator have to be taken into the sum. The code was discovered by a combination of careful reasoning and computations. The code is better than any other stabilizer code of the same type in the sense that for each stabilizer $(n,6,2)$-code we have $n\leq 4$.

7.5 Fault-tolerant Quantum Computation

There are five main reasons why the discovery of quantum error-correcting codes is not by itself a sufficient guarantee that quantum error-correction will be practically successful on "noisy quantum computers".

1. Quantum error-correcting codes help to deal only with the problem of reliable storage and transmission of quantum information. They are by themselves insufficient to have fault-tolerant quantum information processing.

2. Each use of quantum error correction methods brings additional requirements on quantum memory, hardware and computation time. Indeed, to establish necessary redundancy, additional qubits, and thereby additional quantum memory, are needed. In addition encoding, error recovery and decoding operations require additional qubits, gates and circuits. That may slow down the overall computation.

3. Quantum error-correction processing networks are themselves non-trivial quantum networks and therefore, naturally, errors can occur during their performance.

What is therefore needed for quantum error-correction to be practically successful are such networks for error-correction processing that are sufficiently robust to succeed with high reliability even if some errors are made during error-correction processing operations themselves. In short, fault-tolerant quantum networks are what is practically needed. What is especially needed are fault-tolerant networks to compute directly with encoded qubits without any need to decode them.

It was first shown by Shor (1995) that for any polynomial size quantum circuit an equivalent and polynomial size fault-tolerant version can be constructed which can tolerate after t steps inaccuracies and decoherence averaging $\mathcal{O}(\frac{1}{\lg^c t})$ per step for a small constant c.

4. It is not sufficient that we can encode quantum information in such a way that it is stored or transferred reliably for some time. Of key importance for real quantum computing is that we can store and transmit quantum information reliably for a long time and through a long distance.

5. It is also not sufficient that we can process quantum information for some time in a fault-tolerant way. What is badly needed is to be able to do that for a sufficiently long time.

The so-called concatenated codes (see Section 7.5.3), represent a way to deal with the last two problems. Concatenated codes allow us to store and transmit a qubit with maximal error ε, provided gates with errors at most $c\varepsilon$ (where c is a constant not dependent on ε) and storage or channel elements with errors at most ε are given, independently of how long we must store a state or how far we need to transmit it.

Remark 7.5.1 A way to cope with the main drawback of concatenated codes—still high requirements on tolerable error probabilities for transmissions and local operations—has been suggested by Briegel et al. (1998). They suggest to use "quantum repeaters to form entanled pairs for an arbitrarily large distance. The main idea is to use a sequence of imperfectly entangled pairs of particles and a new (nested) purification protocol, to create a single distant pair of particles of high fidelity.

Methods presented in this section allow us to cope with inaccuracies and decoherence problem in three ways: (1) Fault-tolerant circuits are used to perform quantum gates information processing; (2) in between two quantum gates applications to qubits quantum error detection and recovery circuits are used to take care of their recovery from errors; (3) concatenated codes and quantum repeaters are used to perform quantum information processing and transmission over long time and distance.

7.5.1 Fault-tolerant quantum error correction

A network is usually called fault-tolerant if it can recover from some errors during its operations. Of course, there are no perfect networks. In addition, to any network it can happen that all gates malfunction. That is why we need to take a modest but still useful view of what is meant by saying that a network is considered as being fault-tolerant.

We shall be in the following happy to consider a network as fault-tolerant if the probability that the network malfunctions is at most ε^2, where ε is the probability that one gate or qubit error occurs.

Let us first analyze possible ways and circumstances in which error-correcting codes operations may fail, and what has to be changed to have fault-tolerant networks for error-correction operations (Preskill, 1998).

Error propagation

The first main problem we have to learn to deal with is **error propagation** due to entanglement. If a two-qubit gate is applied and one of qubits is erroneous, then the error can propagate to the second qubit.

Example 7.5.2 *As we could see in the previous section, XOR gates are of key importance for quantum error-correcting codes processing operations, especially for error syndrome computation. It is evident that if a error occurs in an XOR's gate control qubit, then the error propagates ("forwards") to its target qubit. Less obvious, but very important for quantum error-correction processing networks, is the fact that under certain circumstances an error can propagate through an XOR gate also "backwards", from the target qubit to the control qubit. This is easily seen from Figure 2.11, where a relation between an XOR gate and its reverse is shown at which control and target bits are interchanged. This network is a very typical component of quantum error correction networks. Indeed, Hadamard rotations applied to qubits change the basis from the standard to the dual (and vice versa) and interchange bit and sign errors. Therefore if a sign error occurs on the target qubit, it can become a bit error of the source qubit.*

Backward error propagation in XOR gates discussed in the above example implies that it is not good to use for error syndrome computation such circuits as in Figure 7.10a, in which one ancilla qubit is used "to xor" information from several data qubits. Indeed, a spontaneous phase error at an ancilla qubit, a far from an unexpected event, could propagate to a bit error in several code qubits. That is why a way of gathering information for syndrome computation depicted in Figure 7.10b is surely superior with respect to error propagation (even if it requires more qubits).

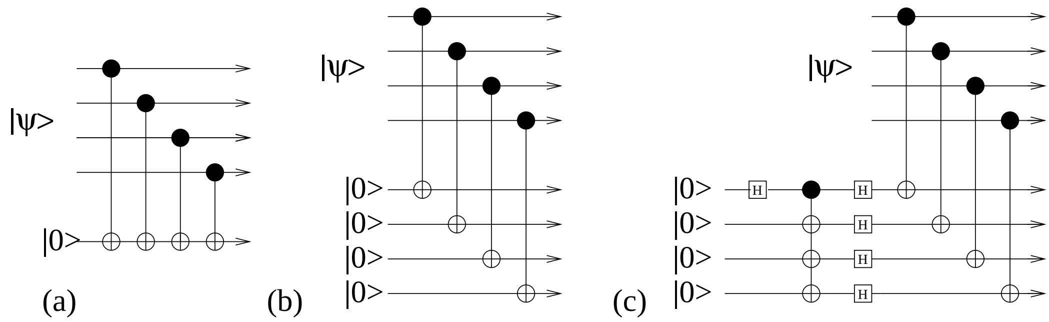

Figure 7.10: Three ways of gathering information from data to ancilla: bad, better and the best

Fault-tolerant ancillas

Unfortunately, even the way of gathering information shown in Figure 7.10b is not good enough for fault-tolerant quantum error correction. The point is that code qubits keep being entangled with the ancilla qubits and therefore a measurement of ancilla qubits can destroy the encoded state $|\psi\rangle$.

Example 7.5.3 *In the case of Steane's codendexcode!Steane's we have seen that information about its last* 4 *qubits is needed to get the first bit of the syndrome* $(x_4 \oplus x_5 \oplus x_6 \oplus x_7)$. *This information can be obtained using a four-qubit ancilla, with all qubits initially in the state* $|0\rangle$, *one qubit of ancilla for each bit of the syndrome. However, if the ancilla qubits yield, after the measurement, the result* 0101, *then in* $|\psi\rangle$ *the state* $|0_E\rangle$ *collapses to the state* $|1010101\rangle$, *and the state* $|1_E\rangle$ *collapses to the state* $|0100101\rangle$, *which is not acceptable because the state lost all protection against phase errors..*

There are three main operations to work with quantum error-correcting codes: **encoding** (code generation), **error recovery** (syndrome computation and code correction) and **decoding**. Networks to perform these operations, especially code generation and error recovery normally require an ancilla. It is therefore of crucial importance for the design of a fault-tolerant quantum network to find a way of copying information into the ancilla without distractive effects on the state being "copied".

One way out was found by Shor (1995) and it is illustrated in Figure 7.10c for the case of an ancilla with 4 qubits. Information is copied into the ancilla with the initial (Shor) state

$$|\phi\rangle = \frac{1}{\sqrt{8}} \sum_{i=0}^{7} |parity(i)\rangle$$

of the equal superposition of all even-parity 4-bit codewords. This state can be created in the way shown in Figure 7.10c. (The first Hadamard gate and the multiple XOR create the "cat state" $\frac{1}{\sqrt{2}}(|0000\rangle + |1111\rangle)$. The next four Hadamard gates then create $|\phi\rangle$.

Why is this way of copping information into the ancilla in the state $|\phi\rangle$ better? What is needed in the case of a syndrome computation is only information about the parity of four bits copied into the ancilla. If this parity is 0, the state $|\psi\rangle$ is not changed; if the parity is one, then the state of ancilla changes into the equal superposition of all odd-weighted 4-bit codewords. If we therefore measure the ancilla state, with respect to the standard basis in

H_{16}, we get a four-bit word, randomly, and its parity is the corresponding syndrome. The key point is that after XOR operations in this case the state of the ancilla is not entangled with encoded qubit and therefore a measurement of ancilla does not hurt it!

Verification of the ancilla state

Another potential problem and a potential source of errors is that the initial ancilla state may be erroneous, and a sign error that occurs during the preparation of the ancilla may propagate to the encoded qubit.

In the case of the 4-qubit ancilla, and its circuit in Figure 7.10c, a single qubit error during the second or third XOR gate (of the multiple XOR gate) can result in two bit errors in the cat state and consequently in two sign errors in the initial state of the ancilla. That will cause errors during the syndrome measurement.

A suggested way to test the cat state is to perform two XOR gates on two randomly chosen qubits of the cat state, and two new qubits originally in the state $|0\rangle$, and discard the resulting state if the measurement of these two new qubits does not agree. In such a case the ancilla has to be constructed again. Of course this means that one needs to use additional qubits.

For the case of 4-qubits, as in Figure 7.10c, there is a simpler way of verifying the cat state. It is based on the observation that for all the ways a single XOR gate can cause two bit errors in the cat state, the first and the fourth qubit will have different values. To verify the cat state it is therefore sufficient to use two additional XOR gates as shown in Figure 7.11.

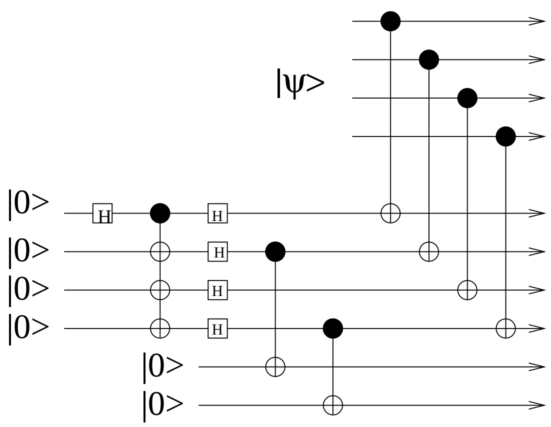

Figure 7.11: Ancilla state verification

Another possible case is that a single error at the cat state designing circuit could cause two sign errors in the cat state and therefore two bit-errors in the initial Shor's state. However, such errors do not damage the encoded qubits; they can only cause that the resulting syndrome measurement provides an incorrect result.

Testing of syndrome measurements

It has been shown that syndrome measurement circuits do exist, but this does not guarantee yet that syndrome measurement performed by such circuits will be correct. The main way to be sure with that is to repeat each syndrome measurement several times, until the same error syndrome is obtained k times in a row, where k is some reliability parameter.

Fault-tolerant syndrome computation for additive codes

A simple procedure for syndrome generation for stabilizer codes was developed by DiVincenzo and Shor (1996).

To each generator $X(x_1, \dots, x_n)Z(z_1, \dots, z_n)$ one stage in the syndrome computation circuit corresponds. Each stage consists of three phases: basis transformations on qubits, syndrome gathering and undoing of the previous basis. The basic idea is to transform the generator to the form

$$X(0, \dots, 0)Z(z_1', \dots, z_n').$$

In order to achieve it the basis of the ith qubit is done by the matrix I if $x_i = 0$, by Hadamard rotation if $x_i = 1$ and $z_i = 0$ and by the matrix $H' = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix}$, if $x_i = z_i = 1$. After this change of the basis $z_i' = x_i \vee z_i$. The next part of the circuit for a given stage is to determine the parity of qubits with non-zero entries in Z. In order to do that fault-tolerantly one can use ancilla in the Shor state. Once this is done the stage ends by undoing the basis. This way the syndrome computing circuit for the LMPZ's code shown in in Figure 7.6 works.

Why does this work? The point is that each measurement of the quantum state, corresponding to one of the generators of the generator matrix G of the code, yields the eigenvalue of the quantum state with respect to that matrix. By that we can reduce the number of eigenspaces in which the quantum state may lie by a factor of 2. Therefore, if the measurement is done for every generator of G, the complete set of eigenvalues of the state with respect to G is known. By that one of the error eigenspaces is determined and that implies which unitary transformation is to be applied to undo the error.

Exercise 7.5.4 *Determine the number of gates needed to compute a syndrome in the fault-tolerant way for additive codes.*

7.5.2 Fault-tolerant quantum gates

A fault-tolerant realization of quantum error-correction operations is a step forward, with respect to the reliability of the storage and transmission of quantum information. It also offers one way to do reliably quantum computation: encoded qubits are first decoded, quantum gates are applied on decoded qubits and then qubits are again encoded. Clearly, this approach has a lot of drawbacks. What is really needed is the ability to apply quantum gates directly to encoded qubits and to work in the "protected" code subspace.. This means that what we really need in the first order are fault-tolerant gates.

There seem to be two ways to have fault-tolerant gates. The first way, discussed in more detail in the first part of this section, is to use "software methods": to design a set of quantum circuits that simulates, in a fault-tolerant way, a universal set of gates. The second way, so far only envisioned, is to use "hardware methods"—to use physically inherently fault-tolerant gates. This will be briefly discussed at the end of this section.

Fault-tolerant circuits for a universal set of gates

It will now be shown, following Shor (1996), that fault-tolerant quantum circuits exist for rotation gates, XOR gate and for the Toffoli gate for a special type of quantum codes, designed from the so-called punctured doubly even codes by the general method discussed on page 288.[11] The choice of these special codes allowed us to show in an easy way how fault-tolerant versions of the above set of gates can be implemented. The same can be shown for other codes, as discussed later.

Let us assume that C is a self-dual classical code of length n, dimension $k = \frac{n+1}{2}$ and distance d. Since the dimension of $C^{\perp}$ is $n-k$, we have, on page 288, that the number of different quantum codewords is $2^{2k-n} = 2$. Let us denote them $|0'_E\rangle$ and $|1'_E\rangle$. One of them consists of codewords with an even parity; the second with an odd parity.

Transferring these codes into into dual base yields:

$$|0_E\rangle = \frac{1}{\sqrt{2}}(|0'_E\rangle + |1'_E\rangle) \qquad |1_E\rangle = \frac{1}{\sqrt{2}}(|0'_E\rangle - |1'_E\rangle) \tag{7.26}$$

An important point is that transformation from one basis to another for $\{|0_E\rangle, |1_E\rangle\}$ codewords and $\{|0'_E\rangle, |1'_E\rangle\}$ codewords, can be done by applying Hadamard transformation qubitwise! This implies that the basis changing transformation is fault-tolerant. Indeed, when error-less gates are applied to individual qubits the number of erroneous qubits is preserved. Moreover an erroneous gate application will introduce only one error.

In a similar way it can be shown that $\frac{\pi}{2}$-rotations can be applied qubitwise and therefore in a fault-tolerant way.

Exercise 7.5.5 *Show that for punctured doubly even self-dual codes the following operations on logical qubits can be performed qubitwise: (a)* $\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$*; (b)* $\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$*; (c)* $\frac{\pi}{2}$*-rotation.*

XOR operation can also be performed qubitwise. Indeed, suppose we have two logical qubits $|d_a\rangle$ and $|d_b\rangle$, see page 289. Then

$$|d_a\rangle|d_b\rangle = \frac{1}{2^{n-k}}(\sum_{w \in C^{\perp}} |w + a\rangle)(\sum_{w' \in C^{\perp}} |w' + b\rangle).$$

By applying XOR from the ith qubit of $|d_a\rangle$ to the ith qubit of $|d_b\rangle$ we get the

$$\frac{1}{2^{n-k}}(\sum_{w \in C^{\perp}} |w + a\rangle)(\sum_{w' \in C^{\perp}} |w' + b + w + a\rangle) = |d_a\rangle|d_{a+b}\rangle,$$

because if $w \in C^{\perp}$, $w' \in C^{\perp}$, then also $w + w' \in C^{\perp}$.

Bitwise can also be done the phase changing operators $|d_a\rangle|d_b\rangle \to (-1)^{a \cdot b}|d_a\rangle|d_b\rangle$.

As the next step a fault-tolerant circuit for the Toffoli gate will be constructed. This will be a harder task.

Let us first assume that we have a fault-tolerant circuit that can perform transformation T: $|i_E, j_E\rangle \to |i_E, j_E, (i \wedge j)_E\rangle$. Using this circuit we can make another fault-tolerant circuit

[11] Doubly even codes are self-dual codes all codewords of which have a number of 1s divisible by 4. A punctured code is obtained from a self-dual code by deleting one bit-component.

which performs a similar transformation:

$$|i_E, j_E, k_E\rangle \to (i_E, j_E, (i \wedge j)_E, k_E\rangle.$$

Let us then attach $\mathrm{XOR}_{4,3}$ and H_4 to the previous circuit outputs. As the result a fault-tolerant circuit is obtained to perform the following transformation:

$$\begin{array}{lcllcl}
|0_E0_E0_E\rangle & \to & \frac{1}{\sqrt{2}}|0_E0_E0_E\rangle(|0_E\rangle + |1_E\rangle) & |0_E1_E0_E\rangle & \to & \frac{1}{\sqrt{2}}|0_E1_E0_E\rangle(|0_E\rangle + |1_E\rangle), \\
|1_E0_E0_E\rangle & \to & \frac{1}{\sqrt{2}}|1_E0_E0_E\rangle(|0_E\rangle + |1_E\rangle) & |1_E1_E0_E\rangle & \to & \frac{1}{\sqrt{2}}|1_E1_E1_E\rangle(|0_E\rangle + |1_E\rangle), \\
|0_E0_E1_E\rangle & \to & \frac{1}{\sqrt{2}}|0_E0_E1_E\rangle(|0_E\rangle - |1_E\rangle) & |0_E1_E1_E\rangle & \to & \frac{1}{\sqrt{2}}|0_E1_E1_E\rangle(|0_E\rangle - |1_E\rangle), \\
|1_E0_E1_E\rangle & \to & \frac{1}{\sqrt{2}}|1_E0_E1_E\rangle(|0_E\rangle - |1_E\rangle) & |1_E1_E1_E\rangle & \to & \frac{1}{\sqrt{2}}|1_E1_E0_E\rangle(|0_E\rangle - |1_E\rangle).
\end{array}$$

A measurement of the fourth qubit with the outcome 0 results in the state $(i_E, j_E, ((i \wedge j) \oplus k)_E)$, the one the Toffoli gate should produce. However, a measurement with outcome 1 produces the state $-|(i_E, j_E, ((i \wedge j) \oplus k)_E\rangle$. In such a case we need to change the sign, which can be done fault-tolerantly, as already mentioned.[12]

As the next step we show how to design a fault-tolerant circuit for the mapping $|i_E, j_E\rangle \to |i_E, j_E, (i \wedge j)_E\rangle$ provided we have a 3-qubit ancilla in the state

$$|A\rangle = \frac{1}{2}(|0_E0_E0_E\rangle + |0_E1_E0_E\rangle + |1_E0_E0_E\rangle + |1_E1_E1_E\rangle).$$

This can be done using the gates that have already been shown to have equivalent fault-tolerant circuits.

Let us first apply gates $\mathrm{XOR}_{3,1}$ and $\mathrm{XOR}_{4,2}$ to the state $|i_E, j_E\rangle|A\rangle$. This way we can realize the mapping

$$\begin{array}{lcl}
|0_E0_E\rangle|A\rangle & \to & \frac{1}{2}(|0_E0_E0_E0_E0_E\rangle + |0_E1_E0_E1_E0_E\rangle + |1_E0_E1_E0_E0_E\rangle + |1_E1_E1_E1_E1_E\rangle), \\
|0_E1_E\rangle|A\rangle & \to & \frac{1}{2}(|0_E1_E0_E0_E0_E\rangle + |0_E0_E0_E1_E0_E\rangle + |1_E1_E1_E0_E0_E\rangle + |1_E0_E1_E1_E1_E\rangle), \\
|1_E0_E\rangle|A\rangle & \to & \frac{1}{2}(|1_E0_E0_E0_E0_E\rangle + |1_E1_E0_E1_E0_E\rangle + |0_E0_E1_E0_E0_E\rangle + |0_E1_E1_E1_E1_E\rangle), \\
|1_E1_E\rangle|A\rangle & \to & \frac{1}{2}(|1_E1_E0_E0_E0_E\rangle + |1_E0_E0_E1_E0_E\rangle + |0_E1_E1_E0_E0_E\rangle + |0_E0_E1_E1_E1_E\rangle).
\end{array}$$

As the next step we measure the first two logical qubits. The outcome 0_E0_E results in the transformation $|i_E, j_E, i_E, j_E, (i \wedge j)_E\rangle$. The outcome 0_E1_E results in a mapping $|i_E, j_E, u_E, v_E, w_E\rangle$ such that

$$NOT_4XOR_{3,5}|i_E, j_E, u_E, v_E, w_E\rangle = |i_E, j_E, i_E, j_E, (i \wedge j)_E\rangle,$$

which is what is needed. In a similar way we can transform the results of other measurements outcomes, using XOR and NOT operations, to the desired mapping.

The last step is to show that the ancilla state $|A\rangle$ can be designed in a fault-tolerant way. Of course, this cannot be done using XOR and rotation gates only.

Let us consider two $3n$-qubit states

$$\begin{array}{lcl}
|A\rangle & = & \frac{1}{2}(|0_E0_E0_E\rangle + |0_E1_E0_E\rangle + |1_E0_E0_E\rangle + |1_E1_E1_E\rangle), \\
|B\rangle & = & \frac{1}{2}(|0_E0_E1_E\rangle + |0_E1_E1_E\rangle + |1_E0_E1_E\rangle + |1_E1_E0_E\rangle),
\end{array}$$

[12]One can make use of the trasformations $|i_Ej_Ek_E\rangle \to (-1)^{ij}(- - 1)^k|i_Ej_Ek_E\rangle$ which can be seen as a composition of operations $|i_Ej_E\rangle \to (-1)^{ij}|i_Ej_E\rangle$ and $|k_E\rangle \to (-1)^k|k_E\rangle$.

i.e., $|A\rangle = \mathrm{NOT}_3|B\rangle$. Moreover,

$$\frac{1}{\sqrt{2}}(|A\rangle + |B\rangle) = \frac{1}{\sqrt{8}}(|0_E\rangle + |1_E\rangle)(|0_E\rangle + |1_E\rangle)(|0_E\rangle + |1_E\rangle)$$

and the last state is easy to construct using gates, the fault-tolerance of which has already been demonstrated.

Let us first assume that we have fault-tolerant gates performing transformations

$$\begin{array}{llllll} |0^{(n)}\rangle|A\rangle & \rightarrow & |0^{(n)}\rangle|A\rangle, & |1^{(n)}\rangle|A\rangle & \rightarrow & |1^{(n)}\rangle|A\rangle, \\ |0^{(n)}\rangle|B\rangle & \rightarrow & |0^{(n)}\rangle|B\rangle, & |1^{(n)}\rangle|B\rangle & \rightarrow & -|1^{(n)}\rangle|B\rangle. \end{array}$$

Once this is done, $|A\rangle$ can be designed as follows. The state $\frac{1}{\sqrt{2}}(|0^{(n)}\rangle + |1^{(n)}\rangle)(|A\rangle + |B\rangle)$ is first designed and then the transformation given just above is applied to produce

$$\frac{1}{\sqrt{2}}(|0^{(n)}\rangle + |1^{(n)}\rangle)|A\rangle + \frac{1}{\sqrt{2}}(|0^{(n)}\rangle - |1^{(n)}\rangle)|B\rangle.$$

Finally, let us measure the first n qubits with respect to the basis

$$\{\frac{1}{\sqrt{2}}(|0^{(n)}\rangle + |1^{(n)}\rangle), \frac{1}{\sqrt{2}}(|0^{(0)}\rangle - |1^{(n)}\rangle)\}.$$

This measurement shows whether unmeasured qubits are in the state $|A\rangle$ or $|B\rangle$. In the second case we can obtain $|A\rangle$ from $|B\rangle$ by a NOT operation.

Remark 7.5.6 A general theory of fault-tolerant operations for stabilizer codes has been developed by Gottesman (1997). It is based on symmetries of the code stabilizer and it is shown that fault-tolerant universal computation is possible for any stabilizer code. Gottesman (1997) discussed in detail fault-tolerant computation based on LMPZ code.

Remark 7.5.7 Another general method, this time for a class of CSS-codes, to build a universal set of fault tolerant operations developed Steane (1998b). In addition, he has introduced new techniques to restrict accumulation of errors before and during recovery operations. On this basis he develops an optimistic evaluation of perspectives of quantum computing, from a certain point of view. Under the standard noise model of stochastic, uncorrelated errors, a quantum computer needs be only an order of magnitude larger than the logical machine contained within it in order to be reliable. For example, a scale-up by a factor 22, with error rate of order 10^{-5}, is sufficient to permit large quantum algorithms, such as factorization of thousand digit numbers.

Fault-tolerant gates

A different approach to fault tolerance of quantum gates was worked out by Kitaev (1997). He showed, theoretically, that there is a universal set of gates that are fault-tolerant by their physical nature and therefore they should be insensitive to local influences (and could be operated quite carelessly).

The possibility of a near-future physical realization of such gates is far from clear or, at least, it does not seem to be in sight. However, Kitaev's approach at least indicates that there may be essentially different ways to stabilization of quantum computations than those discussed above—to search for new physical principles that would lead to fault-tolerant quantum hardware.

Accuracy needed for fault-tolerant quantum computing

It is important to determine the upper bound on qubit and gate errors which still allows fault-tolerant quantum computing to be made sufficiently reliable.

If a code can correct e errors and the probability of one qubit or gate error is ε, then the probability of an error per gate working on encoded qubit is $\binom{m}{e+1}\varepsilon^{e+1}$, where m is the number of places an error can occur in an implementation of the gate, much smaller than ε. It would therefore seem that by choosing e large enough we can make the probability of an error per gate on encoded qubits as small as required. Unfortunately, this idea does not work because as e increases so does the complexity of the code and of all error processing-operations (especially syndrome computations).

For example, if the number of gates needed to perform the syndrome measurement increases as $\Theta(e^b)$, then the probability that $e+1$ errors accumulate before syndrome measurement is complete grows as $\Theta((e^b\binom{m}{e+1}\varepsilon)^{e+1})$.

In order to perform t steps of computation with a low probability of failure a quantum code is needed that can correct $\mathcal{O}(\lg t)$ errors. As analysed by Shor (1995), using the so-called self-dual Reed–Muller code, this can be achieved with codewords of length $\Theta(\lg^2 t)$. Using such codes, computation of syndromes requires the number of gates proportional to the number of 1s in the parity check matrix, which is $\Theta(\lg^3 t)$. If, to make computation reliable, syndrom computing is done $\Theta(\lg t)$ times, then the total number of gate operations needed is $\mathcal{O}(\lg^4 t)$. This implies that gates with error rate $\mathcal{O}(\frac{1}{\lg^4 t})$ are needed. (No much better estimation seems to be known.) This seems to imply that in any case there is a limit to how long a computation can be performed until an error is likely.

7.5.3 Concatenated coding

Fault-tolerant error-correcting codes and fault-tolerant gates are still not enough to ensure long reliable quantum computations. The problem is that these techniques require additional (often noisy) gates, additional computational overhead and, unless the basic error rate is very low, they may introduce more errors than they fix. In addition, the number of gates and time required may increase rapidly with the number of errors corrected. In total, error correction and fault-tolerant techniques are by themselves insufficient to cope with the decoherence and inaccuracies. What is required is a class of codes for which time needed to compute syndromes increases only slowly with error-correcting capabilities of the code and new methods and tools are needed that make long reliable quantum computations and transmissions possible.

Basic idea

The method of Knill and Laflamme (1996) shows one way how to deal with the problem. It allows one to transmit or store a qubit with an error at most ε, regardless of time and distance, provided the gates that are used work with an error at most $c\varepsilon$, for a constant c independent of ε, and storage or channel elements with an error ε at most, for some c and ε.

The basic idea is simple: to encode qubits recursively up to a certain level of recursion or hierarchy (see Figure 7.12), and to perform often recovery operations. The overhead of the method is polynomial in the time and storage and in the distance of transmission. The method could be implemented by having quantum repeaters used and spread at regular time or distance intervals of the quantum channel with sufficiently many parallel paths. The

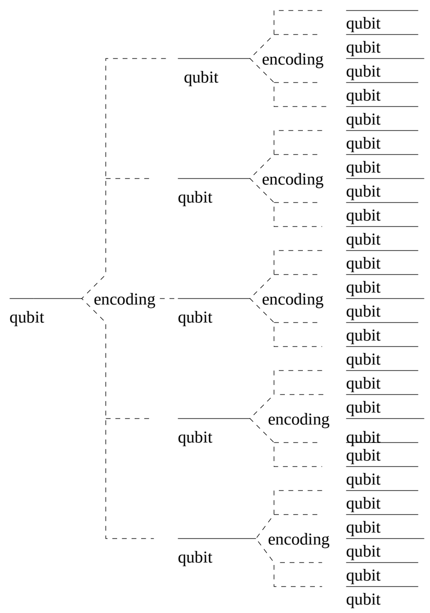

Figure 7.12: Concatenated codes

method can work in principle with any error-correcting code. Concerning the nature and frequency of errors it is only assumed that eerors occur randomly and independently and that "no leakage" assumption (see page 276) is satisfied.

For example, the data is first encoded using an $[[n,k,d]]$-code. Qubits of new codewords are then encoded using an $[[n_1,1,d_1]]$-code. The resulting qubits are again encoded using some $[[n_1,1,d_2]]$-code. This can continue up to a hth hierarchy level resulting in an $[[nn_1 \dots n_{h-1}, k, dd_1 \dots d_{h-1}]]$-code. In a special case all these encodings can be the same. The case $n=5$, $k=1$, $h=2$ is illustrated in Figure 7.12. Computation of the error syndromes can be done in the time proportional to the sum of efforts to do so for all used codes provided we can do that in parallel for all blocks of each hierarchy level once starting with the last hierarchy level.

It is easy to see why several hierarchies of encodings can help. If a single encoding of one qubit into n is used, ε is a probability of an error per qubit, errors are uncorrelated and recovery is fault-tolerant, then the probability of error recovery failure is ε^2. However, if h level of encodings are used, then the probability of an error failure is ε^{2^h} (with the number of qubits n^h).

Recursive encoding has to be combined, in addition, with frequent recovery operations. The method can be simply described, for example (see Knill and Laflamme, 1996), in terms of the recursive concatenated coding procedure $\mathrm{CCP}_{h,r}$, where the parameter h specifies the depth of recursion and r the frequency of the error-recovery operations.

The basic level of the recursion, the procedure, $\mathrm{CCP}_{1,r}$, begins with one qubit, encodes it to n qubits using the code C, then applies the recovery operation $r-1$ times and, finally, decodes it back to a single qubit.

$\mathrm{CCP}_{h,r}$, $h>1$, starts also with one qubit, encodes it into n qubits using the code C, applies $\mathrm{CCP}_{h-1,r}$ to each of n qubits of the code, applies $r-1$ times the recovery operation, then applies again $\mathrm{CCP}_{h-1,r}$ to each qubit and, finally, decodes the state into a single qubit.

Of course, two following recovery operations are to be performed only after some time or distance interval—after the so called recovery period.

Resources needed. If $\mathrm{CCP}_{h,r}$ is used, then the total number of (waiting or transmitting) periods is r^h and the total number of qubits needed is n^h. The number $p(h)$ of error-correcting operations (encodings, recoveries and decodings) is given by the recursion

$$p(1) = r + 1, \quad p(h+1) = r + 1 + rp(h)$$

with a solution $p(h) = \frac{r+1}{r-1}(r^h - 1)$.

The key problem is to determine the error threshold below which arbitrarily long computation is possible (Knill, Laflamme, Zurek (1996, 1997), Gottesman (1997)).

Example 7.5.8 *An analysis of the method for the case of LMPZ's code, made by Knill and Laflamme (1996), shows that a qubit can be stored for arbitrary amount of time or transmitted over arbitrary distance with error ε provided the following holds:*

$\varepsilon \geq \frac{1}{120}$. *The basic storage or channel elements have error at most $\frac{1}{120}$ and one or two qubit gates have error at most $\frac{1}{21600}$.*

$\varepsilon < \frac{1}{120}$. *The basic storage or channel elements have error at most ε and one or two qubit gates have error at most $\frac{\varepsilon(1-60\varepsilon^2)}{90}$.*

The concatenated coding method prefers to use such codes at which all gates used in the error-recovery operations can be performed **transversally**, i.e., a gate can be performed on a logical qubit by being performed on its particular qubits in a qubitwise manner. (For example Steane's 7-qubit code has this property.) If this is the case, then error-recovery operations can be performed simultaneously on all qubits at the highest level of hierarchy.

7.6 Experimental Quantum Processors -

It is still much too early to give a definite answer to the most basic question of quantum information processing from the practical point of view: will we soon (ever) have powerful quantum computers? At the moment the design of quantum gates and processors is in a very rudimentary stage—in its infancy. However, the experimental physics community have launched several projects, with various ingenious ideas, to build at least restricted working prototypes, and two very simple quantum algorithms, using 2 qubits, have already been "implemented in hardware" (see Chuang Chuang et al. 1998, 1998a, and Jones and Mosca, 1998), as well as a simple error-correcting method using 3 qubits (see Cory et al. 1998), and works on implementations of more qubit processors have been reported. The progress in the last years has been encouraging. However, there are still too many hard and basic problems to solve.

It has also been fully realized that the success of such academic projects may not be sufficient. Even if quantum computers could be built, scaling this up to computers large enough to be really significantly more useful than classical ones may cause difficulties hard to overcome.

On the experimental side the main current challenge actually is not to build a complete quantum computer right away, but rather to move from the experiments that exhibit quantum interference and entanglement to the experiments that can control these phenomena.

Currently it seems to be (quite) unfeasible to build a quantum computer sufficiently large and reliable to implement Shor's algorithm to factorize 200-digit integers—what would really be an achievement of great practical importance. However, even much more modest quantum processors would allow us to investigate problems of major scientific interest. It is expected that a 10-qubit processor would allow us to perform teleportation and a 100-qubit quantum processor could be of great importance for cryptography and would allow us to study problems of entanglement at very remote locations.

There are several good reasons why it is important:

1. to get better estimations whether and when and how powerful quantum computers could be available;

2. to have powerful quantum computers;

and therefore to invest a lot into quantum information processing research with the aim of answering these questions.

One important reason is that even a possibility of having a powerful quantum computer only in 20–40 years is of importance and danger already for current cryptography and information-processing security.

7.6.1 Main approaches

The first natural idea when the vision of designing quantum processors came up was, likely, to develop a natural successor of the current solid-state technology and to try to build quantum processors on the solid-state microchips. However, in the solid state devices the coupling with the environment seemed to be hopelessly strong. That is why the first experiments tried to use different technologies even if, as discussed later, a general feeling seems to converge to the understanding that the original idea has to get through if powerful quantum computers are ever built.

There have been a variety of early proposals and experimental attempts to design a quantum processor; for example Lloyd (1993). Some of them never got too far from the paper form; others brought at least some experimental experience. One interesting idea was that of heteropolymer based molecular computer developed in Stuttgart (see Teich, 1988) and at MIT (see Lloyd, 1994). For a detailed description see also Williams and Clearwater (1998).

Three the most interesting/promising current approaches seem to be to use quantum states of the laser-cooled ions in an magnetic trap (Cirac and Zoller, 1995, Monroe, 1995), cavity QED (Turchette et al. 1995), and, especially, nuclear magnetic resonance (Gershenfeld and Chuang, 1997, Cory et al. 1996).[13] These proposals are based on the-state-of-the-art experimental technologies developed already in quantum optics, atomic physics and magnetic resonance, and this makes them promising.

There are several sets of tasks or problems/difficulties that need to be overcome in order to be able to massage a technology to design a quantum processor.

1. Control. The available quantum states must be precisely identified. In addition, techniques have to be available for restricting states of the created quantum systems to the corresponding subspace of the Hilbert space. There must be a way to create quantum registers with qubits adequately isolated from interactions with the environment for the duration of computation.

[13] Of course a variety of other approaches are in different states of their "justifications" and "verifications".

2. Storage. Techniques have to be developed for storing quantum bits reliably for time needed to perform interesting computations.

3. Initial state. It has to be possible, in a reasonably simple way, to set the state vector into a given simple (initial) state; for example to the state $|0^{(n)}\rangle$. (This can sometimes be done by cooling the system into its ground state.)

4. Getting decoherence under a control. This requires to engineering quantum systems in which qubits effect each other, but not the environment. The coupling with the environment has to be sufficiently low in order not to destroy quantum interference and not to create undesirable entanglement.

5. Implementation of an adequate set of quantum gates. Actually, as already discussed in Section 2.3, it is sufficient to have good ways to realize all one-qubit gates and some control two-qubit gates. Quantum gates need to be implemented with sufficient precision. Rotation gates on single qubits are a more easy subproblem. To construct a controlllable "source of entanglement" a mechanism is needed by which two qubits can get entangled and this is usually a difficult task. (Practically, if one can implement with a technology the XOR gate, then "the door is open", at least in principle, to build any quantum circuit.)

6. Communications. The "wiring" or interconnection problem is far from trivial. On the contrary, in general it represents one of the most involved and uncertain features of quantum computing, that requires the development of new techniques not envisaged by former experimentalists. The key issue is to create controlled interactions of qubits.

   For an illustration, let us consider the network from Figure 1.14. When the first XOR gate is performed, there should be an interaction between qubits B and C and coupling between A and C should be zero. On the other hand, when the second XOR gate is performed, there should be a coupling between qubits a and c and zero coupling between a and b and b and c.

7. Measurement. The outcome of a quantum computation should be the result of a sequence of quantum measurements. It is essential that measurements can be made on specific subsystems (qubits) of the computational states; especially on each qubit separately.

7.6.2 Ion trap

The idea is due to Cirac and Zoller (1995), from the Innsbruck group, and was soon picked up by other groups: at NIST in Boulder, at the National Laboratory at Los Alamos, at IBM in Almaden and at the Max Planck Institute in Garching—at least. For a summarization of the efforts of the Boulder and Los Alamos groups and for their estimations of what seems to be achievable see Wineland et al. (1998, 1998a) and Hughes et al. (1998). The first demonstration of an ion trap XOR gate was due to Monroe et al. (1995). We discuss here mainly the Cirac–Zoller approach, in which a sequence of called ions is captured by a combination of static and electric fields in a so-called "Pauli trap" in a high vacuum. A single laser beam is split by beam splitters and acousto-optic modulators into many beam-pairs, one pair illuminating each ion. (The Boulder group used beryllium ions and the trap from radio-frequency waves.)

Problems to deal with:

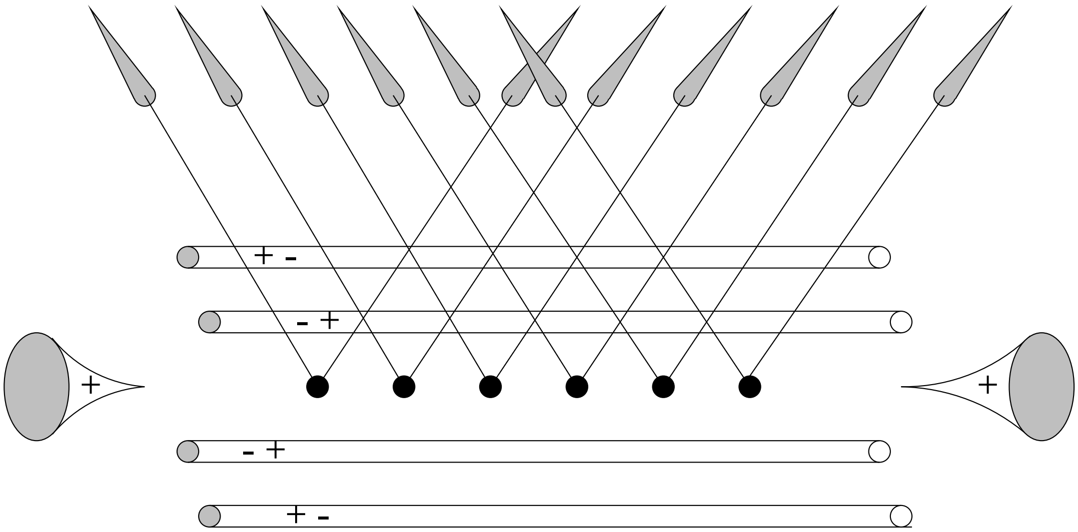

Figure 7.13: **Ion trap procesor**. Electrodes generate a time dependent electric field which generates an effective potential such that a string of ions is trapped—"stored in a linear trap". Each ion stores one qubit and it is addressed by a pair of laser beams. One-qubit operations are performed by shining precissely timed laser pulses on individual ions. The motional degree of freedom serves as a single "qubit bus" to transport quantum information among the ions. Two-qubit operations are performed by a sequence of laser pulses on the two particular ions. Initial state preparation is by optical pumping or laser cooling. Measurement is by electron shelving and resonance fluorerescence. This enables to measure the state of each ion with high signal to noise ratio (adapted from Steane, 1997, and Bennett, 1998).

Qubits. Each ion stores one qubit in its internal energy states. The quantum state of each ion is a superposition of the ground state (interpreted as $|0\rangle$) and the excited state (interpreted as $|1\rangle$).

Isolation. The ions are well isolated and spontaneous decay is the main source of decoherence. Other source of decoherence is the heat produced in the coupling between the charged ion strings and the noise voltage of electrodes.

Initial state. can be set up through optical pumping and laser cooling. This is a nontrivial technological problem because a very low temperature is needed.

One-qubit gates. Ions are sufficiently well separated and therefore they can be individually addressed by a pulsed laser. By shining precisely timed laser pulses and choosing the phase of the laser appropriately, any one-qubit unitary transformation can be performed.

Interaction of qubits. It is provided through the so-called joint vibrational mode. It is not easy to achieve and this is one of the major drawbacks of the ion-trap technology because this seems to prevent design of larger quantum registers. Two-qubit operations are performed by using a laser on one qubit to provide an impulse that ripples through a sequence of ions to the second qubit where another laser pulse stops the rippling and performs the operation. This way a single qubit "bus" is created to transport quantum information among the ions. Cirac and Zoller showed that the XOR gate can be implemented with ion-trap technology using altogether 5 pulses. In order to implement XOR-gate they encoded both qubits within a single beryllium ion.

Measurement., Readout is by electron shelving and resonance fluorescene. This allows the state of each ion to be measured with high reliability.

Speed. This seems to be a serious drawback of the ion-trap computers.

Performance conditions. They are very demanding. Extreme vacuum and extremely low temperature is needed.

The ion-trap approach does not seem to be able to deal easily with registers with a large number of qubits because of the problems with the interactions between ions. It has been proved difficult to progress beyond the one-qubit level. Estimates (guesses) for potentials of this technology much differ—from 10 or 12 till 47 qubits.

7.6.3 Cavity QED

An alternative technology to the ion-trap approach for the design of quantum processors is cavity QED (quantum electrodynamics) technology developed first at the Ecole Normale Supérieure in Paris (microwave cavity, by Horache's group, see Domokos et al. 1995) and then at Caltech (optical cavity, by Kimble's group, see Turchette et al., 1995).

The basic idea of the Caltech group approach is to trap several neutral atoms inside a high-fineness optical cavity. Quantum information is stored in the internal states of atoms or in the polarization of photons. In both cases it is reasonably easy to implement interactions of qubits via cavity QED.

To implement the XOR- gate (for a more detail description see, for example, Williams and Clearwater, 1998), both control and target qubits are implemented by beams of photons of different frequency and polarization (linearly polarized target qubit and circularly polarized control qubit), moving through an optical cavity (created by two mirrors). The target qubit is implemented by a beam of photons all of the same linear polarization. By lowering the intensity of the beam one can achieve that at any time there is at most one photon of the beam in the cavity. The control bit is implemented by a beam of photons all of either the right polarization or the left polarization. Again, by lowering the intensity of the beam one can achieve that at any time there is only one of those photons in the cavity.

To achieve an interaction of photons in the cavity, it is possible to make use of the fact that a linearly polarized photon can be seen as being in an equal superposition of the left and the right circularly polarized states. In order to help to make an interaction, a caesium atom drifts through the cavity. The distance between the mirrors of the cavity can be adjusted to a transition between two energy levels of the caesium atom and both photons.

When the target photon interacts with the caesium atom, its left-circular component is phase shifted by an amount that depends on the excited state of the caesium atom. (A linearly polarized target photon can be seen as being in an equal weight superposition of a left and right polarized state.) This in turn depends on which circular polarization control photon is. This way a conditional change of the state of the target qubit is achieved, depending on the state of the control photon. This rotation of the target qubit polarization is then measured.

7.6.4 Nuclear magnetic resonance (NMR)

A different approach is to adopt for quantum computing the nuclear magnetic resonance spectroscopy technology, that has already been used for years successfully in chemical analysis and for medical images.

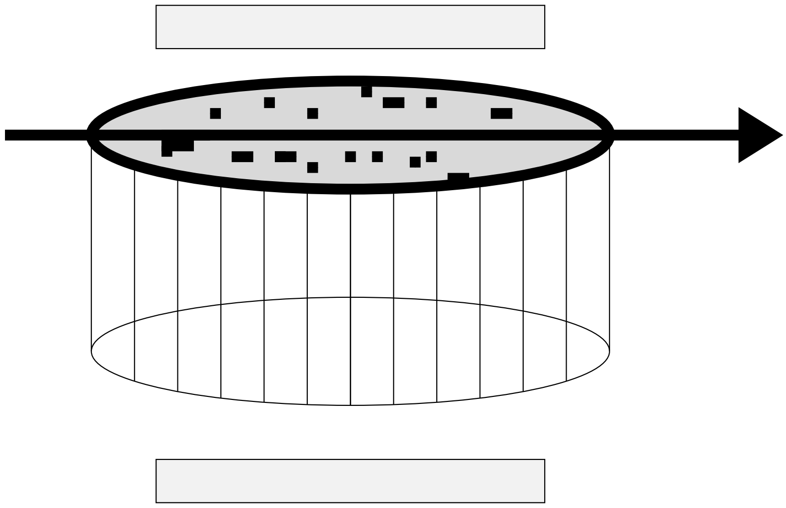

Figure 7.14: **NMR processor**. A liquid of $\approx 10^{20} - 10^{23}$ molecules ("subprocessors") is placed in a sensitive magnetometer which can generate oscillating magnetic fields and also detect the mean magnetic moment of the liquid. Each molecule is an independent processor. Qubits are stored and manipulated in the nuclear spin energy levels states. The spin energy levels of a given nucleus are influenced by neighbouring nuclei in the molecule. This allows XOR gate to be implemented. The initial state is one of thermal equilibrium. Only the average final state can be determined/ Experimental methods exist that can be used to extract out the final state—adopted from Steane (1997).

Research, development and even applications of the nuclear magnetic resonance technology already have long tradition and various experiments, in laboratories and hospitals, have already for many years routinely achieved spin-state manipulations and measurements equivalent in complexity to those required for quantum computing on a few qubits. In addition, this technology is known for relatively high decoherence time and capability to work at room temperature. It is therefore no wonder that with this technology it has proved relatively simple to design NMR quantum systems with 2 or 3 qubits, and it has been natural that the first two-qubits quantum algorithm implementations used NMR technology—as discussed in the example bellow.

The basic idea is that a quantum register is in this case a molecule containing a "backbone" of about ten atoms (with other atoms allowed so as to be able to create needed chemical bounds). Each qubit is realized in the spin orientation of an individual atomic nucleus—the direction of the nuclear magnetic dipole—in the atoms of the molecule. Each dipole can either reinforce or oppose an externally applied magnetic field. The first stage has lower energy than the second. The state can be changed by the absorption or emission of photons of right energy. The molecule is placed in a large magnetic field and the spin states of the nuclei are manipulated by applying oscillating magnetic fields in pulses of controlled duration.

The basic difficulty is that the spin state of the nuclei of a single molecule can neither be prepared nor measured. To overcome the problem not a single molecule but a cup of liquid containing about $10^{20} - 10^{23}$ identical molecules is used. The number of qubits of such an NMR computer is therefore equal to the number of backbone atoms per molecule. This implies that there is an enormous redundancy.

One approach is to encode a qubit in the average spin state of a large number of nuclei. The spin states are then manipulated by magnetic fields and the (average) spin state can be measured with the NMR spectroscopy technology.

Let us now deal in more detail with the problem of addressing and manipulating particular qubits. Using the so-called chemical shifts and spin–spin coupling one can identify the nuclear structure of the molecule of the sample. Single-qubit rotations are easy to implement. To them correspond rotations within a subspace corresponding to a single spin. Such rotations can be achieved using proper radio-frequency pulses. By using molecules with 2–3 magnetically active nuclei one can implement either XOR or Toffoli/Fredkin transformation. In both cases, first the current value of qubits is determined and then the corresponding sequence of radio-frequency pulses is applied.

The other major problem is how to extract results from such "massively parallel computers". The way out is to measure the average spin state. This seems to be a fundamental obstacle because quantum algorithms are probabilistic and averaging on an ensemble of molecules is not equivalent to the computation on a single device. The main new contribution of Gershenfeld et al. (1996) and Cory et al. (1998) was that they found a way how "effective pure state" could be prepared, manipulated and measured by performing suitable operations on the ensemble of molecules. One way to read out quantum information is to excite the spin system and to observe the resulting NMR spectrum. Different qubits correspond to different spins and give rise to signals at different resonance frequencies.

The NMR technology does not seem to scale up for several reasons: (1) an approach to distinguish qubits in a molecule by their chemical identities seems impossible for large molecules; (2) technology requires a too large redundancy; (3) it is hard to establish with sufficient precision an initial state. It is currently expected that with this technology one can hardly perform quantum computation with more than 12 qubits—unless radically new ideas appear.

The NMR technology was used to make the first three implementations of quantum algorithms: for the Deutsch problem discussed in Section 3.1 and for the Grover algorithm, see Section 3.3. In order to give a bit of insight into such an ventures, and a flavour of the expertise needed, let us provide a few basic details about the implementations of algorithms for Deutsch's problem—of slightly different algorithms from those presented in Section 3.1.

Example 7.6.1 *Jones and Mosca (1998) demonstrated that* ^{1}H *nuclei in partially deuterated cytosine can be used to implement a two-bit NMR quantum computer based on two coupled spins. They used a 50 mM solution of the pyrimidine base cytosine in* D_2O. *Rapid exchange of the two amine protons and the single amide proton with the deuterated solvent leaves two remaining protons forming an isolated two-spin system. All experiments used a home-made spectrometer with the* ^{1}H *operating frequency of 500 MHz. The first pair of Hadamard gates was replaced by so-called* 90°_y *pulses and the second pair by* 90°_{-y} *pulses. Also* U_f *transformation was implemented by a series of pulses. The final outcome was obtained by applying a* 90°_y *pulse and by observing the spectrum.*

Chuang et al. (1998) prepared the input state with a 200 mM, 0.5 ml sample of chloroform dissolved in d6-acetone, at room temperature and standard pressure. The $\mathcal{O}(10^{16})$

molecules in the bulk sample can be considered as independent quantum computers, all functioning simultaneously. Pulsed-radio frequence electromagnetic fields were applied to perform Hadamard rotations. The unitary mapping U_f was implemented using pulsed radio frequency and spin–spin interaction. The outcomes were read out by applying a special read-out pulses that transformed spin values into a voltage.

The first proposals for NMR implementations suffered from a signal-to-noise ratio that decayed exponentially with the number of qubits. An NMR implementation in which the signal-to-noise ratio depends only on features of NMR technology and not on the number of qubits was suggested by Schulman and Vazirani (1998). They gave also a new technology for preparing the initial state. In addition, they developed an abstract model of an NMR computer and proved several results.

Remark 7.6.2 For an introductory description of ion trapped and NMR technologies and implementations of NOT and XOR gates as well as basic gates needed to implement QFT (see Section 3.3), see Berman et al. (1998).

7.6.5 Other potential technologies

From other suggested technologies to build quantum gates and processors let us mention three approaches that are considered as promising. All of them have in common that they are solid-state technologies, and one could expect their scalability because the technology for making miniaturization of solid-state devices already exists. Decoherence problems are disadvantage.

1. Quantum-dot[14] ("artificial atoms") technologies. Quantum-dots are regions in a crystal that have artificial inhomogeneity and can be placed in a controlled manner. They can be used to trap single electrons and one can use the orientation of their spin to store information. For details see DiVincenzo (1995), Burkard, Loss and DiVincenzo (1998) and Barenco, Deutsch, Ekert and Jozsa (1995).
2. Super-conducting quantum interference devices (SQUIDs). In these systems the quantization of flux in a superconducting circuit is used to make a qubit.
3. A combination of the NMR technology with semiconductor physics (see Kane, 1998). The computer he suggest is made of the same components as the current semiconductor technology computers. The qubit is again a nucleus spin.

Moral: John von Neumann was one of the founders of modern quantum mechanics and also of modern computing; an extraordinary scientists who had great vision. We shall never know all his thoughts, but who can nowadays imagine von Neumann standing before ENIAC—weighting 60 tons and 30 m long—and envisioning a merger of quantum mechanics and computing? A good rule of thumb in searching for frontiers in computing is, as in life, never to underestimate the power of Nature, to keep learning from it as much as you can, to remember that an excellent theory is one of the most profitable products of mankind and that very fruitful results often come from combining two seemingly unrelated ideas into one.

---

[14] Quantum-dot technology is of importance also for classical computing in its attempts to develop single-electron transistors. It is already possible, even at room temperature, to transfer a single electron from a reservoir into a quantum dot in such a way that once inside, it blocks transfer of other electrons. The current through such a transistor depends on the number of electrons in the quantum dot. This allows us to "write" and "erase" information—see Berman et al. (1998).