# Chapter 14
# Quantum Error Correction

"*I wish to God these calculations had been executed by steam*!"
– Charles Babbage[1]

The descriptions of quantum algorithms and quantum information processing protocols given in the foregoing chapters all assume a correct design, precise implementation, and perfect operation of our quantum computing device. But real quantum hardware, and real quantum computations run on it, are unlikely to be manufactured exactly to their specifications, and unlikely to perform flawlessly. Components of real quantum computers can only be manufactured and assembled to within some finite tolerances. Pulses can only be shaped and timed to within certain limits. Voltages, currents, fluxes, and inter-qubit couplings cannot be turned on and off instantaneously, etc. Moreover, the fundamental paradox of quantum computation is that at one instant we desire our qubits to be isolated perfectly from their environment, but at another, we want them to interact strongly with some "external" measuring apparatus. Turning such environmental interactions wholly on and off at will is challenging. Thus, real quantum computers will be beset with errors causing their computations to go awry.

A similar situation holds in classical computing, of course. But in that case we can identify and correct bit-errors using various classical error-correction techniques. We are helped profoundly in this regard by the ability classical physics gives us to look at the instantaneous state of a classical computation, assess its correctness, and then make the necessary adjustments. In the quantum realm we do not have this luxury, because we cannot read the state of a quantum memory register in the midst of a quantum computation without necessarily, and irreversibly, perturbing the future course of the computation. Thus it is not at all obvious, a priori, whether the techniques developed for correcting errors in classical computers are useful for correcting errors in quantum computers. In fact, shortly after Shor's algorithm was first published several highly respected physicists expressed skepticism about the feasibly of an error-correction method for quantum computers [227, 300, 301, 502].

[1] Source: Computer History Museum, http://www.computerhistory.org/babbage/history/.

C.P. Williams, *Explorations in Quantum Computing,*
Texts in Computer Science,
DOI 10.1007/978-1-84628-887-6_14, © Springer-Verlag London Limited 2011

## 14.1 How Errors Arise in Quantum Computing

In the idealized models of quantum computers that we studied in Chaps. 1–3, the qubits representing the computational state of the computer were assumed to be perfectly isolated from their environment. In other words, from the moment the quantum computer is prepared in some initial state to start the computation off, to the moment it is measured to extract an answer, the logical qubits of ideal quantum computers are supposed to evolve unitarily in accordance with Schrödinger's equation. Unfortunately, such an idealization is unattainable. Any real quantum system couples to its environment over time. In the process, information leaks out of the logical state of the qubits in the quantum memory register. If you did not model the effect of the environment explicitly, it would appear as if the logical qubits were no longer evolving unitarily in accordance with Schrödinger's equation. Indeed, this coupling between a quantum system and its environment, and the resulting loss of coherence, is what prevents quantum effects from being evident at the macroscopic level [202].

Even with our best efforts keeping a quantum memory register isolated from its environment is difficult. For one thing, a quantum memory register has to be built out of *something* so there must be some supporting infrastructure, or "scaffolding," in the vicinity of the computationally active qubits. There is, therefore, a chance that the particles within the scaffolding will couple to the computational elements. In addition, there can be a coupling between the memory register and an ambient thermal heat bath. Also, incoming stray particles such as cosmic rays or gas molecules can interact with the memory register. In fact, there are many physical processes that can perturb the state of a quantum memory register. Broadly speaking, these physical processes fall under the headings of dissipation and decoherence.

### *14.1.1 Dissipation-Induced Bit Flip Errors*

*Dissipation* is a process by which a qubit loses energy to its environment. Thus, for example, if an excited state is used to represent a $|1\rangle$ and a lower energy state is used to represent a $|0\rangle$, a qubit might transition, spontaneously, from the $|1\rangle$ state to the $|0\rangle$ state emitting a photon in the process. In computational terms, a bit in the quantum memory register would have "flipped" spontaneously.

Dissipation causes a bit to flip and this operation is described by the action of the Pauli $X$ matrix. We assume that the qubit starts off in an arbitrary superposed state given by

$$|\psi\rangle = a|0\rangle + b|1\rangle \tag{14.1}$$

such that $|a|^2 + |b|^2 = 1$.

The affect of $\sigma_x$ on the state of a qubit is:

$$\sigma_x(a|0\rangle + b|1\rangle) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} b \\ a \end{pmatrix} = b|0\rangle + a|1\rangle \tag{14.2}$$

This time the action of the operator has caused the bits to flip. That is, $\sigma_x$ causes the transformation $a|0\rangle + b|1\rangle \rightarrow b|0\rangle + a|1\rangle$. So let us call such an operation a "bit flip error."

### 14.1.2 Decoherence-Induced Phase Shift Errors

*Decoherence*, on the other hand, is more insidious. Rather than an overt bit flip, stray interactions between the qubits and the environment cause the quantum memory register and the environment to become entangled with one another. As a result, the initially pure state of our ideal quantum memory register becomes progressively more mixed over time. This mixing alters the relative phases of the computational basis eigenstates of the memory register. As a result, the interference effects, needed in any true quantum computation, become distorted and the quantum computation no longer proceeds as it should.

An overly simplified, but intuitive, model for the impact of such decoherence on a quantum memory register is as follows. Suppose that initially, i.e., at a time $t = 0$, a single qubit in a quantum memory register starts out in the pure state $|\psi\rangle = a|0\rangle + b|1\rangle$. As a density matrix such a state may also be written as:

$$\rho(0) = |\psi\rangle\langle\psi| = \begin{pmatrix} a \\ b \end{pmatrix} \cdot \begin{pmatrix} a^* & b^* \end{pmatrix} = \begin{pmatrix} |a|^2 & ab^* \\ a^*b & |b|^2 \end{pmatrix} \tag{14.3}$$

where the asterisk denotes taking the complex conjugate. After merely "storing" such a qubit in a realistic, i.e., weakly noisy, environment for a time $t$, its density matrix will become:

$$\rho(t) = \begin{pmatrix} |a|^2 & e^{-t/\tau}ab^* \\ e^{-t/\tau}a^*b & |b|^2 \end{pmatrix} \tag{14.4}$$

where $\tau$, called the "decoherence time," sets the characteristic time-scale of the decoherence process, i.e., the time it takes for the off-diagonal elements of $\rho$ to decay appreciably. In the long time limit, $\tau \rightarrow \infty$, the density matrix becomes a mixture of the two possible measurement outcomes for this qubit, namely:

$$\rho(\infty) = \begin{pmatrix} |a|^2 & 0 \\ 0 & |b|^2 \end{pmatrix} \tag{14.5}$$

It is as if the environment has "measured" the qubit.

Similarly, applying $\sigma_z$ to the qubit results in the following transformation:

$$\sigma_z(a|0\rangle + b|1\rangle) = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} a \\ -b \end{pmatrix} = a|0\rangle - b|1\rangle \tag{14.6}$$

That is, $\sigma_z$ causes the "correct" state to evolve according to the rule $a|0\rangle + b|1\rangle \rightarrow a|0\rangle - b|1\rangle$, which has changed the *phase* of the qubit. Consequently, we call such an operation a "phase shift error."

The action of the identity matrix on a state is to leave the state unchanged. So that must represent the "no error" possibility.

That only leaves us to consider what happens when we apply $\sigma_y$ to the state of the qubit:

$$\sigma_y(a|0\rangle + b|1\rangle) = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} -ib \\ ia \end{pmatrix} = -ib|0\rangle + ia|1\rangle \tag{14.7}$$

This operation corresponds to *both* a phase shift and a bit flip. That is, $\sigma_y$ causes the transformation $a|0\rangle + b|1\rangle \rightarrow -ib|0\rangle + ia|1\rangle$. Thus, any error in a single qubit can be described by the action of a linear combination of the operators $\sigma_x$, $\sigma_y$, $\sigma_z$, and $\mathbb{1}$ (the identity operator).

### *14.1.3 Natural Decoherence Times of Physical Systems*

Usually, decoherence occurs on a faster timescale than dissipation. The time it takes a memory register to decohere depends, principally, upon what kind of quantum systems it is made from, the size of the register, the temperature of the thermal environment, and the rate of collisions with ambient gas molecules.

A crude estimate of decoherence times in various settings can be obtained from the Heisenberg Uncertainty Principle, in energy and time,

$$\Delta t \approx \frac{\hbar}{\Delta E} = \frac{\hbar}{k_B T} \tag{14.8}$$

where $k_B$ is Boltzmann's constant (approximately $1.38 \times 10^{-23}$ Joules $\mathrm{K}^{-1}$) and $T$ is the absolute temperature of the environment. In this estimate we have taken the uncertainty in the energy to be of the order of the energy of a typical particle at the ambient temperature. At room temperature, this gives a typical decoherence time of about $10^{-14}$ seconds. At lower temperatures, systems take longer to decohere. For example, at the temperature of liquid helium, it takes about 100 times as long for a system to decohere as it does at room temperature. Consequently, the simplest way to try to combat decoherence is to operate the computer at a lower temperature. Table 14.1 summarizes some characteristic decoherence times, under various physical scenarios. These estimates were derived using a more sophisticated analysis [257].

Once we have chilled our quantum computer and sealed it in as good a vacuum as we can, what else can we do to slow down decoherence? Well, we could try building the quantum memory register out of different types of quantum systems. Certain quantum systems are much more resilient to decoherence than others. David DiVincenzo has collected statistics on the intrinsic decoherence properties of various materials [146]. The data are shown in Table 14.2. They reveal that trapped ions, for example, can potentially sustain a large number of computational steps before the succumb to decoherence. Step counts reported in Table 14.2 suggest that it might

**Table 14.1** Approximate decoherence times (in seconds) for various sized systems in different thermal and gaseous environments [257]

| System size (cm) | Cosmic background radiation | Room temp. (300 K) | Sunlight | Vacuum | Air |
|---|---|---|---|---|---|
| $10^{-3}$ | $10^{-7}$ | $10^{-14}$ | $10^{-16}$ | $10^{-18}$ | $10^{-35}$ |
| $10^{-5}$ | $10^{15}$ | $10^{-3}$ | $10^{-8}$ | $10^{-10}$ | $10^{-23}$ |
| $10^{-6}$ | $10^{24}$ | $10^{5}$ | $10^{-2}$ | $10^{-6}$ | $10^{-19}$ |

**Table 14.2** The maximum number of computational steps that can be accomplished without losing coherence for various quantum systems

| Quantum system | Time per gate operation | Coherence time | Max. no. of coherent steps |
|---|---|---|---|
| Mössbauer nucleus | $10^{-19}$ | $10^{-10}$ | $10^{9}$ |
| GaAs electrons | $10^{-13}$ | $10^{-10}$ | $10^{3}$ |
| Gold 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 | $10^{-3}$ | $10^{4}$ | $10^{7}$ |

be possible to build a quantum memory register that can support a significant number of computational steps. Nevertheless, decoherence looks like it will preclude quantum computation beyond a certain number of steps. This poses a severe problem for anyone wanting to build a universal quantum computer. Ideally, we would like a quantum computer that could, in principle, maintain coherent quantum computations indefinitely. Thus, if can not prevent decoherence, we need to think about ways of undoing its affects. Thus there needs to be a way of doing quantum error correction. But to understand how to correct errors, we need to understand how errors will perturb the quantum states we wish to protect. So what we need next is a mathematical model of the effects of errors on quantum computations.

### *14.1.4 What Makes Quantum Error Correction so Hard?*

With classical computers, it is possible to measure the state of the physical system used to encode a bit without disrupting the bit. Thus, if a voltage were used to represent a classical bit, you could, in principle, detect a slight drop from its nominal value and then give the voltage a nudge to restore it to its correct level.

Secondly, once a full bit-error has occurred, the nature of the error is far more limited in the classical domain than the quantum domain. In particular, the principal types of errors that can afflict a classical bit are either a bit *flip*, i.e., $0 \rightarrow 1$ or

$1 \rightarrow 0$, or, especially in the case of communication channels, the *loss* of a bit or the *insertion* of spurious bit. These types of errors are discrete and flagrant. There is no subtle "drift" in a bit value—it is either correct or flipped, and present or absent. Contrast this with the kinds of errors that can afflict qubits.

Qubits, however, do not have to be in states that are wholly $|0\rangle$ or wholly $|1\rangle$, but can be in superpositions of $|0\rangle$ *and* $|1\rangle$, e.g., $\alpha|0\rangle + \beta|1\rangle$, where the values of the amplitudes span a continuum of values. Thus, qubit states can "drift" off their intended values rather than suffer only gross errors (as do classical bits). This makes the errors that can afflict a qubit potentially far more subtle and insidious that the errors that can afflict a classical bit. Thus, in addition to a qubit bit flip $\alpha|0\rangle + \beta|1\rangle \rightarrow \alpha|1\rangle + \beta|0\rangle$, it is also possible to have qubit phase shifts $\alpha|0\rangle + \beta|1\rangle \rightarrow \alpha|0\rangle + \beta e^{i\phi}|1\rangle$ in which, even though the amplitudes remain the same magnitudes, errors can creep into the relative phase between the $|0\rangle$ and $|1\rangle$ components causing error states such as $\alpha|0\rangle + \beta e^{i\phi}|1\rangle$. Such corruption of the relative phase between the $|0\rangle$ and $|1\rangle$ components can mess up subsequent interference effects that all quantum algorithms rely upon. This particular, failure mode does not exist in the case of a classical computer.

Thirdly, in classical computing, we can make copies of bits we want to protect, replicate computations done on them, and use majority votes of the results to help eliminate errors. This ability to have redundant information is a great asset in error-correcting classical information. In quantum computing, however, the quantum no-cloning theorem precludes the possibility of copying an unknown quantum state. This makes it much more difficult to see how one could exploit redundancy in quantum computations for error correction purposes.

The aforementioned differences between classical and quantum information from the perspective of its intrinsic ability to be error-corrected are summarized in Table 14.3.

**Table 14.3** Intrinsic differences between classical information and quantum information that make quantum error correction more difficult than classical error correction

| Feature | Classical | Quantum |
|---|---|---|
| Information | Discrete encoding (0 or 1) | Continuous encoding $(\alpha\|0\rangle + \beta\|1\rangle)$ |
| Bit Errors | $0 \rightleftarrows 1$ | $\alpha\|0\rangle + \beta\|1\rangle \rightarrow \alpha\|1\rangle + \beta\|0\rangle$ |
| Phase Errors | Phase errors cannot occur for classical bits | $\alpha\|0\rangle + \beta\|1\rangle \rightarrow \alpha\|0\rangle + \beta e^{i\phi}\|1\rangle$ |
| Compound Errors | Compound bit and phase errors cannot occur for classical bits | $\alpha\|0\rangle + \beta\|1\rangle \rightarrow \alpha\|1\rangle + \beta e^{i\phi}\|0\rangle$ |
| Redundancy | Can be used | Cannot be used once the quantum computation is underway because the no-cloning theorem precludes copying an unknown quantum state |
| Monitoring | Can read memory register during computation to ascertain nature of error | Cannot read memory register during computation to ascertain nature of error |

Fortunately, we now know that there is a solution to our dilemma. The answer lies in *quantum* error correction. The trick, as John Preskill of Caltech likes to say, is "to use entanglement to fight entanglement". That is, by creating a specially designed entanglement between a quantum state we want to protect and that of other qubits, we can recognize when our protected state has gone bad and fix it, without damaging the delicate quantum correlations within our protected state. In this chapter we will review some approaches to quantum error correction and explain how entanglement is both the problem, and solution.

## 14.2 Quantum Error Reduction by Symmetrization

Classical computers can be made more reliable through the use of redundancy. Instead of a single computer being used to perform a given computation, several computers are used to perform the same computation simultaneously. If the computers are all running the same deterministic algorithm, they should all produce identical results at each stage of the computation. However, if an error occurs in one of the computers, its computational state will begin to diverge from that of the others. If you periodically poll all the computers and reset their computational states to the majority opinion, you will typically be able to correct errors that arose in a few of the computers since the last poll was taken. This type of majority voting scheme is currently used in the Space Shuttle to improve the reliability of the on-board decision making.

For majority voting to be effective, however, a number of assumptions must hold. First, the individual chances of any one computer obtaining the "correct" result must be greater than 50%. If this were not true then the majority opinion is more likely to be wrong than it is to be right. Secondly, the replicated computations must be independent of one another so that the errors incurred by the different computers are uncorrelated. This can be difficult to achieve in practice if all the computers use the same type of hardware and run the same program. Finally, replicating the computation an odd number of times (i.e., $2N-1$) guarantees a majority opinion always exists. The more replicated computations you use, the better your chances of fixing potential errors. In fact, if there are $2N-1$ computers (for $N = 1, 2, 3, \ldots$) and the individual probability of each computer obtaining the correct answer is $p > 0.5$ then the probability that the majority opinion is correct is given by:

$$\Pr(\text{Majority Correct}) = \sum_{i=N}^{2N-1} \binom{2N-1}{i} p^i (1-p)^{2N-1-i} \tag{14.9}$$

Although unsophisticated, this scheme is actually used today on the Space Shuttle and Boeing 777.

Figure 14.1 shows how the probability of the majority vote being correct increases as the probability of success of the individual computations increases for various numbers of replicated computations.

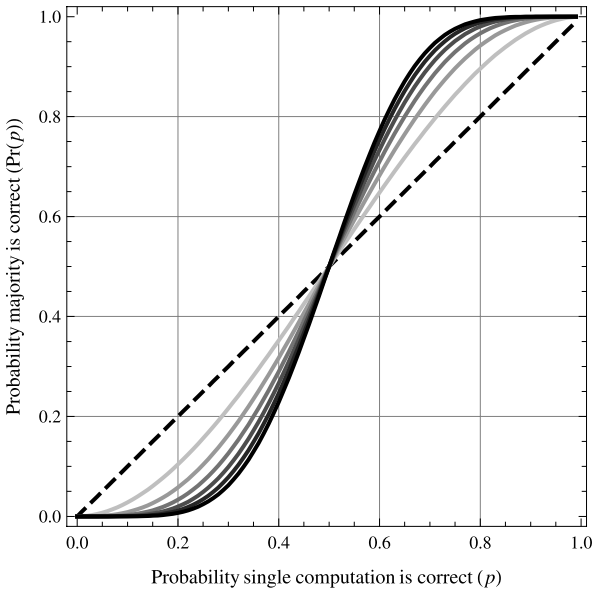


**Fig. 14.1** Probability that the majority vote is correct based on the probability of a single independent computation being correct. The *dashed line* is the case of a single computation without replication; the *lightest curve* is for the same computation repeated on three computers; and the *darkest curve* is for the same computation repeated on 13 computers. When the individual success probability exceeds 0.5 it pays to repeat computations and adopt the majority decision

Unfortunately, in quantum computation we cannot use such a majority voting scheme. This is because at the intermediate stages of typical quantum computations the quantum memory registers will be in superpositions of possible bit string configurations weighted non-uniformly by different probability amplitudes. If we were to readout the memory register during the course of the quantum computation we could project the state of the register into an eigenstate of the memory register thereby destroying the delicate superposition and in fact de-railing the quantum computation. So if we attempted to use naive majority voting within quantum computation, we would unfortunately destroy the computation.

### *14.2.1 The Symmetrization Trick*

There is, however, a more cunning way to use something akin to majority voting within quantum computation. This is called the method of error reduction via symmetrization [35]. The idea is that although we have no idea whatsoever what the instantaneous state of some quantum computation might be, we do know that if we had $R$ replicas of the same quantum computation, that the *joint* state of all $R$ quantum computations would be the tensor product of the individual quantum computations, i.e.,

$$|\Psi(t)\rangle_{\text{ideal}} = |\psi(t)\rangle \otimes |\psi(t)\rangle \otimes \cdots \otimes |\psi(t)\rangle \tag{14.10}$$

This is because, so long as no observations are made, the quantum evolution of an isolated quantum system is governed by Schrödinger's equation, which is a de-

terministic differential equation. Hence, if no errors afflicted any of the quantum computations then the joint state ought to have a tensor product structure.

In reality, however, each quantum computation might experience some error at random and uncorrelated from the errors afflicting the sister quantum computations. If this happens, the actual joint state of the $R$ quantum computations would be something like:

$$|\Psi(t)\rangle_{\text{actual}} = |\psi_1(t)\rangle \otimes |\psi_2(t)\rangle \otimes \cdots \otimes |\psi_R(t)\rangle \tag{14.11}$$

Quantum error correction by symmetrization works by intermittently projecting the joint state of the $R$ quantum computers into the symmetric subspace $\mathcal{SYM}$. The correct part of the quantum computation is always guaranteed to lie within $\mathcal{SYM}$, so by projecting the joint state into $\mathcal{SYM}$ we only knock out parts of the joint state that must be buggy, and thereby boost the proportion of the correct state within $\mathcal{SYM}$. Unfortunately, there are other symmetric states that can lie within $\mathcal{SYM}$ too which are not part of the true state. Nevertheless, provided we project into $\mathcal{SYM}$ often enough and provided we use enough replicas, $R$, of our computation these other types of errors can be suppressed to any desired level.

**Quantum Error Reduction via Symmetrization**

1. Initialize $R$ identical independent quantum computers to be in the same starting state running the same quantum algorithm. If there are no errors then, at any instant, the *joint* state of these $R$ quantum computers would be a state of the form $|\psi\rangle|\psi\rangle\cdots|\psi\rangle$, which is invariant under any permutation of the computers. However, due to independent small errors, the joint state will actually be of the form $|\psi_1\rangle|\psi_2\rangle\cdots|\psi_R\rangle$ where the individual component states (corresponding to the $R$ independent quantum computations) will be slightly different from one another.
2. To suppress the accumulate errors, initialize $O(\log_2 R!) \approx O(R \log_2 R)$ ancillae in state $|0\rangle$.
3. Place the ancillae in an equally weighted superposition of the integers (i.e., bit strings) in the range 0 to $R! - 1$, i.e., perform the transformation:

$$\mathcal{U}\,|0\rangle \rightarrow \frac{1}{\sqrt{R!}} \sum_{i=0}^{R!-1} |i\rangle \tag{14.12}$$

4. Apply the $i$-th permutation to the states of the $R$ individual quantum computers conditioned on the value $|i\rangle$ stored in the ancillae. That is, apply the conditional transformation:

$$|i\rangle|\psi_1\rangle|\psi_2\rangle\cdots|\psi_R\rangle \rightarrow |i\rangle|\psi_{\sigma_i}(1)\rangle|\psi_{\sigma_i}(2)\rangle\cdots|\psi_{\sigma_i}(R)\rangle \tag{14.13}$$

thereby creating the entangled state:

$$\sum_i |i\rangle|\psi_{\sigma_i(1)}\rangle|\psi_{\sigma_i(2)}\rangle\cdots|\psi_{\sigma_i(R)}\rangle \tag{14.14}$$

5. Apply the inverse computation $\mathcal{U}^{-1}$ to that applied in step 3 above. As the forward $\mathcal{U}$ operation mapped $|0\rangle$ into equally weighted superposition of the $R!$ possible integers (representing the possible indices of the permutations of $R$ objects), then the inverse operation does exactly the reverse. Thus the state we obtain can be written as:

$$\sum_i |i\rangle|E_i\rangle \tag{14.15}$$

in which the $|E_0\rangle$ component (i.e., the state of the rest of the register when the ancillae is in state $|0\rangle$ represents the desired (i.e., symmetrized) state, and the other components are error states.
6. Measure the ancillae qubits in the computational basis. If they are all found to be in state $|0\rangle$ then $|\Psi\rangle$ has been successfully projected into the symmetric subspace $\mathcal{SYM}$.

### *14.2.2 Quantum Circuit for Symmetrization*

Projection into the symmetric subspace can be accomplished using a quantum circuit like that shown in Fig. 14.2, which is specialized to the case of three replicated computations. The key insight is realize that you can build up permutations of quantum states *recursively*. Specifically, consider a set of $k+1$ elements $e_1, e_2, \ldots, e_{k+1}$. How can we construct all permutations of this set? Well suppose we already have a permutation, $e_{\sigma(1)}, e_{\sigma(2)}, \ldots, e_{\sigma(k)}$ of the first $k$ elements, $e_1, e_2, \ldots, e_k$ of the

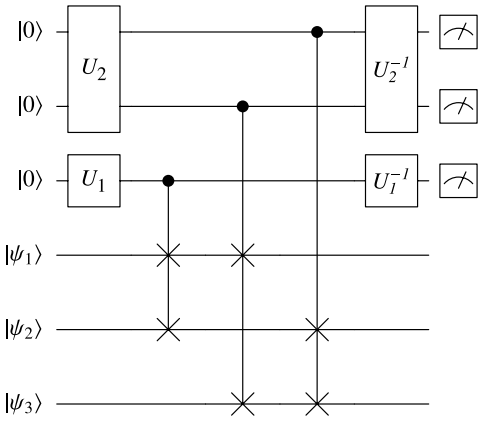


**Fig. 14.2** Quantum circuit for error correction via symmetrization. In this example, we symmetrize the state of three replicas of a 1-qubit quantum computation. Provided that, when measured, the ancillae are all found in state $|0\rangle$, the overlap between the joint correct state and the joint symmetrized state, $\langle\Psi_{\text{correct}}|U_{\mathcal{SYM}}|\Psi_{\text{buggy}}\rangle$, will be higher than the overlap between the joint correct state and the joint unsymmetrized state, $\langle\Psi_{\text{correct}}|\Psi_{\text{buggy}}\rangle$

set. We can then join $e_{k+1}$ to the end of this permutation, creating the permutation $e_{\sigma(1)}, e_{\sigma(2)}, \ldots, e_{\sigma(k)}, e_{k+1}$. The remaining permutations can be constructed by systematically swapping $e_{k+1}$ with each of the $e_{\sigma(i)}$ in turn. This suggests the structure of a quantum circuit sufficient to generate all possible permutations of $k+1$ quantum states. These represent $k+1$ independent realizations of some quantum computation.

In other words, once we have a symmetrized version of the state $|\psi_1\rangle|\psi_2\rangle\cdots|\psi_k\rangle$, we can easily symmetrize the state $|\psi_1\rangle|\psi_2\rangle\cdots|\psi_k\rangle|\psi_{k+1}\rangle$ by adjoining state $|\psi_{k+1}$ and applying a sequence of controlled SWAP operations. As you mat recall from Chap. 2, controlled-SWAP is synonymous with a FREDKIN gate.

$$
\begin{aligned}
U_k = \Bigg[ & \bigodot_{j=k-1}^{1} \mathbb{1}_{2^{j-1}} \otimes \frac{1}{\sqrt{k-j+1}} \\
& \times \begin{pmatrix} \sqrt{-j+k+1} & 0 & 0 & 0 \\ 0 & 1 & \sqrt{k-j} & 0 \\ 0 & -\sqrt{k-j} & 1 & 0 \\ 0 & 0 & 0 & \sqrt{-j+k+1} \end{pmatrix} \otimes \mathbb{1}_{2^{k-(j+1)}} \Bigg] \\
& \cdot \frac{1}{\sqrt{k+1}} \begin{pmatrix} 1 & -\sqrt{k} \\ \sqrt{k} & 1 \end{pmatrix} \otimes \mathbb{1}_{2^{k-1}}
\end{aligned} \tag{14.16}
$$

### *14.2.3 Example: Quantum Error Reduction via Symmetrization*

Suppose we have three replicas of the same quantum computation, such that the correct state should be:

$$|\Psi_{\text{correct}}\rangle = |\psi\rangle|\psi\rangle|\psi\rangle \tag{14.17}$$

where

$$|\psi\rangle = \frac{1}{2}|0\rangle - \frac{\sqrt{3}}{2}i|1\rangle \tag{14.18}$$

Imagine that the independent computations have each drifted slightly off the correct states $|\psi\rangle$ so that what we actually have is:

$$|\Psi_{\text{buggy}}\rangle = |\psi_1\rangle|\psi_2\rangle|\psi_3\rangle \tag{14.19}$$

where

$$
\begin{aligned}
|\psi_1\rangle &= \left\| \frac{1}{2}|0\rangle - \frac{\sqrt{3.5}}{2}i|1\rangle \right\| \\
|\psi_2\rangle &= \left\| \frac{1}{2.5}|0\rangle - \frac{\sqrt{3}}{2}i|1\rangle \right\| \\
|\psi_3\rangle &= \left\| \frac{1}{1.5}|0\rangle - \frac{\sqrt{2.5}}{2}i|1\rangle \right\|
\end{aligned} \tag{14.20}
$$

where the symbol $\| \ \|$ indicates the re-normalized version of the state. We expressed the perturbed states as shown to make it easier to see that they are only slight adrift of their ideal values.

Thus the overall correct state is:

$$\begin{aligned} |\Psi_{\text{correct}}\rangle &= |\psi\rangle|\psi\rangle|\psi\rangle \\ &\approx \begin{pmatrix} 0.125|000\rangle - 0.216506i|001\rangle - 0.216506i|010\rangle - 0.375|011\rangle - \\ 0.216506i|100\rangle - 0.375|101\rangle - 0.375|110\rangle + 0.649519i|111\rangle \end{pmatrix} \end{aligned} \tag{14.21}$$

whereas the actual (buggy) state we have is:

$$\begin{aligned} |\Psi_{\text{buggy}}\rangle &= |\psi_1\rangle|\psi_2\rangle|\psi_3\rangle \\ &\approx \begin{pmatrix} 0.127427|000\rangle - 0.15111i|001\rangle - 0.275888i|010\rangle - 0.327163|011\rangle \\ 0.238395i|100\rangle - 0.282701|101\rangle - 0.51614|110\rangle + 0.612066i|111\rangle \end{pmatrix} \end{aligned} \tag{14.22}$$

Hence, the overlap between the correct state and the buggy state is:

$$\langle \Psi_{\text{correct}} | \Psi_{\text{buggy}} \rangle \approx 0.979791 \tag{14.23}$$

Now let us see what happens when we attempt to re-symmetrize the buggy state. By how much does the error reduce? To construct the error symmetrization operator we need the following gates:

$$\begin{aligned} U_1 &= \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \\ U_2 &= \begin{pmatrix} \frac{1}{\sqrt{3}} & 0 & -\sqrt{\frac{2}{3}} & 0 \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{3}} \\ 0 & \sqrt{\frac{2}{3}} & 0 & \frac{1}{\sqrt{3}} \end{pmatrix} \end{aligned} \tag{14.24}$$

Then, the full error symmetrization operator is constructed from:

$$\begin{aligned} U_{\mathcal{SYM}_3} = (U_2^{-1} \otimes U_1^{-1} \otimes \mathbb{1}_8) \cdot \text{FREDKIN}_{1,5,6;6} \cdot \text{FREDKIN}_{2,4,6;6} \\ \cdot \text{FREDKIN}_{3,4,5;6} \cdot (U_2 \otimes U_1 \otimes \mathbb{1}_8) \end{aligned} \tag{14.25}$$

where the subscript 3 on $\mathcal{SYM}$ indicates the operator is specialized to symmetrizing a triple repetition of the quantum computation, and $\text{FREDKIN}_{i,j,k;\ell}$ means a Fredkin gate inserted in $\ell$ qubits with control on qubit $i$, and the SWAP it performs between qubits $j$ and $k$.

So now let us re-symmetrize the buggy state. That is we compute:

$$U_{\mathcal{SYM}_3}|\Psi_{\text{buggy}}\rangle = 0.129651|000\rangle - 0.232026\, i|001\rangle - 0.246828\, i|010\rangle$$

$$- 0.42901|011\rangle - 0.198151\, i|100\rangle - 0.342299|101\rangle$$
$$- 0.374345|110\rangle + 0.622747\, i|111\rangle \tag{14.26}$$

Hence, the overlap between the correct state and the buggy state after re-symmetrization is:

$$\langle \Psi_{\text{correct}} | U_{\mathcal{SYM}_3} | \Psi_{\text{buggy}} \rangle \approx 0.996889 \tag{14.27}$$

which is higher than it was before re-symmetrization. Hence, error correction by symmetrization has succeeded in reducing the error, even without knowing what the error was!

This idea of coupling two systems, so that measuring the state of one system projects the state of the other into a specific subspace, can be used to perform error correction. This technique is most appropriate for correcting several qubits that are slightly wrong rather than correcting a single qubit that is terribly wrong [388].

Quantum error reduction by symmetrization is most suited to correcting small independent errors (such as random phase drifts rather than bit flips) and is more successful the more frequently it is repeated. However, certain error processes, such as spontaneous emission, can result in sudden *large* errors, such as bit flips. These kinds of errors require a different error-correction strategy based on the idea of error-correcting codes.

## 14.3 Principles of Quantum Error Correcting Codes (QECCs)

Classical error correcting codes are used routinely to immunize classical computations and communications from errors such as accidental bit flips. The key idea is to use classical *codewords*, i.e., carefully chosen bit strings, to encode each logical bit we want to protect, in such a manner that a subsequent error, or perhaps multiple errors, in a codeword can be detected and corrected. Quantum error correcting codes (QECCs) extend this basic idea to the quantum domain but require several modifications to allow the codes to handle quantum, rather than classical, information.

### *14.3.1 Classical Error Correcting Codes*

The simplest classical error correcting code maps the logical bits 0 and 1 into a pair of carefully chosen bitstrings, i.e., *codewords*, chosen so as to be maximally distinguishable from one another. Once so encoded, if a bit-flip occurs within a codeword, causing it to become corrupted, the error can be readily identified and then corrected by replacing the corrupted codeword with the "closest" legal codeword to it. Typically, the distance metric used to assess "closeness" is the Hamming distance between bit strings. This is defined so that, if **x** and **y** are two bit strings, their Hamming distance is the number of places in which **x** and **y** differ.

Of course, far more sophisticated classical error-correcting codes can be devised by elaborating on this basic idea, e.g., by finding ways to encode tuples of logical bits (which one can think of as classical "symbols") as longer tuples of physical bits (the codewords) such that multiple bit-flips, bit drops, or bit insertions, within the codewords are detectable and correctable to the closest legal symbols. NASA did much of the pioneering work in error-correcting codes, motivated by the needs of spacecraft to communicate reliably with Earth over exceedingly large distances and extremely noisy channels. But the field has now blossomed into a rich assortment of techniques that are used routinely in terrestrial telecommunications and data storage. Not surprisingly, the field of error-correcting codes has deep roots in Shannon information theory discussed in Chap. 11.

### *14.3.2 Issues Unique to Quantum Error Correcting Codes*

Unfortunately, error correcting codes cannot be used in quite the same way in the quantum context as they are used in classical context. The problem is that, even if we have mapped the qubits into quantum codewords, we still cannot read a potentially corrupted quantum codeword *directly* at any intermediate step of a quantum computation in an attempt to detect an error. To do so, would cause the superposition to collapse in some unpredictable way, thereby erasing whatever remnants of correct information lay buried in the corrupted encoded state. In fact, such measurements would be likely to make the error worse rather than better. In the early years of quantum computing, this apparent prohibition on reading the corrupted encoded state to extract an error-syndrome led some researchers to speculate that quantum error-correcting codes could not exist [227, 300, 301, 502]. This cast severe doubt on the feasibility of quantum computers, because it seemed as though they would require absolute perfection in fabrication, initialization, operation, and readout—which are not likely, in practice, to be achievable. Thus, the apparent impossibility of quantum error-correcting codes seemed like a major obstacle to the development of quantum computing, because other quantum error correction schemes, such as error-correction via symmetrization, were insufficient to correct all the types of errors that were likely to arise in real quantum computing hardware.

The situation changed in 1995, however, when Peter Shor published the first account of a viable quantum error correcting code [456]. Shor's idea was to encode each logical qubit whose state we wanted to protect within a specially crafted *entangled* state of several qubits. The encoding scheme was such that any error afflicting one of these entangled qubits thereafter could be identified by making measurements on a *subset* of the qubits to obtain what was called an "error syndrome". Once the error syndrome was known, the error that afflicted the logical state we were trying to protect could be reversed by applying an appropriate sequence of unitary gates (i.e., error recovery operations) that were different depending on whichever error syndrome had been obtained. All subsequent quantum codes have followed this basic pattern.

In the following sections we describe the theory of quantum error correcting codes. We will start by specifying the error model as we have to know what kinds of errors can afflict our logical qubits in order to devise codes to detect and correct such errors. We then outline the properties any quantum error correcting code needs to possess to enable it to protect quantum information that is, by its very nature, unreadable without corruption. Finally, we will look at error diagnosis and recovery.

### *14.3.3 Modeling Errors in Terms of Error Operators*

We can think of an "error" as change in the state of our logical qubit that is caused because it is not as well isolated from its environment as it is supposed to be. In this case, instead of the quantum mechanical evolution being the desired unitary evolution on the qubit alone, we obtain instead an undesired unitary evolution on the *joint* state of the qubit *and* its environment. If we then considered the state of the qubit alone, it would no longer be pure but rather now mixed. Thus, we model the error by imagining that our qubit has accidentally become part of a larger quantum system.

If we adopt this perspective, we can develop a mathematical model of how different types of errors will affect the state of our qubit. Let us imagine that the qubit starts off in state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ and the environment starts off in state $|E\rangle$. As the qubit and its environment are assumed to start off independently of one another their initial joint state is a product state of the form:

$$|\Psi\rangle = |\psi\rangle \otimes |E\rangle = (\alpha|0\rangle + \beta|1\rangle) \otimes |E\rangle \tag{14.28}$$

Under a general unitary evolution, $U$, the $|0\rangle|E\rangle$ and $|1\rangle|E\rangle$ components would, ignoring normalization, evolve according to:

$$\begin{aligned} U(|0\rangle \otimes |E\rangle) &= |0\rangle \otimes |E_{00}\rangle + |1\rangle \otimes |E_{01}\rangle \\ U(|1\rangle \otimes |E\rangle) &= |0\rangle \otimes |E_{10}\rangle + |1\rangle \otimes |E_{11}\rangle \end{aligned} \tag{14.29}$$

where $|E_{00}\rangle$, $|E_{01}\rangle$, $|E_{10}\rangle$, and $|E_{11}\rangle$ do not have to be orthogonal to one another. Thus, a qubit in state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ evolves as:

$$\begin{aligned} U|\psi\rangle|E\rangle &= U((\alpha|0\rangle + \beta|1\rangle)|E\rangle) \\ &= \alpha(|0\rangle \otimes |E_{00}\rangle + |1\rangle \otimes |E_{01}\rangle) + \beta(|0\rangle \otimes |E_{10}\rangle + |1\rangle \otimes |E_{11}\rangle) \end{aligned} \tag{14.30}$$

We can re-write the right hand side of (14.30) in terms of distinct states for the qubit. The resulting form indicates that the state of the environment is correlated with the

state of the qubit. In fact, the two are *entangled*.

$$\begin{aligned} U|\psi\rangle|E\rangle &= \alpha(|0\rangle\otimes|E_{00}\rangle + |1\rangle\otimes|E_{01}\rangle) + \beta(|0\rangle\otimes|E_{10}\rangle + |1\rangle\otimes|E_{11}\rangle) \\ &= (\alpha|0\rangle+\beta|1\rangle)\otimes\frac{|E_{00}\rangle+|E_{11}\rangle}{2} \quad \text{(no error)} \\ &\quad + (\alpha|0\rangle-\beta|1\rangle)\otimes\frac{|E_{00}\rangle-|E_{11}\rangle}{2} \quad \text{(phase flip)} \\ &\quad + (\alpha|1\rangle+\beta|0\rangle)\otimes\frac{|E_{01}\rangle+|E_{10}\rangle}{2} \quad \text{(bit flip)} \\ &\quad + (\alpha|1\rangle-\beta|0\rangle)\otimes\frac{|E_{01}\rangle-|E_{10}\rangle}{2} \quad \text{(joint phase flip \& bit flip)} \end{aligned} \tag{14.31}$$

Loosely speaking,[2] this allows us to *interpret* the undesired unitary evolution of the joint state of the qubit and its environment as if one of four possible events have afflicted the qubit: no-error occurred (in which case $\alpha|0\rangle+\beta|1\rangle \to \alpha|0\rangle+\beta|1\rangle$), a phase-flip error occurred (in which case $\alpha|0\rangle+\beta|1\rangle \to \alpha|0\rangle-\beta|1\rangle$), a bit-flip error occurred (in which case $\alpha|0\rangle+\beta|1\rangle \to \alpha|1\rangle+\beta|0\rangle$), or simultaneous phase flip and bit flip errors occurred (in which case $\alpha|0\rangle+\beta|1\rangle \to \alpha|1\rangle-\beta|0\rangle$ up to an unimportant overall phase).

The alert reader will recognize these four error modes as being describable by the action of one of the Pauli matrices, $\mathbb{1}$, $X$, $Y$, and $Z$, on the error-free qubit state $|\psi\rangle=\alpha|0\rangle+\beta|1\rangle$. Recognizing that a Pauli $Y$ operation is $Y = X\cdot Z$ up to an overall phase factor, we can write:

$$\alpha|0\rangle+\beta|1\rangle \xrightarrow{\mathbb{1}} \alpha|0\rangle+\beta|1\rangle \quad \text{(no error)} \tag{14.32}$$

$$\alpha|0\rangle+\beta|1\rangle \xrightarrow{X} \alpha|0\rangle-\beta|1\rangle \quad \text{(phase flip error)} \tag{14.33}$$

$$\alpha|0\rangle+\beta|1\rangle \xrightarrow{Z} \alpha|1\rangle+\beta|0\rangle \quad \text{(bit flip error)} \tag{14.34}$$

$$\alpha|0\rangle+\beta|1\rangle \xrightarrow{X\cdot Z} \alpha|1\rangle-\beta|0\rangle \quad \text{(simultaneous phase flip \& bit flip error)} \tag{14.35}$$

where $X\cdot Z = -iY$. Thus, essentially, the error afflicting the qubit can be thought of as an "unwanted" evolution of the qubit under the action of one of the four Pauli matrices. These correspond to no-error ($\mathbb{1}$), a bit-flip error ($X$), a phase-flip error ($Z$), and a joint bit-flip and phase-flip error ($Y = iX\cdot Z$). In retrospect, this is not surprising perhaps, because the Pauli matrices form a basis for all $2\times 2$ matrices. That is, *any* matrix $\left(\begin{smallmatrix} a & b \\ c & d\end{smallmatrix}\right)$ can be factored as:

$$\begin{pmatrix} a & b \\ c & d\end{pmatrix} = \frac{a+d}{2}\mathbb{1} + \frac{b+c}{2}X + \frac{i(b-c)}{2}Y + \frac{a-d}{2}Z \tag{14.36}$$

[2] We say "loosely speaking" because we can only really adopt this interpretation when the states of the environment are orthogonal to one another.

We can extend the aforementioned error model to multiple qubits by assuming the various error types afflict each qubit independently. Thus, the operators describing all possible independent errors that might afflict $n$-qubits are precisely those of the Pauli group $\mathcal{P}_n$—the group consisting of all direct products of the Pauli operators $\mathbb{1}$, $X$, $X \cdot Z$, and $Z$ having overall phase $\pm 1$ or $\pm i$. Thus, if we were interested in encoding, say, one qubit into an entangled state of five qubits, there would (ignoring overall phase) be $4^5 = 1024$ distinct error operators formed from the direct product of a single qubit error operator, $\mathbb{1}$, $X$, $Z$, $X \cdot Z$, for each qubit in all possible ways. However, if we only wanted to guarantee the ability to correct up $t$ errors amongst any of these five qubits, then we need only consider a sub-group of these Pauli operators that contained at most $t$ Pauli terms (treating $X \cdot Z$ as *one* "Pauli" term as $X \cdot Z = -iY$).

The error operators that make up the Pauli group possess certain properties that will be of use to us later:

- The eigenvalues of $\mathcal{E}_\alpha \in \mathcal{P}_n$ are $\pm 1$ or $\pm i$.
- Squaring an error operator is the identity up to a real phase factor of $\pm 1$, i.e., $\forall \mathcal{E}_\alpha \in \mathcal{P}_n : \mathcal{E}_\alpha^2 = \pm \mathbb{1}$
- The group is closed under the dot product of its elements, i.e., $\forall \mathcal{E}_\alpha, \mathcal{E}_\beta \in \mathcal{P}_n : \mathcal{E}_\alpha \cdot \mathcal{E}_\beta \in \mathcal{P}_n$
- Pairs of error operators either commute or anti-commute, i.e., $\forall \mathcal{E}_\alpha, \mathcal{E}_\beta \in \mathcal{P}_n : [\mathcal{E}_\alpha, \mathcal{E}_\beta] = 0$ (commute) or $\{\mathcal{E}_\alpha, \mathcal{E}_\beta\} = 0$ (anti-commute)

Later we will specialize our interest to sub-groups of these operators, e.g., the sub-group of error operators that contain at most one Pauli error per operator.

### *14.3.4 Protecting Quantum Information via Encoding*

Next we turn to the question of how to encode the logical state we want to protect within a larger Hilbert space so that any errors that subsequently afflict our encoded state can be guaranteed to be detectable and correctable.

The trick, as John Preskill says, is "to use entanglement to fight entanglement". The key idea is to entangle, in a special way, a logical qubit we want to protect with $n$ ancillae qubits such that a subsequent measurement, in the computational basis, of just the $n$ ancillae qubits will project the (now specially entangled) $(n+1)$-qubit state into a *different orthogonal subspace* depending on which type of error ("bit-flip", "phase-flip" "joint bit-flip and phase-flip", or "no-error") has afflicted which of the $n+1$ qubits. The set of quantum states that span this encoding space are called the "quantum codewords", $\{|\psi_i\rangle\}$. The rationale for doing this is that, if the right set of measurements on $n$ of the $(n+1)$ qubits can project the $(n+1)$-qubit state into a different orthogonal subspace depending on what error occurred, we can use the outcome of these measurements to serve as a so-called "error-syndrome", which diagnoses what error occurred. Once, the error has become known, it then becomes easy to correct it by applying the appropriate unitary operator.

To ensure our quantum codewords will behave in the way need them to, they must be designed with the type of error we want them to protect against in mind. In fact, a little thought allows us to stipulate a criterion that the quantum codewords will have to meet in order to guarantee that we can always detect an error [270].

**Quantum Codewords: Criterion for Errors to Be Detectable** For an error $\mathcal{E}_\alpha \in \mathcal{E}$ that afflicts a quantum codeword to be detectable then, for every pair of valid quantum codewords $|\psi_i\rangle$ and $|\psi_j\rangle$ that span the encoding space, we require:

$$\langle\psi_j|\mathcal{E}_\alpha|\psi_i\rangle = c_\alpha \delta_{ij} \tag{14.37}$$

When this criterion is met, it will guarantee that an error-afflicted codeword, $\mathcal{E}_\alpha|\psi_i\rangle$, will be distinguishable from all the valid codewords, $|\psi_j\rangle$.

The aforementioned criterion tells us a property our codewords will need to possess to be able to *detect* errors. But what property must they possess to also be able to *correct* errors? Well, what do we need to ensure to guarantee we can correct any error? We have to be certain that we won't confuse one error with another when acting on different quantum codewords. Rather, the error syndrome has to be unique for each of the different types of errors acting on the different possible quantum codewords. This basic strategy is the foundation of all quantum error correcting codes (QECCs).

Thus, to ensure our codewords will be useful for correcting errors, in addition to detecting them, we therefore need them to satisfy the following correctability criterion.

**Quantum Codewords: Criterion for Errors to Be Correctable** For an error $\mathcal{E}_\alpha \in \mathcal{E}$ that afflicts a quantum codeword to be correctable, it needs to be distinguishable from all errors afflicting all other codewords. That is, if $|\psi_i\rangle$ and $|\psi_j\rangle$ are any pair of valid codewords, we require:

$$\langle\psi_j|\mathcal{E}_\beta^\dagger\mathcal{E}_\alpha|\psi_i\rangle = c_{\alpha\beta}\delta_{ij} \quad \forall\mathcal{E}_\alpha, \mathcal{E}_\beta \in \mathcal{E} \tag{14.38}$$

When this criterion is met, we can guarantee that an error $\mathcal{E}_\alpha \in \mathcal{E}$ afflicting one codeword is distinguishable from an error $\mathcal{E}_\beta \in \mathcal{E}$ afflicting a different codeword. In this case we would have $\langle\psi_i|\mathcal{E}_\beta^\dagger\mathcal{E}_\alpha|\psi_j\rangle = 0$. Furthermore, the criterion also guarantees that when different error operators afflict the same codeword, as described by $\langle\psi_i|\mathcal{E}_\beta^\dagger\mathcal{E}_\alpha|\psi_i\rangle$, that the result is independent of the codeword. This means that neither the environment nor the decoding operation learns anything about the encoded state during error detection and correction. This is an essential requirement to be sure the error detection and correction procedures do not cause more damage to the state we are trying to protect.

Thus, it should be apparent that the family of errors that we want to be able to detect and correct, and the number of qubits into which we encode each logical qubit we want to protect, will influence the options available to us for picking quantum

codewords that meet the detectability and correctability criteria. As we show below, quantum codewords having the desired properties can be constructed, and we will give examples of 9-qubit, 7-qubit, and 5-qubit coding schemes that are able to correct a single Pauli error, $\mathbb{1}$, $X$, $Z$, $X \cdot Z$, afflicting any of their qubits.

### *14.3.5 Digitizing and Diagnosing Errors by Measuring Error Syndromes*

A striking aspect of such quantum error-correcting codes, is that the act of measuring the ancillae qubits to obtain the error-syndrome can be viewed as *determining which error has afflicted which qubit*. Prior to such measurements, which error (if any) has occurred is undetermined. In fact, pre-measurement, the state may contain a superposition of possible errors any of which are still possible outcomes. However, by making the error-syndrome measurements a particular error is determined. Forcing such an error decision is a rational thing to do, because the error then becomes known, and a large known error is entirely correctable, whereas a small unknown one is not. So the cleverness of quantum error-correcting codes is that they exploit the superposition-destroying nature of quantum measurements to render an unknown error known, and entanglement to link the measured error-syndrome to the error-type afflicting the logical qubit.

### *14.3.6 Reversing Errors via Inverse Error Operators*

Once the error becomes known, as a result of measuring the error syndrome, it can be corrected by applying the inverse of the appropriate Pauli error operator.

### *14.3.7 Abstract View of Quantum Error Correcting Codes*

The general approach to quantum error correcting codes outlined above, can be abstracted into a theory based on the properties of operators and sub-spaces. Stepping back a moment, the general idea is to encode a logical qubit whose state we want to protect within a set of $n$-qubits, i.e., within a $2^n$-dimensional Hilbert space, such that there is a special sub-space $\mathcal{C}$, called the codespace, that is spanned by a set of quantum states, $\text{span}(\{|\psi_i\rangle\})$, i.e. the quantum codewords. The codewords are carefully chosen so that we can guarantee, for a given set of error operators, $\mathcal{E}$, that the error detectability and error correctability criteria are met. That is, every error operator $\mathcal{E}_\alpha \in \mathcal{E}$ takes a codeword into a state that is orthogonal to all other codewords, and the errors induced by one error operator can be distinguished from those induced by another. Thus every error is uniquely identifiable and hence correctable.

#### 14.3.7.1 Minimal Distance of a Code

Our primary concern is how many errors a given code can correct? We approach this by determining the *distance* of the code.

Let us start with the error operators. These are all direct products of 1-qubit Pauli matrices and the identity matrix. Define the *weight* of such an operator to be the number non-identity operators in its direct product representation. We can then define the *minimum distance* of a code to be equal to the smallest weight of any operator $\mathcal{E}_\gamma \in \mathcal{E}$ such that the error correctability criterion (14.38) is violated.

What does this imply about the relationship between the distance of a code and how many errors it can correct? Well, for a QECC to be useful, it has to be able to distinguish between how different errors affect different codewords. So in the correctability criterion we use the operator $\mathcal{E}_\beta^\dagger \mathcal{E}_\alpha$. But if error operators $\mathcal{E}_\alpha$ and $\mathcal{E}_\beta$ are in the group $\mathcal{E}$, then so is the operator $\mathcal{E}_\gamma = \mathcal{E}_\beta^\dagger \mathcal{E}_\alpha$. If we are working with a sub-group of error operators such that each operator contains at most $t$ Pauli matrices, then the operator $\mathcal{E}_\beta^\dagger \mathcal{E}_\alpha$ could have weight up to $2t$. To guarantee correctability we therefore require the minimum distance $d$ to exceed this potential weight, i.e., $d \geq 2t + 1$. This implies that our code can only be guaranteed to correct up to at most $t = \lfloor \frac{d-1}{2} \rfloor$ general errors, i.e., bit-flips, phase-flips, or joint bit-flips and phase flips.

#### 14.3.7.2 $[\![n, K, d]\!]$ Quantum Error Correcting Code

Thus, the principal characteristics of a quantum code are the number of qubits used in the encoding, $n$, the dimension of the codespace, $K$, and the minimum distances of the code $d$, which is related to the maximum number of errors the code can be guaranteed to correct, $t_{\max}$, via $t_{\max} = \lfloor \frac{d-1}{2} \rfloor$. Quantum error correcting codes are therefore often described using the notation $[\![n, K, d]\!]$. An $[\![n, K, d]\!]$ code can *detect* up to $(d-1)$ errors, and *correct* up to $\lfloor \frac{(d-1)}{2} \rfloor$ general 1-qubit errors. The smallest quantum error correcting code able to correct a single general error is a $[\![5, 2, 3]\!]$ code. In this case, $n = 5$, $K = 2$, $d = 3$ and so $t_{\max} = \lfloor \frac{d-1}{2} \rfloor = 1$.

#### 14.3.7.3 Additive (Stabilizer) Code Versus Non-additive Code

Within the class of quantum codes, the most important distinction is between the additive codes and the non-additive ones. If the codespace of a quantum error correcting code is specified by the joint $+1$ eigenspace of an Abelian sub-group of local Pauli operators (i.e., operators writable as a direct product of Pauli matrices that all commute with one another), then the code is said to be an "additive" or "stabilizer" code. That is if the errors are specified as an Abelian sub-group of the Pauli group, and have the property that on the codewords $\{|\psi_i\rangle\}$ that $\forall \mathcal{E}_\alpha \in \mathcal{E} \in \mathcal{P}_n : \mathcal{E}_\alpha |\psi_i\rangle = +1 |\psi_i\rangle$, then the code is an additive or stabilizer code.

If the aforementioned condition on the codespace does not hold, the code is "non-additive".

**Table 14.4** Notation often used to describe classical and quantum error-correcting codes

| Notation | Name | Meaning |
|---|---|---|
| $(n, K, d)$ | Classical code | An $n$-bit classical code having a $K$-dimensional codespace and distance $d$ |
| $\langle\!\langle n, K, d \rangle\!\rangle$ | Quantum code | An $n$-qubit quantum code having a $K$-dimensional codespace and distance $d$. This class includes additive and non-additive quantum codes. The latter have the potential to be more efficient than additive codes. Note that the codespace dimension of a non-additive code need not be a power of two |
| $[\![n, k, d]\!]$ | Quantum stabilizer code | An additive (stabilizer) $n$-qubit quantum code having a $2^k$-dimensional codespace and distance $d$. This class of quantum codes includes the 9-qubit Shor, 7-qubit Steane, and 5-qubit Laflamme codes. The codespace dimension of an additive code is always a power of two |

The notation $\langle\!\langle n, K, d \rangle\!\rangle$ is used to describe both additive and non-additive quantum codes. However, the codespace dimension of additive codes is always a power of two, i.e., $K = 2^k$ for some $k$, whereas this is not necessarily so for a non-additive code. The additive (stabilizer) codes are often described in terms of a special notation $[\![n, k, d]\!]$ (where $k = \log_2 K$). Thus, whereas we can speak of an additive code protecting $k$ qubits within $n$-qubit quantum codewords, we cannot really say this for a non-additive code since $\log_2 K$ is not necessarily an integer. However, the greater complexity of non-additive codes is offset by their potential to be more efficient than additive codes. Table 14.4 summarizes the notation we just discussed.

Quantum codes have other characteristics that can be of interest including whether they are pure or impure, degenerate or non-degenerate, and perfect or imperfect.

#### 14.3.7.4 Pure Versus Impure Code

If distinct elements of $\mathcal{E}$ produce orthogonal results, the code is said to be pure. Otherwise it is impure.

#### 14.3.7.5 Degenerate Versus Non-degenerate Code

If linearly independent correctable errors acting on the codespace are guaranteed to yield linearly independent states, the code is said to be *non-degenerate*. Thus, a non-degenerate code assigns a unique linearly independent error-syndrome to each possible error. Most known quantum error correcting codes are non-degenerate additive (stabilizer) codes. If a additive (stabilizer) code is also a pure code, it is guaranteed to be non-degenerate, but the converse need not be true.

Conversely, if linearly independent correctable errors acting on the codeword space can produce linearly dependent states, the code is said to be *degenerate*. Degenerate codes are interesting because they have the potential to be much more efficient than non-degenerate codes.

Theorems placing bounds on non-degenerate quantum codes do not necessarily apply to degenerate codes. Therefore, before applying a theorem, verify that the theorem holds for the type of code with which you are working.

#### 14.3.7.6 Perfect Versus Imperfect Code

If every error syndrome corresponds to a correctable error, the code is said to be *perfect* otherwise it is imperfect.

Having re-considered quantum correcting codes in the abstract let us now turn to a concrete example of the optimal additive quantum code able to correct a single general error.

## 14.4 Optimal Quantum Error Correcting Code

A natural question to ask is how good a quantum error correcting code can be? That is what are the tradeoffs between the number of qubits used in the codeword, the number of qubits the code protects, and the number of general errors that such a code can correct? The following simple argument suggests a $[\![5, 1, 3]\!]$ code is the smallest code able to correct a single general error.

### *14.4.1 Laflamme-Miquel-Paz-Zurek's 5-Qubit Code*

Imagine a code that encodes one logical qubit into $n$ qubits and we wish to protect against a single error on any one of these $n$ qubits. If we assume that errors are sufficiently rare that at most one error can afflict one of the $n$ qubits, then each qubit can undergo one of three types of error so there are $3n$ ways the error can be introduced. Add to this the possibility that none of the qubits have an error, we obtain a total of $(3n+1)$ possible error "diagnoses". If we are to distinguish between the possible $(3n+1)$ error diagnoses by making measurements on $n-1$ qubits, i.e., the ancillae, then these can index $2^{n-1}$ different states, and so the code has to satisfy $(3n+1) \leq 2^{n-1}$. The smallest integer satisfying this condition is $n = 5$. Hence, the smallest code sufficient to correct a single general error must be at least a 5-qubit code. Such a 5-qubit code was constructed by Raymond Laflamme, Cesar Miquel, Pablo Paz, and Wojciech Zurek [297] in 1996.

### *14.4.2 Error Operators for the 5-Qubit Code*

In the Laflamme-Miquel-Paz-Zurek 5-qubit code we wish to be able to correct a general error amongst the five qubits in the encoded state. Hence, the only error

operators we need consider are:

$$\begin{aligned}
\mathcal{E}_{\text{None}} &= \mathbb{1}\otimes\mathbb{1}\otimes\mathbb{1}\otimes\mathbb{1}\otimes\mathbb{1}\\
\mathcal{E}_{\text{B5}} &= \mathbb{1}\otimes\mathbb{1}\otimes\mathbb{1}\otimes\mathbb{1}\otimes X\\
\mathcal{E}_{\text{BP5}} &= \mathbb{1}\otimes\mathbb{1}\otimes\mathbb{1}\otimes\mathbb{1}\otimes X\cdot Z\\
\mathcal{E}_{\text{P5}} &= \mathbb{1}\otimes\mathbb{1}\otimes\mathbb{1}\otimes\mathbb{1}\otimes Z\\
\mathcal{E}_{\text{B4}} &= \mathbb{1}\otimes\mathbb{1}\otimes\mathbb{1}\otimes X\otimes\mathbb{1}\\
\mathcal{E}_{\text{BP4}} &= \mathbb{1}\otimes\mathbb{1}\otimes\mathbb{1}\otimes X\cdot Z\otimes\mathbb{1}\\
\mathcal{E}_{\text{P4}} &= \mathbb{1}\otimes\mathbb{1}\otimes\mathbb{1}\otimes Z\otimes\mathbb{1}\\
\mathcal{E}_{\text{B3}} &= \mathbb{1}\otimes\mathbb{1}\otimes X\otimes\mathbb{1}\otimes\mathbb{1}\\
\mathcal{E}_{\text{BP3}} &= \mathbb{1}\otimes\mathbb{1}\otimes X\cdot Z\otimes\mathbb{1}\otimes\mathbb{1}\\
\mathcal{E}_{\text{P3}} &= \mathbb{1}\otimes\mathbb{1}\otimes Z\otimes\mathbb{1}\otimes\mathbb{1}\\
\mathcal{E}_{\text{B2}} &= \mathbb{1}\otimes X\otimes\mathbb{1}\otimes\mathbb{1}\otimes\mathbb{1}\\
\mathcal{E}_{\text{BP2}} &= \mathbb{1}\otimes X\cdot Z\otimes\mathbb{1}\otimes\mathbb{1}\otimes\mathbb{1}\\
\mathcal{E}_{\text{P2}} &= \mathbb{1}\otimes Z\otimes\mathbb{1}\otimes\mathbb{1}\otimes\mathbb{1}\\
\mathcal{E}_{\text{B1}} &= X\otimes\mathbb{1}\otimes\mathbb{1}\otimes\mathbb{1}\otimes\mathbb{1}\\
\mathcal{E}_{\text{BP1}} &= X\cdot Z\otimes\mathbb{1}\otimes\mathbb{1}\otimes\mathbb{1}\otimes\mathbb{1}\\
\mathcal{E}_{\text{P1}} &= Z\otimes\mathbb{1}\otimes\mathbb{1}\otimes\mathbb{1}\otimes\mathbb{1}
\end{aligned} \tag{14.39}$$

which includes, you notice, the possibility of there being no errors at all, i.e., $\mathcal{E}_{\text{None}}$.

### *14.4.3 Encoding Scheme for the 5-Qubit Code*

In the Laflamme-Miquel-Paz-Zurek code a single logical qubit is encoded in a 5-qubit entangled state of the form:

$$\begin{aligned}
|0\rangle_L = \frac{1}{2\sqrt{2}}\big(&|00000\rangle + |00110\rangle + |01001\rangle - |01111\rangle\\
&+ |10011\rangle + |10101\rangle + |11010\rangle - |11100\rangle\big)\\
|1\rangle_L = \frac{1}{2\sqrt{2}}\big(&|11111\rangle + |11001\rangle + |10110\rangle - |10000\rangle\\
&- |01100\rangle - |01010\rangle - |00101\rangle + |00011\rangle\big)
\end{aligned} \tag{14.40}$$

A general state of a qubit we want to protect, $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, is mapped into an entangled state of the form $|\psi\rangle_L = \alpha|0\rangle_L + \beta|1\rangle_L$. Subsequently, if a single bit-flip, phase-flip, or joint bit-flip and phase-flip afflicts any of these five qubits, there is sufficient information in their entanglement to be able to determine, from the measured error syndrome, the operation that must be performed on the unmeasured qubit to restore it to its original state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$.

Figure 14.3 shows a quantum circuit for creating the entangled encoded state used in the Laflamme-Miquel-Paz-Zurek quantum error-correcting code.This circuit entangles a single qubit in an arbitrary state $|\psi\rangle$ with four ancillae qubits, each initially in state $|0\rangle$, to create the encoded state $|\psi\rangle_L = \alpha|0\rangle_L + \beta|1\rangle_L$, having basis vectors $|0\rangle_L$ and $|1\rangle_L$ as in (14.40), and single qubit gates $L$ and $L^\dagger$ defined by:

$$\begin{aligned} L &= \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \\ L^\dagger &= \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \end{aligned} \tag{14.41}$$

After such an encoding, the 5-qubit state may then be afflicted with a single bit-flip, phase-flip, or joint bit-flip and phase-flip in the region marked "ERROR" in Fig. 14.4. This would correspond to an error being introduced while an encoded qubit was being stored in a quantum memory. For example, if a single bit-flip occurs

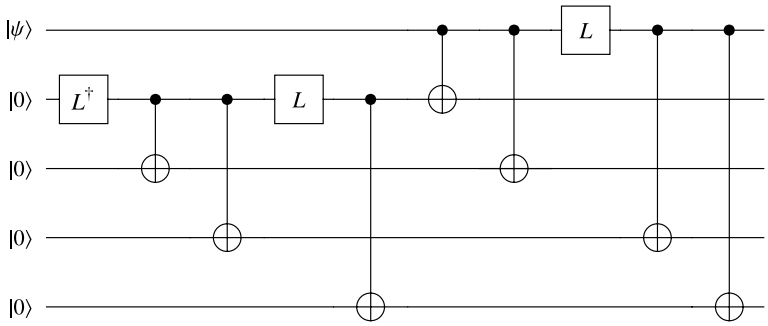


**Fig. 14.3** Quantum circuit for encoding unknown quantum state $|\psi\rangle$ amongst the amplitudes of a 5-qubit entangled state such that any subsequent bit flip, phase shift or joint bit flip/phase shift error can be detected and corrected

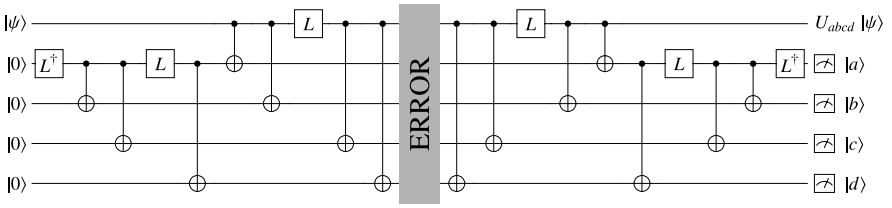


**Fig. 14.4** Quantum circuit implementing the Laflamme-Miquel-Paz-Zurek 5-qubit quantum error correcting code. The *left hand size* of the circuit encodes a single logical qubit in an entangled 5-qubit state. In the encoded form, the state is protected against a single bit-flip, phase-flip, or joint bit-flip and phase-flip acting on any of the five qubits. To diagnose and correct the error, the encoded state must be decoded and the error syndrome measured. Depending on the outcome, $|a\rangle|b\rangle|c\rangle|d\rangle$, a unitary operator $U_{abcd}$ is applied to the top qubit which rotates it into the original state of the logical qubit and hence the error is undone

on the third qubit of the encoded state $|\psi\rangle_L$, the state would change according to:

$$|\psi\rangle_L \xrightarrow{\mathcal{E}_{B3}} -\beta|01111\rangle + \alpha|11111\rangle \tag{14.42}$$

Likewise, if a single phase-flip occurs on the fourth qubit of the encoded state $|\psi\rangle_L$, the state would change according to:

$$|\psi\rangle_L \xrightarrow{\mathcal{E}_{P4}} \alpha|01110\rangle - \beta|11110\rangle \tag{14.43}$$

Similar state changes are induced by a bit-flip, a phase-flip, or a joint bit-flip and phase-flip on any of the qubits in the encoded state $|\psi\rangle_L$.

Now that we know how an error changes the encoded state, we next need to figure out how to diagnose what error has occurred and determine the appropriate corrective action to undo that error and restore the logical qubit to its pristine state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. To do this, we will use the part of the circuit in Fig. 14.4 to the right of the region marked "ERROR".

### *14.4.4 Error Syndromes & Corrective Actions for the 5-Qubit Code*

To diagnose what single error has afflicted the 5-qubit encoded state, we run the buggy encoded state through the decoding circuit shown in Fig. 14.5. This is just the encoding circuit run in the reverse direction. The decoding operation produces an output entangled state that can be factored as a *superposition* of the form $\sum_{a=0}^{1}\sum_{b=0}^{1}\sum_{c=0}^{1}\sum_{d=0}^{1}\alpha_{abcd}|\varphi_{abcd}\rangle|abcd\rangle$ where $|abcd\rangle$ is a 4-bit computational basis vector, and the $|\varphi_{abcd}\rangle$ states are unitary rotations of state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. We can, given knowledge of the encoding-decoding circuitry and the error-modes, build a lookup table that gives the required rotation to map each $|\varphi\rangle$ back into $|\psi\rangle$. Such rotation are shown in Table 14.5. Thus, by measuring the four ancillae qubits, in the computational basis, in the output from the decoding circuit we can project out a specific state $|\varphi_{abcd}\rangle$ state, and use Table 14.5 to determine the corrective action needed to recover the correct state ($|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$) of the logical qubit.

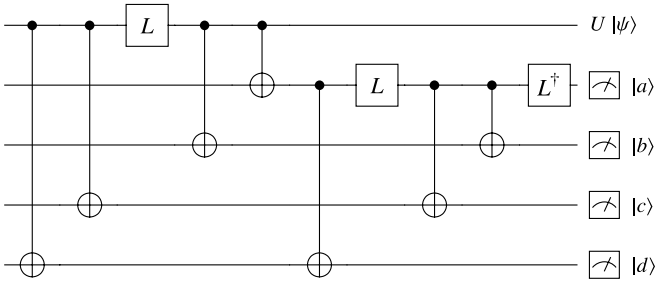


**Fig. 14.5** Quantum circuit for decoding a (potentially corrupted) entangled state and measuring its error syndrome to reveal whether or not an error had occurred, and if so, what action to apply to the top qubit to undo the error

**Table 14.5** To protect an unknown quantum state $|\psi\rangle$ while in storage in a quantum memory register, we entangle $|\psi\rangle$ with four ancilla qubits each prepared initially in the state $|0\rangle$, using the left hand side of the quantum circuit shown in Fig. 14.4. Once encoded, the state can be corrupted by a single bit-flip, a single phase-flip or a single phase-flip followed by a bit-flip on any of the five qubits. However, when we want to retrieve our protected state, we decode the entangled state by running it through the right hand side of the circuit shown in Fig. 14.4 and then "measure the error syndrome", i.e., read the bit values of the four ancilla qubits. Based on the observed values we can then look up corrective action to apply to the top qubit to restore it to its pristine (yet unknown) state. In the table an error BP$i$ means a phase flip followed by a bit flip on the $i$-th qubit

| Error type | State produced $\|\psi\rangle\|a\rangle\|b\rangle\|c\rangle\|d\rangle$ | Error syndrome | Corrective action $U_{abcd}$ | Result |
|---|---|---|---|---|
| None | $-\beta\|01011\rangle+\alpha\|11011\rangle$ | $\{1,0,1,1\}$ | $Z\cdot X$ | $(\alpha\|0\rangle+\beta\|1\rangle)\|1011\rangle$ |
| B1 | $\beta\|01000\rangle+\alpha\|11000\rangle$ | $\{1,0,0,0\}$ | $X$ | $(\alpha\|0\rangle+\beta\|1\rangle)\|1000\rangle$ |
| B2 | $-\beta\|00010\rangle+\alpha\|10010\rangle$ | $\{0,0,1,0\}$ | $Z\cdot X$ | $(\alpha\|0\rangle+\beta\|1\rangle)\|0010\rangle$ |
| B3 | $-\beta\|01111\rangle+\alpha\|11111\rangle$ | $\{1,1,1,1\}$ | $Z\cdot X$ | $(\alpha\|0\rangle+\beta\|1\rangle)\|1111\rangle$ |
| B4 | $-\beta\|01001\rangle+\alpha\|11001\rangle$ | $\{1,0,0,1\}$ | $Z\cdot X$ | $(\alpha\|0\rangle+\beta\|1\rangle)\|1001\rangle$ |
| B5 | $-\beta\|01010\rangle+\alpha\|11010\rangle$ | $\{1,0,1,0\}$ | $Z\cdot X$ | $(\alpha\|0\rangle+\beta\|1\rangle)\|1010\rangle$ |
| P1 | $\alpha\|00110\rangle-\beta\|10110\rangle$ | $\{0,1,1,0\}$ | $Z$ | $(\alpha\|0\rangle+\beta\|1\rangle)\|0110\rangle$ |
| P2 | $\beta\|01101\rangle+\alpha\|11101\rangle$ | $\{1,1,0,1\}$ | $X$ | $(\alpha\|0\rangle+\beta\|1\rangle)\|1101\rangle$ |
| P3 | $\beta\|00011\rangle+\alpha\|10011\rangle$ | $\{0,0,1,1\}$ | $X$ | $(\alpha\|0\rangle+\beta\|1\rangle)\|0011\rangle$ |
| P4 | $\alpha\|01110\rangle-\beta\|11110\rangle$ | $\{1,1,1,0\}$ | $Z$ | $(\alpha\|0\rangle+\beta\|1\rangle)\|1110\rangle$ |
| P5 | $\alpha\|00000\rangle-\beta\|10000\rangle$ | $\{0,0,0,0\}$ | $Z$ | $(\alpha\|0\rangle+\beta\|1\rangle)\|0000\rangle$ |
| BP1 | $-\alpha\|00101\rangle-\beta\|10101\rangle$ | $\{0,1,0,1\}$ | $Z\cdot X\cdot Z\cdot X$ | $(\alpha\|0\rangle+\beta\|1\rangle)\|0101\rangle$ |
| BP2 | $\beta\|00100\rangle+\alpha\|10100\rangle$ | $\{0,1,0,0\}$ | $X$ | $(\alpha\|0\rangle+\beta\|1\rangle)\|0100\rangle$ |
| BP3 | $\beta\|00111\rangle+\alpha\|10111\rangle$ | $\{0,1,1,1\}$ | $X$ | $(\alpha\|0\rangle+\beta\|1\rangle)\|0111\rangle$ |
| BP4 | $\alpha\|01100\rangle-\beta\|11100\rangle$ | $\{1,1,0,0\}$ | $Z$ | $(\alpha\|0\rangle+\beta\|1\rangle)\|1100\rangle$ |
| BP5 | $\alpha\|00001\rangle-\beta\|10001\rangle$ | $\{0,0,0,1\}$ | $Z$ | $(\alpha\|0\rangle+\beta\|1\rangle)\|0001\rangle$ |

### *14.4.5 Example: Correcting a Bit-Flip*

Suppose the logical qubit we wish to protect is in state $|\psi\rangle=\alpha|0\rangle+\beta|1\rangle$. To protect this qubit against error we augment $|\psi\rangle$ state with four ancillae qubits each prepared in state $|0\rangle$ to give us the input state:

$$|\Psi_{\text{in}}\rangle=\alpha|00000\rangle+\beta|10000\rangle \tag{14.44}$$

Encoding this state using the Laflamme-Miquel-Paz-Zurek 5-qubit encoding circuit gives us the state:

$$\begin{aligned}|\Psi_{\text{middle}}\rangle=\frac{1}{2\sqrt{2}}\big(&\alpha|00000\rangle+\beta|00011\rangle-\beta|00101\rangle+\alpha|00110\rangle\\&+\alpha|01001\rangle-\beta|01010\rangle-\beta|01100\rangle-\alpha|01111\rangle\\&-\beta|10000\rangle+\alpha|10011\rangle+\alpha|10101\rangle+\beta|10110\rangle\\&+\beta|11001\rangle+\alpha|11010\rangle-\alpha|11100\rangle+\beta|11111\rangle\big)\end{aligned} \tag{14.45}$$

This is an entangled state that can now protect our logical qubit from error. For example, imagine introducing a bit-flip error on the third qubit in this state creating the buggy state:

$$\begin{aligned}|\Psi_{\text{buggy}}\rangle = \frac{1}{2\sqrt{2}}\big(&\alpha|00100\rangle + \beta|00111\rangle - \beta|00001\rangle + \alpha|00010\rangle \\ &+ \alpha|01101\rangle - \beta|01110\rangle - \beta|01000\rangle - \alpha|01011\rangle \\ &- \beta|10100\rangle + \alpha|10111\rangle + \alpha|10001\rangle + \beta|10010\rangle \\ &+ \beta|11101\rangle + \alpha|11110\rangle - \alpha|11000\rangle + \beta|11011\rangle\big) \end{aligned} \tag{14.46}$$

Decoding the buggy state using the Laflamme-Miquel-Paz-Zurek decoding circuit gives us the state:

$$|\Psi_{\text{out}}\rangle = -\beta|01111\rangle + \alpha|11111\rangle \tag{14.47}$$

Reading ancillae state $|abcd\rangle$ then gives the error syndrome is 1111. Using the lookup table, Table 14.5, the appropriate corrective action to apply to the top (unmeasured qubit) should be $Z \cdot X$. Applying this operation, we see that we do indeed restore the top qubit to its error free state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$.

Quantum error correcting codes *are* therefore feasible even though we are unable to read the encoded data directly without necessarily perturbing the state being read. The proof of the feasibility of QECCs was one of the most important discoveries in the development of quantum computing because their existence means that it is not necessary to fabricate, initialize, and run quantum computers perfectly in order to obtain correct results.

## 14.5 Other Additive Quantum Error Correcting Codes

The 5-qubit was not the first quantum error correcting code discovered that was able to correct for a single general error amongst the encoded qubits. In fact, two other codes pre-date it, but both require more qubits to encode the data being protected.

### *14.5.1 Shor's 9-Qubit Code*

The first quantum error-correcting code (QECC) was devised by Peter Shor in 1995 [456]. It encodes each logical qubit in nine physical qubits in such manner that a single bit-flip, phase-flip, or joint bit-flip and phase flip, afflicting any of these nine qubits can be identified, and undone by performing an appropriate unitary operation which differs depending on the outcome of the ancilla measurements.

The encoding step in the 9-qubit code involves mapping each logical qubit to encoded form according to:

$$\begin{aligned}|0\rangle_L &= \frac{1}{2\sqrt{2}}(|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle) \\ |1\rangle_L &= \frac{1}{2\sqrt{2}}(|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle) \end{aligned} \tag{14.48}$$

Once in this form the encoded data is protected against a single error in any qubit amongst any of the nine qubits.

### *14.5.2 Steane's 7-Qubit Code*

In 1996 Andrew Steane improved upon Peter Shor's 9-qubit code with a 7-qubit code [477, 478]. The encoding step in the 7-qubit code involves mapping each logical qubit to encoded form according to:

$$\begin{aligned}|0\rangle_L &= \frac{1}{2\sqrt{2}}\big(|0000000\rangle + |1010101\rangle + |0110011\rangle + |1100110\rangle \\ &\quad + |0001111\rangle + |1011010\rangle + |0111100\rangle + |1101001\rangle\big) \\ |1\rangle_L &= \frac{1}{2\sqrt{2}}\big(|1111111\rangle + |0101010\rangle + |1001100\rangle + |0011001\rangle \\ &\quad + |1110000\rangle + |0100101\rangle + |1000011\rangle + |0010110\rangle\big)\end{aligned} \tag{14.49}$$

Once in this form the encoded data is protected against a single error in any qubit amongst any of the seven qubits.

## 14.6 Stabilizer Formalism for Quantum Error Correcting Codes

The foregoing quantum error correcting codes were either constructed based on analogies with pre-existing classical codes, or discovered via extensive computer searches. As a result, each code was created in a somewhat makeshift fashion and few, if any, general design principles for quantum codes were learned. One could easily get the impression, therefore, that quantum error correcting codes are discovered serendipitously rather than being constructed systematically to meet desired criteria. Furthermore, one could also get the impression, from our description of the 5-qubit Laflamme-Miquel-Paz-Zurek code, that error correction requires that we periodically map the encoded (and therefore protected qubit) back to its (unprotected) logical basis at which times the qubit is exposed to uncorrectable errors. Neither of these impressions is correct.

In 1996 Daniel Gottesman invented a unified way to think about an important sub-class of QECCs that allows them to be constructed in a more systematic fashion and to perform error correction entirely within the encoded basis so we never re-expose the protected quantum information during the error correction procedure. The 9-qubit Shor, 7-qubit Steane, and 5-qubit Laflamme-Miquel-Paz-Zurek codes as special cases of Gottesman's formalism, which later became known as the "stabilizer formalism" [207, 208]. Using the stabilizer formalism it becomes straightforward to design QECCs to protect against a given set of errors, and to find quantum circuits that will perform the required error diagnosis and recovery operations while staying entirely within the encoded (and therefore protected) basis.

Rather than discuss the stabilizer formalism in the abstract, we will use it to re-analyze the 5-qubit Laflamme-Miquel-Paz-Zurek code, which is the best QECC capable of correcting a single bit-flip, phase-flip, or joint bit-flip and phase-flip afflicting any one of five qubits.

### 14.6.1 Group Theory for Stabilizer Codes

As the stabilizer formalism relies upon ideas from group theory, we will begin with a brief summary of the key ideas of group theory.

A "group", $\mathcal{G}$, is a collection of objects, $g_1, g_2, \ldots \in \mathcal{G}$, together with a multiplication operation "$\cdot$", which possess the following properties:

**Group Theory**

- **Closure**: the group is closed under "$\cdot$", i.e., if $g_i, g_j \in \mathcal{G}$ then $g_i \cdot g_j \in \mathcal{G}$.
- **Associativity**: i.e., $(g_i \cdot g_j) \cdot g_k = g_i \cdot (g_j \cdot g_k)$.
- **Existence of Identity**: the group contains an identity element, i.e., $\exists e \in \mathcal{G}$ such that $\forall g_i \in \mathcal{G}, e \cdot g_i = g_i$.
- **Existence of Inverse**: each member of the group has an inverse, i.e., $\forall g_i \in \mathcal{G}$, $\exists g_i^{-1} \in \mathcal{G}$ s.t. $g_i \cdot g_i^{-1} = e$.

In the context of quantum error correcting codes, the following types of groups and concepts are the most important.

**Types of Groups**

- **Pauli group**: the group consisting of tensor products of the Pauli matrices, $\mathbb{1}$, $X$, $Y$, $Z$, with an overall phase of $\pm 1$ or $\pm i$.
- **Finite group**: a group $\mathcal{G}$ is finite if the number of elements in it is finite, i.e., the group contains only the elements $g_1, g_2, \ldots, g_n \in \mathcal{G}$ for some finite positive integer $n$.
- **Abelian group**: a group is "Abelian" iff $\forall g_i, g_j \in \mathcal{G}, g_i \cdot g_j = g_j \cdot g_i$.
- **Sub-group**: $\mathcal{S}$ is a sub-group of $\mathcal{G}$ iff the elements $s_1, s_2, \ldots \in \mathcal{S}$ are a subset of the elements of $g_1, g_2, \ldots \in \mathcal{G}$, and obey the rules for a group in their own right under the same group multiplication operator as that of $\mathcal{G}$.

The final concept we shall need is that of a "group generator". The generator, $\{g_1, g_2, \ldots, g_\ell\}$, of a group $\mathcal{G}$ is the smallest subset of elements of $\mathcal{G}$ sufficient to generate every member of $\mathcal{G}$ under the multiplication operation for $\mathcal{G}$. That is we can obtain any element of $\mathcal{G}$ from products of the elements in $\{g_1, g_2, \ldots, g_\ell\}$ with repetitions allowed.

We can now describe the basic machinery of the stabilizer formalism using these group-theoretic concepts.

### 14.6.2 The Stabilizer

A "stabilizer" $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_K\}$ is a carefully chosen group of tensor products of the Pauli operators, $\mathcal{S}_i \in \{\mathbb{1}, X, Z\}^{\otimes n}$ whose elements are required have a simultaneous eigenvalue of $+1$. That is, for some family of states $|\psi\rangle_L$ the stabilizer $\mathcal{S}$ is

a group of tensor products of Pauli operators such that:

$$\begin{aligned} \mathcal{S}_1|\psi\rangle_L &= +1|\psi\rangle_L \\ \mathcal{S}_2|\psi\rangle_L &= +1|\psi\rangle_L \\ &\vdots \\ \mathcal{S}_K|\psi\rangle_L &= +1\ |\psi\rangle_L \end{aligned} \tag{14.50}$$

Furthermore, it is known from pure mathematics that a group of operators can only share a simultaneous eigenvalue when the operators commute with one another. This means that the stabilizer group has to be a finite *Abelian* sub-group of the Pauli group. That is, for the operators in a valid stabilizer, $\mathcal{S}_i \cdot \mathcal{S}_j = \mathcal{S}_j \cdot \mathcal{S}_i$.

### *14.6.3 Example: A Stabilizer for the 5-Qubit Code*

There are many sets of tensor products that we could pick as stabilizers, and different choices would induce different quantum error correcting codes. If we focus on the case of QECCs that involve just five physical qubits, then all the relevant stabilizers must involve only five Pauli matrices. But remember, we don't accept just any old set of tensor products. We are specifically looking for sets of tensor products that form an Abelian sub-group.

Of the many alternatives available to us, suppose we had picked the following set of tensor products of Pauli, $\mathbb{1}$, $X$, and $Z$, matrices as our stabilizer:

$$\begin{aligned} \mathcal{S}_1 &= X \otimes X \otimes Z \otimes X \otimes \mathbb{1} \\ \mathcal{S}_2 &= X \otimes Z \otimes X \otimes \mathbb{1} \otimes X \\ \mathcal{S}_3 &= Z \otimes \mathbb{1} \otimes X \otimes X \otimes Z \\ \mathcal{S}_4 &= Z \otimes X \otimes \mathbb{1} \otimes Z \otimes X \\ \mathcal{S}_5 &= \mathbb{1} \otimes Z \otimes Z \otimes Z \otimes Z \\ \mathcal{S}_6 &= \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \end{aligned}$$

This set of tensor products satisfies all the criteria for a group given above. Moreover, the group is Abelian because all its elements commute with one another. However, we can whittle this group down a little further and work just with its generators, i.e. a minimal set of group elements sufficient to generate all members of the group via their products, with repetitions allowed. In particular, we can immediately see that we do not need the element $\mathcal{S}_6 = \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1}$ because the square of any element of the stabilizer is $\mathcal{S}_6$. For example, in particular we have:

$$\mathcal{S}_1 \cdot \mathcal{S}_1 = (X \otimes X \otimes Z \otimes X \otimes \mathbb{1})^2 = (\mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1}) = \mathcal{S}_6$$

Likewise, we can drop any one of the remaining five group elements from this sub-group too. For example, we do not need (say) $\mathcal{S}_5 = \mathbb{1} \otimes Z \otimes Z \otimes Z \otimes Z$ because:

$$\begin{aligned}\mathcal{S}_1 \cdot \mathcal{S}_2 \cdot \mathcal{S}_3 \cdot \mathcal{S}_4 &= (X \otimes X \otimes Z \otimes X \otimes \mathbb{1}) \cdot (X \otimes Z \otimes X \otimes \mathbb{1} \otimes X) \\ &\quad \cdot (Z \otimes \mathbb{1} \otimes X \otimes X \otimes Z) \cdot (Z \otimes X \otimes \mathbb{1} \otimes Z \otimes X) \\ &= (\mathbb{1} \otimes Z \otimes Z \otimes Z \otimes Z) = \mathcal{S}_5 \end{aligned} \tag{14.51}$$

Thus, to define the stabilizer, we need only work with the generators of the associated Abelian sub-group, namely the tensor products $\langle \mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4 \rangle$ where:

$$\begin{aligned}\mathcal{S}_1 &= X \otimes X \otimes Z \otimes X \otimes \mathbb{1} \\ \mathcal{S}_2 &= X \otimes Z \otimes X \otimes \mathbb{1} \otimes X \\ \mathcal{S}_3 &= Z \otimes \mathbb{1} \otimes X \otimes X \otimes Z \\ \mathcal{S}_4 &= Z \otimes X \otimes \mathbb{1} \otimes Z \otimes X\end{aligned}$$

With these definitions, the tensor products in the generator $\{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4\}$ is an Abelian sub-group of the Pauli group and therefore meets all the criteria needed to be a stabilizer.

### *14.6.4 Using a Stabilizer to Find the Codewords It Stabilizes*

Given a choice of stabilizer $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_K\}$ we can find the family of states $|\psi\rangle_L$ it stabilizes quite easily. As the $\mathcal{S}_j$ are all hermitian, we can characterize the states we seek as the $+1$ simultaneous eigenstates of the operators $\{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4\}$. These are the states spanned by the encoded basis vectors $|0\rangle_L$ and $|1\rangle_L$ that correspond to the simultaneous $+1$ eigenstates of every element of the stabilizer, when the inputs are $|00000\rangle$ and $|11111\rangle$ respectively.

In the case of the 5-qubit Laflamme-Miquel-Paz-Zurek code the stabilizer has four elements $\{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4\}$. We can measure the eigenvalue of each of these operators individually using the circuit shown in Fig. 14.6. The encoded basis state $|0\rangle_L$ is the output when the eigenvalues are all measured to be $+1$ (indicated by finding the output in state $|0\rangle$) when the input state is $|00000\rangle$.

Likewise, the encoded basis state $|1\rangle_L$ is the output from the circuit in Fig. 14.7 when the eigenvalues are all measured to be $+1$ (again, indicated by finding the output in state $|0\rangle$) when the input state is $|11111\rangle$.

Hence, the codespace that is invariant with respect to this stabilizer $\{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4\}$ is the set of states spanned by:

$$\begin{aligned}|0\rangle_L &= \frac{1}{2\sqrt{2}}\big(|00000\rangle + |00110\rangle + |01001\rangle - |01111\rangle \\ &\qquad + |10011\rangle + |10101\rangle + |11010\rangle - |11100\rangle\big) \\ |1\rangle_L &= \frac{1}{2\sqrt{2}}\big(|11111\rangle + |11001\rangle + |10110\rangle - |10000\rangle \\ &\qquad - |01100\rangle - |01010\rangle - |00101\rangle + |00011\rangle\big)\end{aligned} \tag{14.52}$$

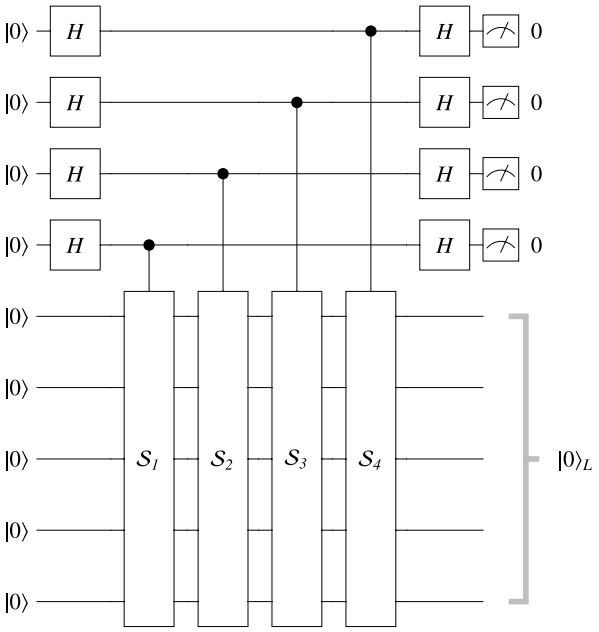


**Fig. 14.6** Quantum circuit for using the generators, $\{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4\}$, of a stabilizer sub-group $\mathcal{S}$ to find the corresponding logical 0 codeword, i.e., $|0\rangle_L$. The logical 0 has the property that $\mathcal{S}_j|0\rangle_L = +1|0\rangle_L$ for all $\mathcal{S}_j \in \mathcal{S}$ and is therefore a $+1$ eigenstate of the stabilizer sub-group $\mathcal{S}$. As in the eigenvalue estimation algorithm, when the measurement outcomes made on the ancillae are all 0000 the state on the remaining unmeasured five qubits will be projected into a simultaneous eigenstate of the operators $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$, and $\mathcal{S}_4$ with eigenvalue $+1$

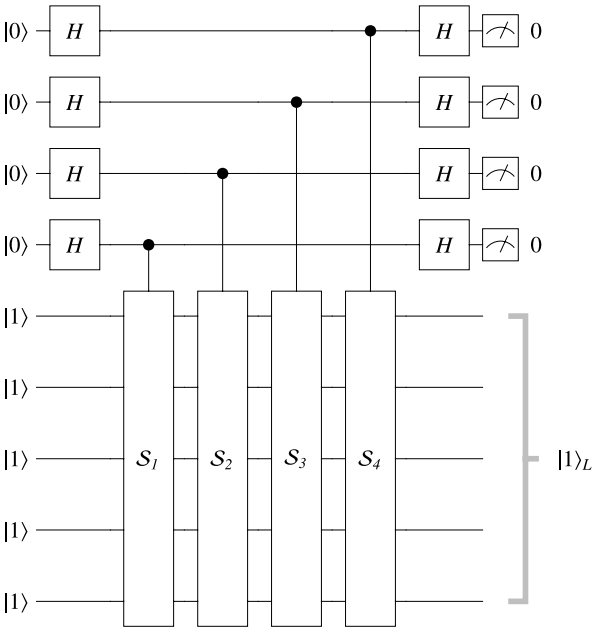


**Fig. 14.7** Quantum circuit for using the generators, $\{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4\}$, of a stabilizer sub-group $\mathcal{S}$ to find the corresponding logical 1 codeword, i.e., $|1\rangle_L$. The logical 1 has the property that $\mathcal{S}_j|1\rangle_L = +1|1\rangle_L$ for all $\mathcal{S}_j \in \mathcal{S}$ and is therefore a $+1$ eigenstate of the stabilizer sub-group $\mathcal{S}$. As in the eigenvalue estimation algorithm, when the measurement outcomes made on the ancillae are all 0000 the state on the remaining unmeasured five qubits will be projected into a simultaneous eigenstate of the operators $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$, and $\mathcal{S}_4$ with eigenvalue $+1$

These "happen to be" exactly the logical qubits we used on our 5-qubit Laflamme-Miquel-Paz-Zurek code! This means that any state of the form:

$$|\psi\rangle_L = \alpha|0\rangle_L + \beta|1\rangle_L \tag{14.53}$$

such that $|\alpha|^2 + |\beta|^2 = 1$ is stabilized by our stabilizer $\{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4\}$.

### *14.6.5 How the Stabilizer is Related to the Error Operators*

So far in our discussion I have treated a stabilizer as nothing more than an arbitrary Abelian sub-group of the Pauli group, and I "happened to pick" a stabilizer whose codewords matched those of Sam Braunstein and John Smolin's version of the pre-existing 5-qubit Laflamme-Miquel-Paz-Zurek code [79, 297]. This was intended to make the connection between stabilizer codes and the 5-qubit code explicit. But, clearly, this is cheating—as I had foreknowledge of the codewords sought, and I worked backwards to find a stabilizer that produced those codewords! It would be more honest to *start* with the *error operators* we want a quantum error correcting code to correct, and use the *error operators* to derive a stabilizer code able to protect against them. This is the purpose of this section.

In the case of the 5-qubit Laflamme-Miquel-Paz-Zurek code, our intention is to encode a single logical qubit within an entangled 5-qubit state so that the encoded qubit is protected against a single bit-flip, phase-flip, or joint bit-flip and phase flip on any of these five qubits. In this case, the family of error operators we need to protect against is, as we explained earlier, given by:

$$\begin{aligned}
\mathcal{E}_{\text{None}} &= \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \\
\mathcal{E}_{\text{B5}} &= \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes X \\
\mathcal{E}_{\text{BP5}} &= \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes X \cdot Z \\
\mathcal{E}_{\text{P5}} &= \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes Z \\
\mathcal{E}_{\text{B4}} &= \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes X \otimes \mathbb{1} \\
\mathcal{E}_{\text{BP4}} &= \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes X \cdot Z \otimes \mathbb{1} \\
\mathcal{E}_{\text{P4}} &= \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes Z \otimes \mathbb{1} \\
\mathcal{E}_{\text{B3}} &= \mathbb{1} \otimes \mathbb{1} \otimes X \otimes \mathbb{1} \otimes \mathbb{1} \\
\mathcal{E}_{\text{BP3}} &= \mathbb{1} \otimes \mathbb{1} \otimes X \cdot Z \otimes \mathbb{1} \otimes \mathbb{1} \\
\mathcal{E}_{\text{P3}} &= \mathbb{1} \otimes \mathbb{1} \otimes Z \otimes \mathbb{1} \otimes \mathbb{1} \\
\mathcal{E}_{\text{B2}} &= \mathbb{1} \otimes X \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \\
\mathcal{E}_{\text{BP2}} &= \mathbb{1} \otimes X \cdot Z \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \\
\mathcal{E}_{\text{P2}} &= \mathbb{1} \otimes Z \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \\
\mathcal{E}_{\text{B1}} &= X \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \\
\mathcal{E}_{\text{BP1}} &= X \cdot Z \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \\
\mathcal{E}_{\text{P1}} &= Z \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1}
\end{aligned} \tag{14.54}$$

which includes the possibility of there being no errors at all, i.e., $\mathcal{E}_{\text{None}}$.

Intuitively, it seems reasonable to expect that our error diagnosis and recovery operations must somehow be related to these error operators. This intuition is indeed correct. The connection is made by way of the stabilizer. Specifically, we want to pick a stabilizer such that every error operator we want to protect against, $\mathcal{E}_\alpha \in \mathcal{E}$, *anti-commutes* with at least one element of the stabilizer, $\mathcal{S}_i \in \mathcal{S}$.

The motivation for this requirement is the following. If an error operator $\mathcal{E}_\alpha \in \mathcal{E}$ *commutes* with an element of the stabilizer $\mathcal{S}_i \in \mathcal{S}$, we have $\mathcal{S}_i \cdot \mathcal{E}_\alpha = \mathcal{E}_\alpha \cdot \mathcal{S}_i$ and so:

$$\mathcal{S}_i \cdot \mathcal{E}_\alpha |\psi\rangle_L = \mathcal{E}_\alpha \cdot \mathcal{S}_i |\psi\rangle_L = +1\ \mathcal{E}_\alpha |\psi\rangle_L \tag{14.55}$$

and thus has the eigenvalue $+1$. This means that when we measure the eigenvalue of the operator $\mathcal{S}_i$ whether the input state is pristine, i.e., $|\psi\rangle_L$ or error-afflicted, i.e., $\mathcal{E}_\alpha |\psi\rangle_L$, the eigenvalue will be $+1$ either way. So a good input and a corrupted input will not be distinguishable.

However, if an error operator $\mathcal{E}_\alpha \in \mathcal{E}$ *anti-commutes* with an element of the stabilizer $\mathcal{S}_i \in \mathcal{S}$, we have $\mathcal{S}_i \cdot \mathcal{E}_\alpha = -1\ \mathcal{E}_\alpha \cdot \mathcal{S}_i$ and so:

$$\mathcal{S}_i \cdot \mathcal{E}_\alpha |\psi\rangle_L = -\ \mathcal{E}_\alpha \cdot \mathcal{S}_i |\psi\rangle_L = -1\ \mathcal{E}_\alpha |\psi\rangle_L \tag{14.56}$$

and thus has the eigenvalue $-1$. In this case, the presence of an error is signalled by the fact that the eigenvalue of the operator $\mathcal{S}_i$ has become $-1$.

Hence by measuring the eigenvalue of each element of the stabilizer we can detect whether or not an error has occurred. Moreover, the pattern of anti-commutativity over all the elements in the stabilizer is unique to each different type of error. This allows the pattern of eigenvalues to be used as an *error syndrome* that diagnoses what error occurred unambiguously.

### 14.6.6 Example: Stabilizers and Error Operators for the 5-Qubit Code

Let us make this concrete in the case of the 5-qubit Laflamme-Miquel-Paz-Zurek code. In this case the error operators we want to protect against are those defined in (14.54). As we are interested in a 5-qubit code, the potential stabilizer elements are therefore all tensor products of any five Pauli matrices taken from the set $\{\mathbb{1}, X, Z\}^{\otimes 5}$. There are 243 distinct possibilities, namely:

$$\begin{array}{ccccccccccc}
\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1} & \mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}X & \mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}Z & \mathbb{1}\mathbb{1}\mathbb{1}X\mathbb{1} & \mathbb{1}\mathbb{1}\mathbb{1}XX & \mathbb{1}\mathbb{1}\mathbb{1}XZ & \mathbb{1}\mathbb{1}\mathbb{1}Z\mathbb{1} & \mathbb{1}\mathbb{1}\mathbb{1}ZX & \mathbb{1}\mathbb{1}\mathbb{1}ZZ \\
\mathbb{1}\mathbb{1}X\mathbb{1}\mathbb{1} & \mathbb{1}\mathbb{1}X\mathbb{1}X & \mathbb{1}\mathbb{1}X\mathbb{1}Z & \mathbb{1}\mathbb{1}XX\mathbb{1} & \mathbb{1}\mathbb{1}XXX & \mathbb{1}\mathbb{1}XXZ & \mathbb{1}\mathbb{1}XZ\mathbb{1} & \mathbb{1}\mathbb{1}XZX & \mathbb{1}\mathbb{1}XZZ \\
\mathbb{1}\mathbb{1}Z\mathbb{1}\mathbb{1} & \mathbb{1}\mathbb{1}Z\mathbb{1}X & \mathbb{1}\mathbb{1}Z\mathbb{1}Z & \mathbb{1}\mathbb{1}ZX\mathbb{1} & \mathbb{1}\mathbb{1}ZXX & \mathbb{1}\mathbb{1}ZXZ & \mathbb{1}\mathbb{1}ZZ\mathbb{1} & \mathbb{1}\mathbb{1}ZZX & \mathbb{1}\mathbb{1}ZZZ \\
\mathbb{1}X\mathbb{1}\mathbb{1}\mathbb{1} & \mathbb{1}X\mathbb{1}\mathbb{1}X & \mathbb{1}X\mathbb{1}\mathbb{1}Z & \mathbb{1}X\mathbb{1}X\mathbb{1} & \mathbb{1}X\mathbb{1}XX & \mathbb{1}X\mathbb{1}XZ & \mathbb{1}X\mathbb{1}Z\mathbb{1} & \mathbb{1}X\mathbb{1}ZX & \mathbb{1}X\mathbb{1}ZZ \\
\mathbb{1}XX\mathbb{1}\mathbb{1} & \mathbb{1}XX\mathbb{1}X & \mathbb{1}XX\mathbb{1}Z & \mathbb{1}XXX\mathbb{1} & \mathbb{1}XXXX & \mathbb{1}XXXZ & \mathbb{1}XXZ\mathbb{1} & \mathbb{1}XXZX & \mathbb{1}XXZZ
\end{array}$$

$$
\begin{array}{ccccccccccc}
\mathbb{1}X\mathbb{1}\mathbb{1} & \mathbb{1}X\mathbb{1}X & \mathbb{1}X\mathbb{1}Z & \mathbb{1}XZX\mathbb{1} & \mathbb{1}XZXX & \mathbb{1}XZXZ & \mathbb{1}XZZ\mathbb{1} & \mathbb{1}XZZX & \mathbb{1}XZZZ \\
\mathbb{1}Z\mathbb{1}\mathbb{1} & \mathbb{1}Z\mathbb{1}X & \mathbb{1}Z\mathbb{1}Z & \mathbb{1}Z\mathbb{1}X\mathbb{1} & \mathbb{1}Z\mathbb{1}XX & \mathbb{1}Z\mathbb{1}XZ & \mathbb{1}Z\mathbb{1}Z\mathbb{1} & \mathbb{1}Z\mathbb{1}ZX & \mathbb{1}Z\mathbb{1}ZZ \\
\mathbb{1}ZX\mathbb{1}\mathbb{1} & \mathbb{1}ZX\mathbb{1}X & \mathbb{1}ZX\mathbb{1}Z & \mathbb{1}ZXX\mathbb{1} & \mathbb{1}ZXXX & \mathbb{1}ZXXZ & \mathbb{1}ZXZ\mathbb{1} & \mathbb{1}ZXZX & \mathbb{1}ZXZZ \\
\mathbb{1}ZZ\mathbb{1}\mathbb{1} & \mathbb{1}ZZ\mathbb{1}X & \mathbb{1}ZZ\mathbb{1}Z & \mathbb{1}ZZX\mathbb{1} & \mathbb{1}ZZXX & \mathbb{1}ZZXZ & \mathbb{1}ZZZ\mathbb{1} & \mathbb{1}ZZZX & \mathbb{1}ZZZZ \\
X\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1} & X\mathbb{1}\mathbb{1}\mathbb{1}X & X\mathbb{1}\mathbb{1}\mathbb{1}Z & X\mathbb{1}\mathbb{1}X\mathbb{1} & X\mathbb{1}\mathbb{1}XX & X\mathbb{1}\mathbb{1}XZ & X\mathbb{1}\mathbb{1}Z\mathbb{1} & X\mathbb{1}\mathbb{1}ZX & X\mathbb{1}\mathbb{1}ZZ \\
X\mathbb{1}X\mathbb{1}\mathbb{1} & X\mathbb{1}X\mathbb{1}X & X\mathbb{1}X\mathbb{1}Z & X\mathbb{1}XX\mathbb{1} & X\mathbb{1}XXX & X\mathbb{1}XXZ & X\mathbb{1}XZ\mathbb{1} & X\mathbb{1}XZX & X\mathbb{1}XZZ \\
X\mathbb{1}Z\mathbb{1}\mathbb{1} & X\mathbb{1}Z\mathbb{1}X & X\mathbb{1}Z\mathbb{1}Z & X\mathbb{1}ZX\mathbb{1} & X\mathbb{1}ZXX & X\mathbb{1}ZXZ & X\mathbb{1}ZZ\mathbb{1} & X\mathbb{1}ZZX & X\mathbb{1}ZZZ \\
XX\mathbb{1}\mathbb{1}\mathbb{1} & XX\mathbb{1}\mathbb{1}X & XX\mathbb{1}\mathbb{1}Z & XX\mathbb{1}X\mathbb{1} & XX\mathbb{1}XX & XX\mathbb{1}XZ & XX\mathbb{1}Z\mathbb{1} & XX\mathbb{1}ZX & XX\mathbb{1}ZZ \\
XXX\mathbb{1}\mathbb{1} & XXX\mathbb{1}X & XXX\mathbb{1}Z & XXXX\mathbb{1} & XXXXX & XXXXZ & XXXZ\mathbb{1} & XXXZX & XXXZZ \\
XXZ\mathbb{1}\mathbb{1} & XXZ\mathbb{1}X & XXZ\mathbb{1}Z & XXZX\mathbb{1} & XXZXX & XXZXZ & XXZZ\mathbb{1} & XXZZX & XXZZZ \\
XZ\mathbb{1}\mathbb{1}\mathbb{1} & XZ\mathbb{1}\mathbb{1}X & XZ\mathbb{1}\mathbb{1}Z & XZ\mathbb{1}X\mathbb{1} & XZ\mathbb{1}XX & XZ\mathbb{1}XZ & XZ\mathbb{1}Z\mathbb{1} & XZ\mathbb{1}ZX & XZ\mathbb{1}ZZ \\
XZX\mathbb{1}\mathbb{1} & XZX\mathbb{1}X & XZX\mathbb{1}Z & XZXX\mathbb{1} & XZXXX & XZXXZ & XZXZ\mathbb{1} & XZXZX & XZXZZ \\
XZZ\mathbb{1}\mathbb{1} & XZZ\mathbb{1}X & XZZ\mathbb{1}Z & XZZX\mathbb{1} & XZZXX & XZZXZ & XZZZ\mathbb{1} & XZZZX & XZZZZ \\
Z\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1} & Z\mathbb{1}\mathbb{1}\mathbb{1}X & Z\mathbb{1}\mathbb{1}\mathbb{1}Z & Z\mathbb{1}\mathbb{1}X\mathbb{1} & Z\mathbb{1}\mathbb{1}XX & Z\mathbb{1}\mathbb{1}XZ & Z\mathbb{1}\mathbb{1}Z\mathbb{1} & Z\mathbb{1}\mathbb{1}ZX & Z\mathbb{1}\mathbb{1}ZZ \\
Z\mathbb{1}X\mathbb{1}\mathbb{1} & Z\mathbb{1}X\mathbb{1}X & Z\mathbb{1}X\mathbb{1}Z & Z\mathbb{1}XX\mathbb{1} & Z\mathbb{1}XXX & Z\mathbb{1}XXZ & Z\mathbb{1}XZ\mathbb{1} & Z\mathbb{1}XZX & Z\mathbb{1}XZZ \\
Z\mathbb{1}Z\mathbb{1}\mathbb{1} & Z\mathbb{1}Z\mathbb{1}X & Z\mathbb{1}Z\mathbb{1}Z & Z\mathbb{1}ZX\mathbb{1} & Z\mathbb{1}ZXX & Z\mathbb{1}ZXZ & Z\mathbb{1}ZZ\mathbb{1} & Z\mathbb{1}ZZX & Z\mathbb{1}ZZZ \\
ZX\mathbb{1}\mathbb{1}\mathbb{1} & ZX\mathbb{1}\mathbb{1}X & ZX\mathbb{1}\mathbb{1}Z & ZX\mathbb{1}X\mathbb{1} & ZX\mathbb{1}XX & ZX\mathbb{1}XZ & ZX\mathbb{1}Z\mathbb{1} & ZX\mathbb{1}ZX & ZX\mathbb{1}ZZ \\
ZXX\mathbb{1}\mathbb{1} & ZXX\mathbb{1}X & ZXX\mathbb{1}Z & ZXXX\mathbb{1} & ZXXXX & ZXXXZ & ZXXZ\mathbb{1} & ZXXZX & ZXXZZ \\
ZXZ\mathbb{1}\mathbb{1} & ZXZ\mathbb{1}X & ZXZ\mathbb{1}Z & ZXZX\mathbb{1} & ZXZXX & ZXZXZ & ZXZZ\mathbb{1} & ZXZZX & ZXZZZ \\
ZZ\mathbb{1}\mathbb{1}\mathbb{1} & ZZ\mathbb{1}\mathbb{1}X & ZZ\mathbb{1}\mathbb{1}Z & ZZ\mathbb{1}X\mathbb{1} & ZZ\mathbb{1}XX & ZZ\mathbb{1}XZ & ZZ\mathbb{1}Z\mathbb{1} & ZZ\mathbb{1}ZX & ZZ\mathbb{1}ZZ \\
ZZX\mathbb{1}\mathbb{1} & ZZX\mathbb{1}X & ZZX\mathbb{1}Z & ZZXX\mathbb{1} & ZZXXX & ZZXXZ & ZZXZ\mathbb{1} & ZZXZX & ZZXZZ \\
ZZZ\mathbb{1}\mathbb{1} & ZZZ\mathbb{1}X & ZZZ\mathbb{1}Z & ZZZX\mathbb{1} & ZZZXX & ZZZXZ & ZZZZ\mathbb{1} & ZZZZX & ZZZZZ
\end{array}
$$

Next we determine which of these *potential* stabilizer elements anti-commute with each error operator $\mathcal{E}_\alpha \in \mathcal{E}$. Two operators anti-commute when $\{A, B\} = A \cdot B + B \cdot A = 0$. Error operators that anti-commute with any element of the stabilizer, correspond to errors that are *detectable* by the corresponding stabilizer code. However, to be *correctable*, each error operator needs to have a different pattern of anti-commutativity with the elements of the stabilizer. We can find the patterns of anti-commutativity by computer search very easily. The result will look something like:

$$
\begin{aligned}
\mathcal{E}_{\text{None}} &= \mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1} \text{ anti-commutes with } F_1 = \{\} \\
\mathcal{E}_{\text{B5}} &= \mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}X \text{ anti-commutes with } F_2 = \{\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}Z, \mathbb{1}\mathbb{1}\mathbb{1}XZ, \mathbb{1}\mathbb{1}\mathbb{1}ZZ, \mathbb{1}\mathbb{1}X\mathbb{1}Z, \ldots\} \\
\mathcal{E}_{\text{BP5}} &= \mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}(XZ) \text{ anti-commutes with } F_3 = \{\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}X, \mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}Z, \mathbb{1}\mathbb{1}\mathbb{1}XX, \mathbb{1}\mathbb{1}\mathbb{1}XZ, \ldots\} \\
\mathcal{E}_{\text{P5}} &= \mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}Z \text{ anti-commutes with } F_4 = \{\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}X, \mathbb{1}\mathbb{1}\mathbb{1}XX, \mathbb{1}\mathbb{1}\mathbb{1}ZX, \mathbb{1}\mathbb{1}X\mathbb{1}X, \ldots\} \\
&\vdots \\
\mathcal{E}_{\text{P1}} &= Z\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1} \text{ anti-commutes with } F_{15} = \{X\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}, X\mathbb{1}\mathbb{1}\mathbb{1}X, X\mathbb{1}\mathbb{1}\mathbb{1}Z, X\mathbb{1}\mathbb{1}X\mathbb{1}, \ldots\}
\end{aligned}
\tag{14.57}
$$

This gives us sets of tensor products of Pauli operators, $\{F_1, F_2, \ldots, F_{15}\}$, that anti-commute with the different error operators. The required stabilizer will then be a *minimum hitting set* of the sets $\{F_1, F_2, \ldots, F_{15}\}$. By minimum hitting set we mean that the desired stabilizer $\{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4\}$ is the smallest set that intersects with at least one element in every set $F_1, F_2, \ldots, F_{15}$. Given the results of a computer

search shown in (14.57) a minimal hitting set is $\mathcal{S}$ is found to require only four tensor products, $\{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4\}$, where:

$$\mathcal{S}_1 = X \otimes X \otimes Z \otimes X \otimes \mathbb{1}$$
$$\mathcal{S}_2 = X \otimes Z \otimes X \otimes \mathbb{1} \otimes X$$
$$\mathcal{S}_3 = Z \otimes \mathbb{1} \otimes X \otimes X \otimes Z$$
$$\mathcal{S}_4 = Z \otimes X \otimes \mathbb{1} \otimes Z \otimes X$$

which coincides with the stabilizer we picked to generate codewords that match those used in the 5-qubit Laflamme-Miquel-Paz-Zurek code.

The pattern of anti-commutativity between each error operator $\mathcal{E}_\alpha \in \mathcal{E}$ and the elements of the stabilizer, $\mathcal{S}_i \in \mathcal{S}$, is shown in Table 14.6. In the table a check mark signifies $\mathcal{E}_\alpha$ and $\mathcal{S}_i$ anti-commute whereas a cross signifies they do not. As you can see, each error operator anti-commutes with at least one element of the stabilizer. Moreover, the pattern of anti-commutativity is unique to each error operator. We can exploit this property to associate each error type with a different error syndrome.

So to sum up, we can either pick a stabilizer as a random finite Abelian sub-group of the Pauli group and then see whatever errors it protects against. Alternatively, we can fix the set of errors we want to protect against and use them to induce an

**Table 14.6** Two operators anti-commute when $\{A, B\} = A \cdot B + B \cdot A = 0$. Each error operator, $\mathcal{E}_{\text{None}}, \mathcal{E}_{\text{B5}}, \mathcal{E}_{\text{BP5}}, \ldots$ etc., describing a single error amongst five qubits, anti-commutes with at least one element of the stabilizer $\{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4\}$. Furthermore, the pattern of anti-commutativity is unique to each operator. This property can be exploited to associate each type of error with a unique error syndrome

| Error type | Error operator | $\{\mathcal{E}_\alpha, \mathcal{S}_1\} = 0?$ | $\{\mathcal{E}_\alpha, \mathcal{S}_2\} = 0?$ | $\{\mathcal{E}_\alpha, \mathcal{S}_3\} = 0?$ | $\{\mathcal{E}_\alpha, \mathcal{S}_4\} = 0?$ |
|---|---|---|---|---|---|
| $\mathcal{E}_{\text{None}}$ | $\mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1}$ | × | × | × | × |
| $\mathcal{E}_{\text{B5}}$ | $\mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes X$ | × | × | ✓ | × |
| $\mathcal{E}_{\text{BP5}}$ | $\mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes X \cdot Z$ | × | ✓ | ✓ | ✓ |
| $\mathcal{E}_{\text{P5}}$ | $\mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes Z$ | × | ✓ | × | ✓ |
| $\mathcal{E}_{\text{B4}}$ | $\mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes X \otimes \mathbb{1}$ | × | × | × | ✓ |
| $\mathcal{E}_{\text{BP4}}$ | $\mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes X \cdot Z \otimes \mathbb{1}$ | ✓ | × | ✓ | ✓ |
| $\mathcal{E}_{\text{P4}}$ | $\mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes Z \otimes \mathbb{1}$ | ✓ | × | ✓ | × |
| $\mathcal{E}_{\text{B3}}$ | $\mathbb{1} \otimes \mathbb{1} \otimes X \otimes \mathbb{1} \otimes \mathbb{1}$ | ✓ | × | × | × |
| $\mathcal{E}_{\text{BP3}}$ | $\mathbb{1} \otimes \mathbb{1} \otimes X \cdot Z \otimes \mathbb{1} \otimes \mathbb{1}$ | ✓ | ✓ | ✓ | × |
| $\mathcal{E}_{\text{P3}}$ | $\mathbb{1} \otimes \mathbb{1} \otimes Z \otimes \mathbb{1} \otimes \mathbb{1}$ | × | ✓ | ✓ | × |
| $\mathcal{E}_{\text{B2}}$ | $\mathbb{1} \otimes X \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1}$ | × | ✓ | × | × |
| $\mathcal{E}_{\text{BP2}}$ | $\mathbb{1} \otimes X \cdot Z \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1}$ | ✓ | ✓ | × | ✓ |
| $\mathcal{E}_{\text{P2}}$ | $\mathbb{1} \otimes Z \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1}$ | ✓ | × | × | ✓ |
| $\mathcal{E}_{\text{B1}}$ | $X \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1}$ | × | × | ✓ | ✓ |
| $\mathcal{E}_{\text{BP1}}$ | $X \cdot Z \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1}$ | ✓ | ✓ | ✓ | ✓ |
| $\mathcal{E}_{\text{P1}}$ | $Z \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1}$ | ✓ | ✓ | × | × |

acceptable stabilizer as the solution to a minimum hitting set problem. Given the stabilizer, the quantum codewords it stabilizes, and the errors it protects against, can be obtained automatically.

Next we see how the stabilizer formalism also simplifies the search for the required encoding and decoding circuits, and allows us to perform error correction while staying entirely within the encoded basis.

### *14.6.7 Stabilizer-Based Error Correction: The Encoding Step*

To protect a qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ we encode it into the state $|\psi\rangle_L = \alpha|0\rangle_L + \beta|1\rangle_L$ using the Laflamme-Miquel-Paz-Zurek encoding circuit shown in Fig. 14.3. Once in encoded form the logical qubit is protected against a single error amongst any of the five qubits.

### *14.6.8 Stabilizer-Based Error Correction: Introduction of the Error*

We model the introduction of an error on our encoded state as the application of one of the error operators, $\mathcal{E}_\alpha \in \mathcal{E}$, our stabilizer is know to correct.

### *14.6.9 Stabilizer-Based Error Correction: Error Diagnosis & Recovery*

We use the same quantum circuit to perform the actual error correction as we use to find the encoded basis states $|0\rangle_L$ and $|1\rangle_L$. As illustrated in Fig. 14.8, we imagine that we have an encoded state entering the circuit, which has been afflicted with an unknown error, $\mathcal{E}_\alpha \in \mathcal{E}$, where:

$$\mathcal{E} = \{\mathcal{E}_{\text{None}}, \mathcal{E}_{\text{B5}}, \mathcal{E}_{\text{P5}}, \mathcal{E}_{\text{BP5}}, \mathcal{E}_{\text{B4}}, \mathcal{E}_{\text{P4}}, \mathcal{E}_{\text{BP4}}, \mathcal{E}_{\text{B3}}, \mathcal{E}_{\text{P3}}, \mathcal{E}_{\text{BP3}}, \mathcal{E}_{\text{B2}}, \mathcal{E}_{\text{P2}}, \mathcal{E}_{\text{BP2}}, \mathcal{E}_{\text{B1}}, \mathcal{E}_{\text{P1}}, \mathcal{E}_{\text{BP1}}\} \tag{14.58}$$

which includes the possibly no error whatsoever. The circuit essentially measures the eigenvalue of each element of the stabilizer, $\{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4\}$ with respect to the incoming state $\mathcal{E}_\alpha|\psi\rangle_L$. If the stabilizer element commutes with the (unknown) error operator, the eigenvalue will be $+1$. But if the error-operator anti-commutes with the element of the stabilizer, the eigenvalue will be $-1$. Thus, the pattern of anti-commutativity can therefore be used as an error syndrome $a\ b\ c\ d$, which can diagnose that the error afflicting $|\psi\rangle_L$ is $\mathcal{E}_\alpha$ and hence the appropriate corrective action needed to restore the state is $\mathcal{E}_\alpha^{-1}$.

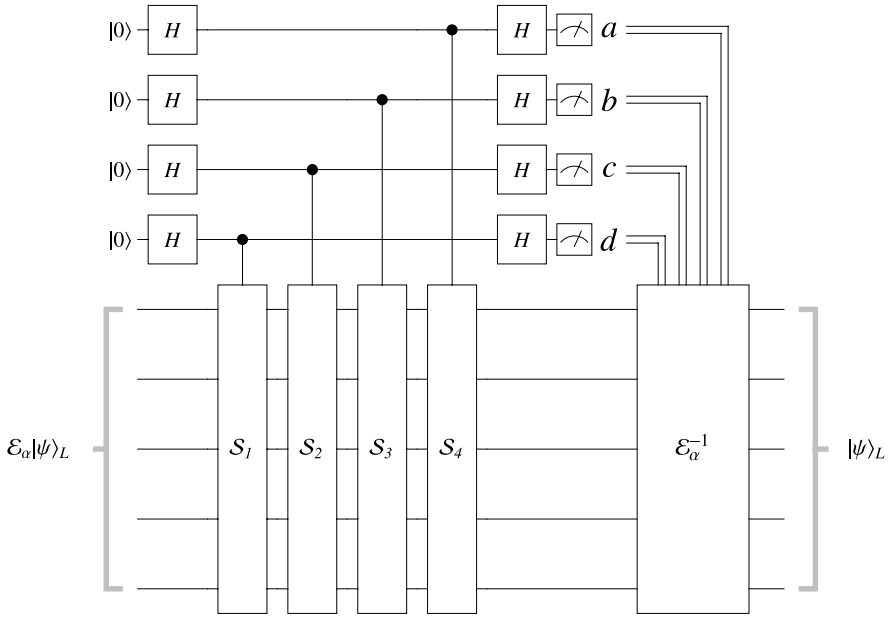


**Fig. 14.8** Quantum circuit for error recovery based on the stabilizer formalism. An error-afflicted encoded state, $\mathcal{E}_\alpha|\psi\rangle_L$, enters the circuit. At this point we have no idea what error has occurred, i.e., we do not know $\mathcal{E}_\alpha$. To discover the identity of $\mathcal{E}_\alpha$, we measure the eigenvalue of each element of the stabilizer with respect to the state fed into the circuit. If the error-afflicted state commutes with the element of the stabilizer, the eigenvalue is $+1$. If, on the other hand, the error-afflicted state anti-commutes with the element of the stabilizer, the eigenvalue is $-1$. Hence, the pattern of anti-commutativity revealed in the results $a\ b\ c\ d$, provides sufficient information diagnose what error occurred. That is, after these measurements we now know $\mathcal{E}_\alpha$. It is then straightforward to predict the error-restoration operation, $\mathcal{E}_\alpha^{-1}$, needed to restore the encoded qubit to its pristine, and still encoded, state $|\psi\rangle_L$. Note that during this error-diagnosis and error-correction process the single logical qubit remains in its encoded basis throughout. Hence, the stabilizer formalism is especially good because we never need to re-expose the logical qubit in an unprotected form at any time

Notice that, whereas in the original Laflamme-Miquel-Paz-Zurek scheme we periodically decoded the encoded state back to a single logical qubit, and thereby exposed it to an uncorrectable error, in the stabilizer formalism once the state has been encoded it is never re-exposed as a single logical qubit. Rather, in the stabilizer formalism, the whole error-correction procedure takes place within the encoded subspace. This is a very smart thing to do because it avoids having to periodically re-expose the logical qubit in order to error correct it.

### *14.6.10 Stabilizers for Other Codes*

An $[\![n, k, d]\!]$ quantum code (with square parentheses and a lowercase letter $k$) is a special notation for quantum stabilizer codes. Such as code uses $n$ physical qubits

to encode $k < n$ logical qubits within a $K = 2^k$-dimensional codespace and has minimum distance $d$. Hence, the number of 1-qubit changes needed to get from one codeword to another is at least $d$, which means that the code can correct up to $t = \lfloor \frac{d-1}{2} \rfloor$ single qubit errors.

The 9-qubit Shor, the 7-qubit Steane, and the 5-qubit Laflamme-Miquel-Paz-Zurek codes are respectively $[\![9, 1, 3]\!]$, $[\![7, 1, 3]\!]$, and $[\![5, 1, 3]\!]$ stabilizer codes, which can each correct at most $t = \lfloor (3-1)/2 \rfloor = 1$ error within their respective blocks of 9, 7, and 5 physical qubits.

A stabilizer for Shor's 9-qubit code is [141]:

$$\begin{aligned}
\mathcal{S}_1 &= ZZ\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1} \\
\mathcal{S}_2 &= Z\mathbb{1}Z\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1} \\
\mathcal{S}_3 &= \mathbb{1}\mathbb{1}\mathbb{1}ZZ\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1} \\
\mathcal{S}_4 &= \mathbb{1}\mathbb{1}\mathbb{1}Z\mathbb{1}Z\mathbb{1}\mathbb{1}\mathbb{1} \\
\mathcal{S}_5 &= \mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}ZZ\mathbb{1} \\
\mathcal{S}_6 &= \mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}\mathbb{1}Z\mathbb{1}Z \\
\mathcal{S}_7 &= XXXXXX\mathbb{1}\mathbb{1}\mathbb{1} \\
\mathcal{S}_8 &= XXX\mathbb{1}\mathbb{1}\mathbb{1}XXX
\end{aligned} \tag{14.59}$$

and one for Steane's 7-qubit code is [141]:

$$\begin{aligned}
\mathcal{S}_1 &= \mathbb{1}\mathbb{1}\mathbb{1}XXXX \\
\mathcal{S}_2 &= X\mathbb{1}X\mathbb{1}X\mathbb{1}X \\
\mathcal{S}_3 &= \mathbb{1}XX\mathbb{1}\mathbb{1}XX \\
\mathcal{S}_4 &= \mathbb{1}\mathbb{1}\mathbb{1}ZZZZ \\
\mathcal{S}_5 &= Z\mathbb{1}Z\mathbb{1}Z\mathbb{1}Z \\
\mathcal{S}_6 &= \mathbb{1}ZZ\mathbb{1}\mathbb{1}ZZ
\end{aligned} \tag{14.60}$$

Notice that the stabilizers for the Shor and Steane codes have only $X$'s or only $Z$'s within their respective stabilizer elements, making them so-called Calderbank-Shor-Steane ("CSS") codes. By comparison, the 5-qubit code is also a stabilizer code but it is not a CSS code because some of its stabilizer elements mix $Z$'s and $X$'s together.

## 14.7 Bounds on Quantum Error Correcting Codes

One can gain an intuition for the tradeoffs between the number of physical qubits ($n$), the number of logical qubits ($k$), and the maximum number of correctable errors $t = \lfloor \frac{d-1}{2} \rfloor$ by finding those tuples of values of $n$, $k$, and $d$ that simultaneously satisfy three important bounds on quantum codes: the quantum Hamming, Gilbert-Varshamov and Singleton bounds.

### 14.7.1 Quantum Hamming Bound

The quantum Hamming bound was discovered by Artur Ekert and Chiara Macchiavello [168]. It places an upper bound on the number of codewords we can have in a quantum code if the code has to be guaranteed to be able to encode $k$ logical qubits in $n$ physical qubits and protect against up to $t$ single qubit errors.

The bound is obtained via a counting argument on the number of buggy states in comparison to the number of states we can fit in a Hilbert space of dimension $2^n$. The argument goes as follows. If a code is to correct up to $t$ errors, then each codeword must be able to tolerate up to $t$ errors and yet still be distinct from every other codeword and every other potentially corrupted codeword. We can therefore imagine each codeword as being surrounded by a "cloud" of buggy states that have anywhere from zero to $t$ errors in them. All these states need to be distinct from the other buggy states in similar clouds around all the other codewords. All these buggy codewords have to fit within our Hilbert space of $n$ qubits.

Making this argument more quantitative, consider a single codeword of length $n$ qubits. We can introduce $i$ errors to this codeword by picking a particular subset of $i$ out of $n$ qubits, and assign single qubit errors to those qubits in all possible ways. There are $\binom{n}{i}$ ways to pick a particular subset of $i$ qubit locations, and there are three types of error ($X$, $Z$, and $(X \cdot Z)$) possible per location. Hence, there are $3^i\binom{n}{i}$ states describing $i$ errors to our codeword. But we want to protect against up to $t$ errors. Therefore, we can think of each codeword as being surrounded by a cloud of $\sum_{i=0}^{t} 3^i\binom{n}{i}$ "buggy" codewords. There are a total of $2^k$ such codewords. And the union of all these clouds of states needs to fit within the dimension of our $n$-qubit Hilbert space. Hence, we arrive at the quantum Hamming bound which places an upper bound on the number of codewords ($2^k$) or equivalently the number of logical qubits ($k$), that we can have in a quantum code that uses $n$ physical qubits. Hence, for a non-degenerate $[\![n, k, d]\!]$ code we must have:

$$2^k \sum_{i=0}^{t} 3^i \binom{n}{i} \leq 2^n \tag{14.61}$$

where $t = \lfloor \frac{d-1}{2} \rfloor$. Note that this bound gives a *necessary* condition for the existence of a quantum code and is really no more than a generalization of the argument we gave in Sect. 14.4.1 to deduce the allowed relationships between $n$, $k$ and $d$ for the 5-qubit Laflamme-Miquel-Paz-Zurek code.

### 14.7.2 Quantum Singleton Bound

The quantum Singleton bound was discovered by Raymond Laflamme and Manny Knill [292]. This states that if a pure or impure $[\![n, k, d]\!]$ code exists then:

$$n - k \geq 4\lfloor (d-1)/2 \rfloor \tag{14.62}$$

The quantum Singleton bound for pure codes was strengthened, slightly, by Calderbank, Rains, Shor, and Sloane [95] to:

$$n - k \geq 2(d-1) \tag{14.63}$$

This reduces to the Laflamme and Knill formula when $d$ is odd, but is slightly stronger when $d$ is even. It is also a necessary condition for the existence of a quantum code.

### 14.7.3 Quantum Gilbert-Varshamov Bound

The quantum Gilbert-Varshamov bound was discovered by Artur Ekert and Chiara Macchiavello [168]. It states that for an $[\![n, k, d]\!]$ code:

$$2^k \sum_{i=0}^{2t} 3^i \binom{n}{i} \geq 2^n \tag{14.64}$$

where the number of errors that can be corrected, $t$, is given by $t = \lfloor \frac{d-1}{2} \rfloor$. This bound gives a *sufficient* condition on the existence of a code but it is not necessary. The bound states that the number of codewords times the number of buggy codewords reachable in up to $2t$ errors must not be smaller than the dimension of the Hilbert space for $n$ physical qubits.

### 14.7.4 Predicting Upper and Lower Bounds on Additive Codes

The quantum Hamming, Singleton, and Gilbert-Varshamov bounds can be used to find upper and lower bounds on the minimum distance $d$ of feasible quantum codes. This in turn bounds the maximum possible number of errors such codes can correct, $t$, because we have $t = \lfloor \frac{d-1}{2} \rfloor$. Note that the quantum Hamming and Singleton bounds gives us an upper bound on $d$, whereas the quantum Gilbert-Varshamov bound gives us a (loose) lower bound on $d$. Nevertheless, the bounds are tight enough that we can use them to gain a rough intuition for the tradeoffs between the number of physical qubits, the number of logical qubits, the minimum distance and hence the maximum number of correctable errors.

Table 14.7 shows the approximate upper and lower bounds on the minimal distance $d$ in any $[\![n, k, d]\!]$ pure quantum error-correcting code as constrained by the quantum Hamming, quantum Singleton and quantum Gilbert-Varshamov bounds. The lower bounds are obtained from the quantum Gilbert-Varshamov inequality (14.64) and the upper bound is obtained from the lesser of the quantum Hamming (14.61) and quantum Singleton bound for pure codes (14.63). In Table 14.7 when a range of values is given these are lower and upper bounds on $d$. As the quantum Hamming and quantum Singleton bound provide a *necessary* condition on $d$,

**Table 14.7** Approximate bounds on the highest achievable minimal distance $d$ for any pure $[\![n, k, d]\!]$ quantum error-correcting code. The upper bound is obtained by finding the smallest value of distance $d$, for given values of $n$ and $k$, able satisfy the quantum Hamming bound (14.61) and the quantum Singleton bound (for pure codes) (14.63) *simultaneously*. These provide a *necessary* upper bound on $d$ for the existence of an $[\![n, k, d]\!]$ code. Hence the stated upper bound on $d$ is a hard constraint. The lower bound is obtained by finding the largest value of $d$, for given values of $n$ and $k$, able satisfy the quantum Gilbert-Varshamov bound. The latter bound is *sufficient* to guarantee the existence of an $[\![n, k, d]\!]$ code but it is not necessary. Hence, the lower bound given is loose

| $n \setminus k$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 3 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 3 | 3 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 3–4 | 3 | 2 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 3–4 | 3–4 | 3 | 2 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 3–4 | 3–4 | 3–4 | 3 | 2 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 5 | 3–4 | 3–4 | 3–4 | 3 | 2 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 5–6 | 5 | 3–4 | 3–4 | 3–4 | 3 | 2 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 5–6 | 5–6 | 3–4 | 3–4 | 3–4 | 3–4 | 2 | 2 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | 5–6 | 5–6 | 5–6 | 3–4 | 3–4 | 3–4 | 3–4 | 2 | 2 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | 5–6 | 5–6 | 5–6 | 5–6 | 3–4 | 3–4 | 3–4 | 3–4 | 2 | 2 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 14 | 5–8 | 5–6 | 5–6 | 5–6 | 5–6 | 3–4 | 3–4 | 3–4 | 3–4 | 2 | 2 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 5–8 | 5–8 | 5–6 | 5–6 | 5–6 | 5–6 | 3–4 | 3–4 | 3–4 | 3–4 | 2 | 2 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 16 | 5–8 | 5–8 | 5–8 | 5–6 | 5–6 | 5–6 | 3–4 | 3–4 | 3–4 | 3–4 | 3–4 | 2 | 2 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 5–8 | 5–8 | 5–8 | 5–6 | 5–6 | 5–6 | 5–6 | 3–4 | 3–4 | 3–4 | 3–4 | 3–4 | 2 | 2 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | 5–8 | 5–8 | 5–8 | 5–8 | 5–6 | 5–6 | 5–6 | 5–6 | 3–4 | 3–4 | 3–4 | 3–4 | 3–4 | 2 | 2 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 19 | 7–10 | 5–8 | 5–8 | 5–8 | 5–8 | 5–6 | 5–6 | 5–6 | 5–6 | 3–4 | 3–4 | 3–4 | 3–4 | 3–4 | 2 | 2 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 |
| 20 | 7–10 | 7–10 | 5–8 | 5–8 | 5–8 | 5–8 | 5–6 | 5–6 | 5–6 | 5–6 | 3–4 | 3–4 | 3–4 | 3–4 | 3–4 | 2 | 2 | 2 | 2 | 1 | 1 | 0 | 0 | 0 |

**Table 14.7** (Continued)

| $n \setminus k$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 21 | 7–10 | 7–10 | 7–10 | 5–8 | 5–8 | 5–8 | 5–6 | 5–6 | 5–6 | 5–6 | 5–6 | 3–4 | 3–4 | 3–4 | 3–4 | 3–4 | 2 | 2 | 2 | 2 | 1 | 1 | 0 | 0 |
| 22 | 7–10 | 7–10 | 7–10 | 5–8 | 5–8 | 5–8 | 5–8 | 5–6 | 5–6 | 5–6 | 5–6 | 3–4 | 3–4 | 3–4 | 3–4 | 3–4 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 0 |
| 23 | 7–10 | 7–10 | 7–10 | 7–10 | 5–8 | 5–8 | 5–8 | 5–8 | 5–6 | 5–6 | 5–6 | 5–6 | 3–4 | 3–4 | 3–4 | 3–4 | 3–4 | 2 | 2 | 2 | 2 | 2 | 1 | 1 |
| 24 | 7–12 | 7–10 | 7–10 | 7–10 | 7–10 | 5–8 | 5–8 | 5–8 | 5–8 | 5–6 | 5–6 | 5–6 | 5–6 | 3–4 | 3–4 | 3–4 | 3–4 | 3–4 | 2 | 2 | 2 | 2 | 2 | 1 |
| 25 | 7–12 | 7–12 | 7–10 | 7–10 | 7–10 | 5–8 | 5–8 | 5–8 | 5–8 | 5–8 | 5–6 | 5–6 | 5–6 | 5–6 | 3–4 | 3–4 | 3–4 | 3–4 | 3–4 | 2 | 2 | 2 | 2 | 2 |
| 26 | 7–12 | 7–12 | 7–10 | 7–10 | 7–10 | 7–10 | 5–8 | 5–8 | 5–8 | 5–8 | 5–6 | 5–6 | 5–6 | 5–6 | 5–6 | 3–4 | 3–4 | 3–4 | 3–4 | 3–4 | 2 | 2 | 2 | 2 |
| 27 | 7–12 | 7–12 | 7–12 | 7–10 | 7–10 | 7–10 | 7–10 | 5–8 | 5–8 | 5–8 | 5–8 | 5–6 | 5–6 | 5–6 | 5–6 | 5–6 | 3–4 | 3–4 | 3–4 | 3–4 | 3–4 | 2 | 2 | 2 |
| 28 | 7–12 | 7–12 | 7–12 | 7–12 | 7–10 | 7–10 | 7–10 | 7–10 | 5–8 | 5–8 | 5–8 | 5–8 | 5–6 | 5–6 | 5–6 | 5–6 | 5–6 | 3–4 | 3–4 | 3–4 | 3–4 | 3–4 | 2 | 2 |
| 29 | 9–14 | 7–12 | 7–12 | 7–12 | 7–12 | 7–10 | 7–10 | 7–10 | 7–10 | 5–8 | 5–8 | 5–8 | 5–8 | 5–6 | 5–6 | 5–6 | 5–6 | 5–6 | 3–4 | 3–4 | 3–4 | 3–4 | 3–4 | 2 |
| 30 | 9–14 | 9–14 | 7–12 | 7–12 | 7–12 | 7–10 | 7–10 | 7–10 | 7–10 | 5–8 | 5–8 | 5–8 | 5–8 | 5–8 | 5–6 | 5–6 | 5–6 | 5–6 | 5–6 | 3–4 | 3–4 | 3–4 | 3–4 | 3–4 |

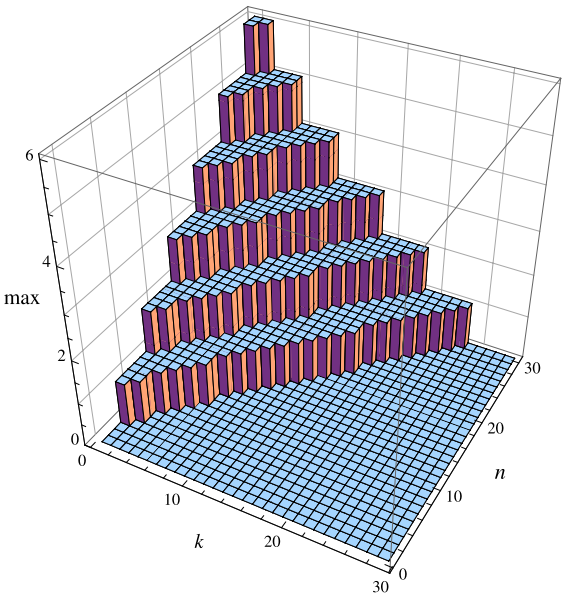


**Fig. 14.9** Plot of the approximate upper bounds on minimal distance $d$ of an $[\![n, k, d]\!]$ quantum error correcting code for $1 \leq n \leq 30$ and $0 \leq k \leq 28$. The data correspond to the upper bounds given in Table 14.7, which come from finding the largest value of $d$, for given values of $n$ and $k$, such that the quantum Hamming bound, and quantum Singleton bound are satisfied simultaneously. This is a *necessary* condition on the existence of the corresponding $[\![n, k, d]\!]$ code, so this upper bound on the minimal distance cannot be beaten

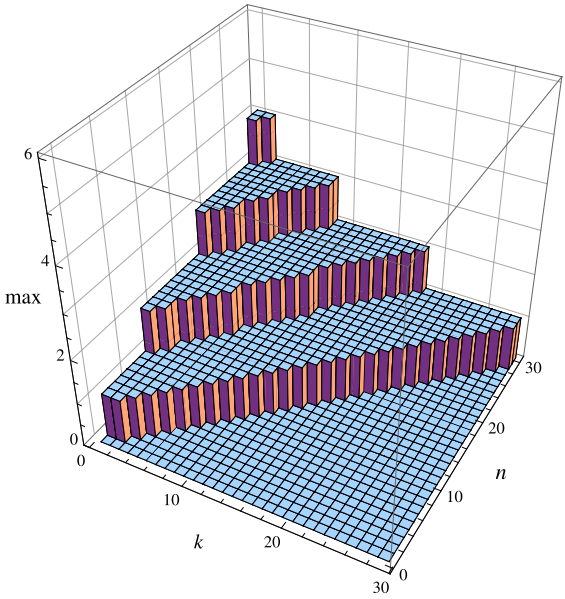


**Fig. 14.10** Plot of the approximate (very loose) lower bounds on minimal distance $d$ of an $[\![n, k, d]\!]$ quantum error correcting code for $1 \leq n \leq 30$ and $0 \leq k \leq 28$. The data correspond to the lower bounds given in Table 14.7, which come from finding the smallest value of $d$, for given values of $n$ and $k$, such that the quantum Gilbert-Varshamov bound is satisfied. This is only a *sufficient* condition on the existence of the corresponding $[\![n, k, d]\!]$ code, so this lower bound on the minimal distance can be beaten

whereas the quantum Gilbert-Varshamov bound provides a *sufficient* condition on $d$, the Hamming and Singleton bounds take precedence on upper bounding $d$.

The upper and lower bound data on predicted minimum distance in Table 14.7 is visualized in Figs. 14.9 and 14.10.

### 14.7.5 Tightest Proven Upper and Lower Bounds on Additive Codes

It is naturally to ask whether it is possible that more efficient codes could exist, i.e., codes that can correct more than one error per block. Indeed they can, but the complexity of the quantum circuits needed to implement them grows rapidly.

Using far more sophisticated methods, one can obtain tighter upper bounds, as well as proper lower bounds, on the highest achievable minimal distance $d$ of any $[\![n, k, d]\!]$ quantum error-correcting code (see Table 14.8). Comparing Table 14.8 with (the much more easily obtained) Table 14.7 shows the estimated bounds on $d$ from necessary and sufficient conditions are pretty good.

Plots of the tightest proven upper and lower bounds on minimum distance are shown in Figs. 14.11 and 14.12.

## 14.8 Non-additive (Non-stabilizer) Quantum Codes

The original 9-qubit Shor, 7-qubit Steane, and the 5-qubit Laflamme-Miquel-Paz-Zurek codes were all additive (stabilizer) codes. However, it is possible to have codes that possess a fundamentally different structure than the stabilizer codes. These so-called "non-additive" codes may be harder to find, but they are potentially more efficient than the stabilizer codes.

The first non-additive quantum error correcting code, that was provably better than an additive (stabilizer) code was the $(\!|5, 6, 2|\!)$ code discovered using numerical techniques by E.M. Rains, R.H. Hardin, P.W. Shor, and N.J.A. Sloane in 1997 [407]. This generalizes to a family of codes of the form $(\!|2n + 1, 3 \times 2^{2n-3}, 2|\!)$. Thomas Beth and Markus Grassl showed that the $(\!|5, 6, 2|\!)$ code could be obtained from union of additive codes [212]. That is, if $\mathcal{C}_1$ and $\mathcal{C}_2$ are respectively $(\!|n, K_1, d_1|\!)$ and $(\!|n, K_2, d_2|\!)$ quantum codes, the union of these codes is an $(\!|n, K_1 + K_2, \min(d_1, d_2)|\!)$ quantum code such that the set of errors the new code can correct is the intersection of the sets of errors the old codes could correct. Note that, whereas the dimension for the codespace of an additive code is always a power of two, the dimension of the codespace of a non-additive code built from the union of two additive codes need not be a power of two.

The first non-additive code found that can outperform the *optimal* $[\![5, 1, 3]\!]$ (Laflamme-Miquel-Paz-Zurek) stabilizer code was the $(\!|9, 12, 3|\!)$ non-additive code found by Sixia Yu, Qing Chen, C. Lai, and C. Oh [554].

## 14.9 Fault-Tolerant Quantum Error Correcting Codes

The discussions of quantum error correcting codes given in the preceding sections have made implicit assumptions about where and when errors occur. For example,

**Table 14.8** Known bounds on the highest achievable minimal distance $d$ for any $[\![n, k, d]\!]$ quantum error-correcting code [211]

| $n \setminus k$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 2 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 2 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 3 | 3 | 2 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 4 | 3 | 2 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 3 | 3 | 2 | 2 | 2 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 4 | 3 | 3 | 3 | 2 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 4 | 3 | 3 | 3 | 2 | 2 | 2 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 4 | 4 | 4 | 3 | 3 | 2 | 2 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 5 | 5 | 4 | 3 | 3 | 3 | 2 | 2 | 2 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | 6 | 5 | 4 | 4 | 4 | 3 | 3 | 2 | 2 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | 5 | 5 | 4 | 4 | 4 | 3–4 | 3 | 3 | 2 | 2 | 2 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 14 | 6 | 5 | 5 | 4–5 | 4 | 4 | 4 | 3 | 3 | 2 | 2 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 6 | 5 | 5 | 5 | 4 | 4 | 4 | 3 | 3 | 3 | 2 | 2 | 2 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 16 | 6 | 6 | 6 | 5 | 5 | 4–5 | 4 | 4 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 7 | 7 | 6 | 5–6 | 5 | 4–5 | 4–5 | 4 | 4 | 4 | 3 | 3 | 2 | 2 | 2 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | 8 | 7 | 6 | 5–6 | 5–6 | 5 | 5 | 4 | 4 | 4 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 19 | 7 | 7 | 6 | 5–6 | 5–6 | 5–6 | 5 | 4–5 | 4 | 4 | 3–4 | 3 | 3 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 20 | 8 | 7 | 6–7 | 6–7 | 6 | 5–6 | 5–6 | 4–5 | 4–5 | 4 | 4 | 3–4 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 0 | 0 | 0 |

**Table 14.8** (Continued)

| $n \setminus k$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 21 | 8 | 7 | 6–7 | 6–7 | 6–7 | 6 | 5–6 | 5–6 | 4–5 | 4–5 | 4 | 4 | 3–4 | 3 | 3 | 3 | 2 | 2 | 2 | 1 | 1 | 1 | 0 | 0 |
| 22 | 8 | 7–8 | 6–8 | 6–7 | 6–7 | 6–7 | 5–6 | 5–6 | 5–6 | 4–5 | 4–5 | 4 | 4 | 3–4 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 0 |
| 23 | 8–9 | 7–9 | 7–8 | 6–8 | 6–7 | 6–7 | 5–7 | 5–6 | 5–6 | 4–6 | 4–5 | 4–5 | 4 | 4 | 3–4 | 3 | 3 | 2 | 2 | 2 | 2 | 1 | 1 | 1 |
| 24 | 8–10 | 8–9 | 7–8 | 7–8 | 6–8 | 6–7 | 6–7 | 5–7 | 5–6 | 5–6 | 5–6 | 4–5 | 4–5 | 4 | 4 | 3–4 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 1 |
| 25 | 8–9 | 9 | 7–8 | 7–8 | 7–8 | 7–8 | 6–7 | 5–7 | 5–7 | 5–6 | 5–6 | 4–6 | 4–5 | 4–5 | 4 | 4 | 3–4 | 3 | 3 | 2 | 2 | 2 | 2 | 1 |
| 26 | 8–10 | 9 | 8–9 | 8–9 | 8 | 7–8 | 6–8 | 6–8 | 6–7 | 5–7 | 5–6 | 5–6 | 5–6 | 4–5 | 4–5 | 4 | 4 | 3–4 | 3 | 3 | 2 | 2 | 2 | 2 |
| 27 | 9–10 | 9 | 9 | 9 | 8–9 | 7–8 | 6–8 | 6–8 | 6–8 | 6–7 | 5–7 | 5–6 | 5–6 | 5 | 4–5 | 4–5 | 4 | 4 | 3–4 | 3 | 3 | 2 | 2 | 2 |
| 28 | 10 | 10 | 10 | 9 | 8–9 | 7–9 | 6–8 | 6–8 | 6–8 | 6–8 | 6–7 | 5–7 | 5–6 | 5–6 | 5–6 | 4–5 | 4 | 4 | 4 | 3–4 | 3 | 3 | 2 | 2 |
| 29 | 11 | 11 | 10 | 9–10 | 8–9 | 7–9 | 7–9 | 6–8 | 6–8 | 6–8 | 6–7 | 5–7 | 5–6 | 5–6 | 5–6 | 4–5 | 4–5 | 4 | 4 | 4 | 3–4 | 3 | 3 | 2 |
| 30 | 12 | 11 | 10 | 9–10 | 8–10 | 8–9 | 7–9 | 7–9 | 7–8 | 6–8 | 6–8 | 6–7 | 6–7 | 5–6 | 5–6 | 5–6 | 5 | 4–5 | 4 | 4 | 4 | 3–4 | 3 | 3 |

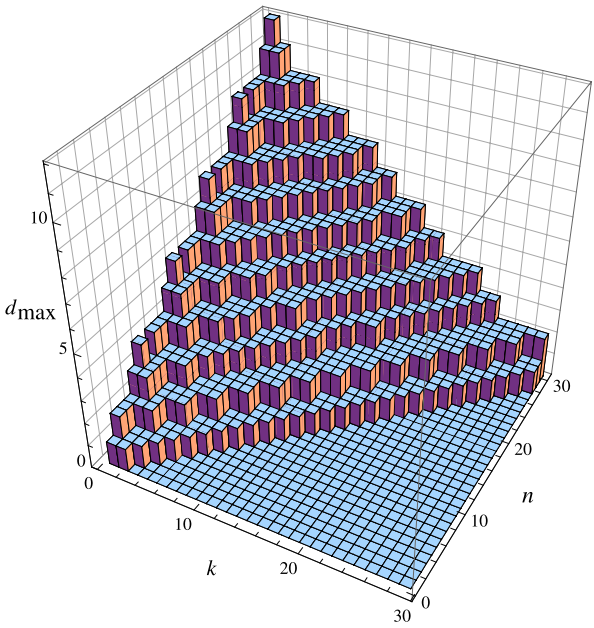


**Fig. 14.11** Plot of the known upper bounds on minimal distance $d$ of an $[\![n, k, d]\!]$ quantum error correcting code for $1 \leq n \leq 30$ and $0 \leq k \leq 28$. The data correspond to the upper bounds given in Table 14.8, which come from Markus Grassl's curated database of code parameters [211]

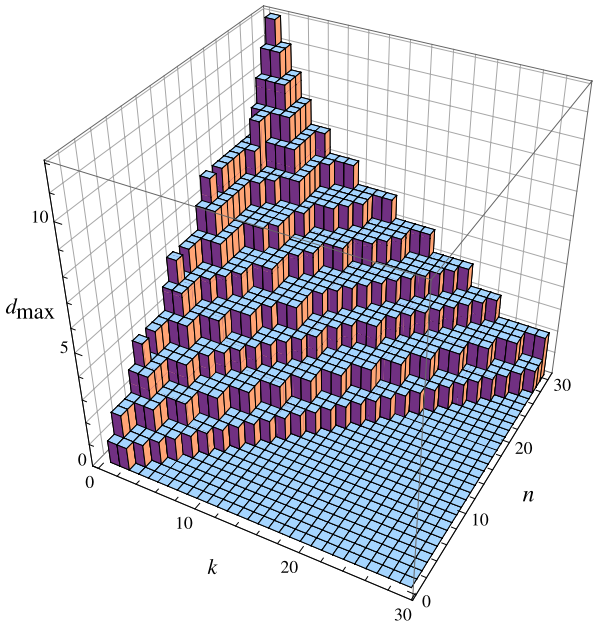


**Fig. 14.12** Plot of the known lower bounds on minimal distance $d$ of an $[\![n, k, d]\!]$ quantum error correcting code for $1 \leq n \leq 30$ and $0 \leq k \leq 28$. The data correspond to the lower bounds given in Table 14.8, which come from Markus Grassl's curated database of code parameters [211]

in the original formulation of the 5-qubit quantum error-correcting code, given in Sect. 14.4, error correction required us to map the encoded (protected) logical qubit back to its unprotected form periodically. Once error-free, the logical qubit would be

re-encoded to protect it again. If the error occurs when the qubit is back in its logical (unprotected) state the logical qubit will be vulnerable to irreversible corruption. For this strategy to work, we must implicitly assume that no errors can arise while the qubit is re-exposed in the unencoded basis. If this assumption holds, we will be able to store the state of a logical qubit indefinitely without error. Unfortunately, such an assumption is clearly unjustified by the physics of the situation. There is no good reason to expect errors should only afflict encoded qubits. However, by using the stabilizer formulation of the 5-qubit code, you will remember that error correction can be performed entirely within the encoded basis, never needing to re-expose the logical qubit to potential uncorrectable errors. Nevertheless, this is still not yet a complete solution, because we don't just want to protect quantum information when in storage, but also during quantum computation itself. This means that we need perform gate operations directly on the encoded data.

The second assumption we need to question is where do the errors occur? So far, we have implicitly assumed that the gates implementing the error correction operations are perfect. But what happens if they are imperfect? Can we error correct a quantum computation using imperfect quantum gates?

These concerns prompted further research into quantum error correcting codes that revealed how to them work even when the underlying error correction hardware is itself imperfect. The result is so-called *fault-tolerant* quantum error correction [209, 270, 400, 457].

A quantum circuit is deemed "fault-tolerant" when it can be made to output the correct result even though errors arise during its operation. John Preskill of the California Institute of Technology has identified five principles of quantum circuit design, distilled from Peter Shor's original paper on fault-tolerant quantum computation [457], which will make for fault-tolerant quantum circuits [399].

1. *Don't use the same ancilla qubit twice*. The intuition behind this principle is that if an ancilla qubit becomes corrupted, we want to limit the damage it can do by limiting the number of other gate operations that rely on the same ancilla. Thus, examples of good and bad quantum circuit structures that use ancillae are shown in Fig. 14.13. Error propagation in quantum circuits is much more problematic than in classical circuits because in controlled quantum gates errors can propagate in both directions, i.e., from control qubits to targets (as happens classically) and from target

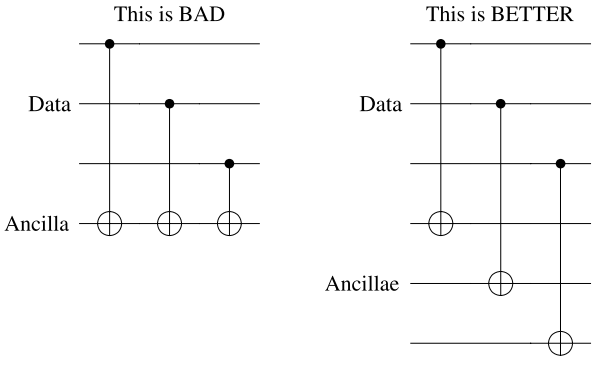


**Fig. 14.13** The first principle of fault-tolerant quantum computing: "do not use the same ancilla twice." This suppresses correlated error propagation from bad ancillae

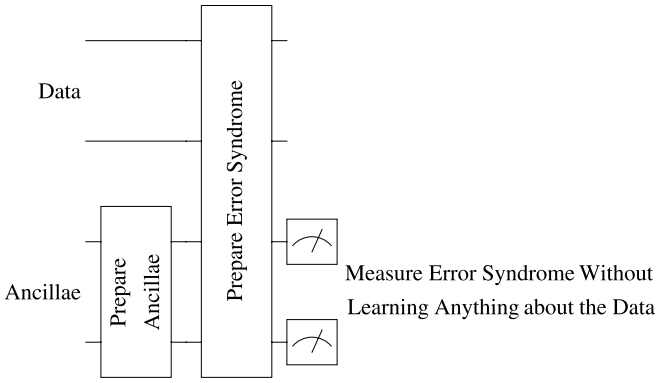


**Fig. 14.14** The second principle of fault-tolerant quantum computing: "Copy the errors not the data." The ancillae measurements must not extract any information about the logical state of the qubits being protected, only the errors that have afflicted them

qubits to controls (which does not happen classically). So quantum controlled gates are especially susceptible to the spread of error.

2. *Error syndrome measurements should reveal the error but not the data*. We need to be careful to prepare the ancilla qubit in such a way that when we measure the ancilla to obtain an error-syndrome we do not learn anything about the state we seek to protect, but only an error that may have afflicted it. This requires us to prepare the ancilla in a special entangled state prior to linking it to the state we wish to protect. A diagram of this is shown in Fig. 14.14.

3. *Verify when encoding a known quantum state.* The potential for corruption is greatest when qubits are exposed in their raw state before they been protected using some quantum error-correcting encoding. However, whenever we do know the complete description of the quantum state with which we are dealing, and we do know the operation we intend to perform on it, we have an opportunity to verify that we synthesized the correct state before using it further. This situation can arise, e.g, when we start off with some ancillae qubits in a known state, and we entangle them in some prescribed way. In such circumstances it is worth taking the time to verify the entangled state is correct before making use of it in subsequent quantum computations. For example, if our intent is to encode three physical qubits $|0\rangle|0\rangle|0\rangle$ into some encoded block of three qubits, which we will call $|\bar{0}\rangle$, we might perform a test to convince ourselves that we synthesized the block $|\bar{0}\rangle$ correctly before using it in subsequent quantum computations. This idea is illustrated in Fig. 14.15.

4. *Repeat operations*. Figure 14.15 also illustrates a fourth principle of fault-tolerant quantum computation. Just because we have verified the encoding of a state once does not mean necessarily that it is correct as our error syndrome measurement could be faulty. It would be just as disastrous to correct an error, or non-error, in the wrong way as it would to miss an error in the first place. However, by *repeating* measurements, we can increase our confidence that the error syndrome is actually what we think it is. Thus, repeating quantum measurements to a good habit as illustrated in Fig. 14.15.

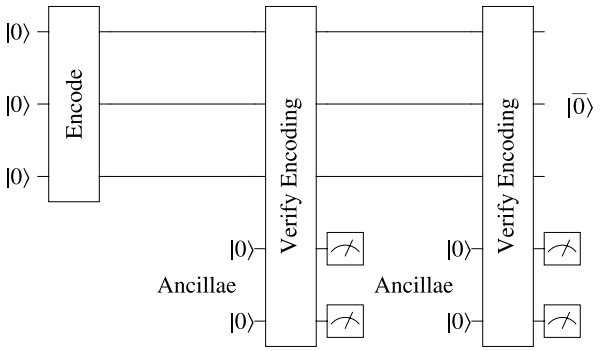


**Fig. 14.15** The third principle of fault-tolerant quantum computing: "Verify when encoding a known quantum state." A known state should be verified (perhaps repeatedly) before being deemed fit for use

5. *Stay in the encoded basis*. The 9-qubit, 7-qubit and 5-qubit quantum codes we described above are, as presented thus far, geared towards error-correcting qubits while they are inactive, i.e., merely stored in memory. However, typically, we want to do more than merely *store* qubits—we want to *compute* with them. That is, we anticipate needing to apply quantum logic gates in order to perform a purposeful quantum computation. However, the theory of quantum error correcting codes outlined above, does not describe how to perform quantum gates on the encoded qubits. Instead, when one wants to perform a quantum gate, one would need to map the encoded qubits back to the logical basis, apply the quantum gate, and then re-encode the result back into the encoded basis. Such a strategy is at least cumbersome, and worse, periodically exposes the qubits to corruption as the quantum gates are being applied. To circumvent this problem Wojciech Zurek and Raymond Laflamme, and Peter Shor independently devised a schemes for performing quantum gate operations on the encoded qubits *directly*, without removing them from the safety of the encoded basis [457, 566].

Obeying these five principles of fault-tolerant quantum circuit design will help to ensure that a quantum computer will operate reliably.

### *14.9.1 Concatenated Codes and the Threshold Theorem*

So far, we have seen that quantum error codes are possible in principle, and codes that can correct up an arbitrary number of errors exist. Moreover, we have seen it is possible to use such codes in an intelligent way by employing a fault-tolerant architecture. Unfortunately, there is still a problem. Although it is indeed possible to devise more complex quantum codes that can correct up to $t$ errors in a block, the complexity of the quantum circuits needed to implement such codes rises rapidly. In fact, before long, we have to use so many gates that the probability of making an error within the error correcting circuitry becomes higher than the probability of the original error. So merely increasing the code complexity to correct for more errors per block is not necessarily the best way to improve reliability.

Julia Kempe has provided the following intuitive analysis of the tradeoffs between how many errors a quantum error correcting code can correct and the complexity of its required quantum circuitry [270]. If the original probability of failure per gate operation or per measurement is $\epsilon$ then in the $t$-error resilient code the failure rate would change to $\epsilon^{t+1}$, which is good. But the price we pay is that the number of gates needed in the error correction circuitry grows too, typically as some polynomial in $t$, $t^a$, with $a > 1$. So overall, the probability of having $t + 1$ errors occur before error correction has completed grows as $(t^a\epsilon)^{t+1}$. This expression is minimized when $t = c\epsilon^{-\frac{1}{a}}$, for some constant $c$, and the value of the failure probability is then $p_{\text{fail}} \geq \exp(-ca\epsilon^{-\frac{1}{a}})$. If we repeat $t$-error resilient error correction $N$ times, the failure probability will therefore become $Np_{\text{fail}} = N\exp(-ca\epsilon^{-\frac{1}{a}}) = \exp(-ca(\log N)\epsilon^{-\frac{1}{a}})$. For this overall failure probability to be much less than 1, we will therefore need $\epsilon$ to scale as $1/(\log N)^a$. In other words, the longer the computation, the smaller $\epsilon$ needs to be. Unfortunately, this is not practical. Instead we need an error correction scheme that allows the error probability per gate operation to be held constant whilst allowing longer and longer computations are performed reliably.

An alternative way to improve the reliability is to *concatenate* the simpler quantum codes we know about [7, 11]. The idea is that each logical qubit is encoded in $n$ physical qubits (to make a "level-1" encoding), and each of these $n$ physical qubits are themselves encoded in $n$ other physical qubits (to make a "level-2" encoding), and so on. The number of levels of concatenation can be chosen so as to achieve any desired probability in the correctness of the final result. Figure 14.16 shows a schematic illustrating the basic idea.

It is fairly involved to calculate the exact effects of concatenation on the overall reliability of the circuit, although people have done so for different physical schemes and quantum computer architectures [31, 291, 350, 479, 484, 486]. In part, this is

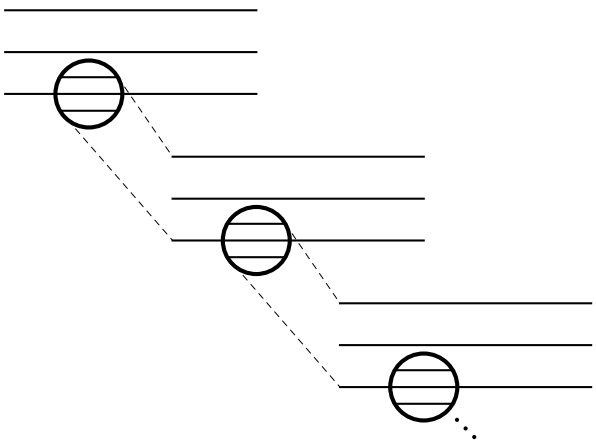

**Fig. 14.16** Schematic view of concatenated coding: each qubit is encoded in several qubits, which are each encoded in several qubits, which are each encoded in several qubits, etc.

because the details of the calculation depend upon many factors such as the actual code used, the error model assumed, the degree to which fault-tolerant design principles have been followed, the architectural assumptions made, and the extent to which opportunities for gate-parallelism have been exploited. However, a simple back-of-the-envelope argument is sufficient to convey the main idea, that the use of concatenation is beneficial provided the error probability per qubit per gate is less than a certain threshold.

Think of it this way. Suppose we are using a code that encodes each logical qubit in $n$ physical qubits. The quantum error-correcting codes we looked at earlier can correct a single arbitrary error (bit-flip, phase-flip, or joint bit-flip and phase-flip) in a coding block. So for the logical qubit to be in error at the end of some error-correcting cycle we will have had to have had *two* or more errors introduced into a block. If the probability of an error per qubit per gate operation is $p$, then (since all the 1-qubit errors are fixable), by using the code the error probability becomes $p_{\text{fail}}^{\text{level}-1} = cp^2$ where $c$ counts the number of ways pairs of errors can be inserted amongst the physical qubits in a coding block.

Now concatenate this process. For each physical qubit in the coding block, imagine using the same code to encode it into $n$ more physical qubits. Now $n^2$ physical qubits are involved in encoding one logical qubit. What will it take for our logical qubit to be in error now? Let us call this the level-2 encoding. We have $p_{\text{fail}}^{\text{level}-2} = c(p_{\text{fail}}^{\text{level}-1})^2 = c^3 p^4$.

Repeating concatenation steps in this fashion we can write down the error probability as a function of the number of levels of concatenation as follows:

$$\begin{aligned} p_{\text{fail}}^{\text{level}-1} &= cp^2 \\ p_{\text{fail}}^{\text{level}-2} &= c(p_{\text{fail}}^{\text{level}-1})^2 = c^3 p^4 \\ p_{\text{fail}}^{\text{level}-3} &= c(p_{\text{fail}}^{\text{level}-2})^2 = c^7 p^8 \\ &\vdots \\ p_{\text{fail}}^{\text{level}-k} &= c(p_{\text{fail}}^{\text{level}-(k-1)})^2 = \frac{c^{2^k} p^{2^k}}{c} \end{aligned} \tag{14.65}$$

Thus, successive levels of concatenation will tend to suppress the error in the logical qubit provided $c\,p < 1$, where $p$ is the probability of error per physical qubit per gate operation. Hence, there is a *threshold* in error probability of:

$$p < p_{\text{threshold}} \equiv \frac{1}{c} \tag{14.66}$$

in which case the error will be reduced with successive levels of concatenation. Hence, provided this error probability per qubit per gate threshold is met, it will be possible to implement quantum computations of arbitrary length to arbitrary accuracy. That is, one can quantum compute forever without error!

But what is the overhead in gate count we have to pay to achieve $k$ levels of concatenation? Again following Julia Kempe's intuitive argument [270], if the circuit

we wish to implement has $N$ gates when done without error correction, and we desire a final success probability of order $1 - p$, then in such a circuit each gate has to have a failure probability of less than or equal to $p/N$ because errors compound. Hence, if we concatenate $k$ times we will require:

$$p_{\text{fail}}^{\text{level-}k} = \frac{c^{2^k} p^{2^k}}{c} = p_{\text{threshold}} \left( \frac{p}{p_{\text{threshold}}} \right)^{2^k} \leq \frac{p}{N} \tag{14.67}$$

which implies

$$2^k \leq \frac{\log(N\epsilon_{\text{th}}/p)}{\log(\epsilon_{\text{th}}/\epsilon)} \tag{14.68}$$

So after $k$ levels of concatenation, each gate turns into $G^k$ gates where:

$$G^k = 2^{k \log G \leq (\frac{\log(N\epsilon_{\text{th}}/p)}{\log(\epsilon_{\text{th}}/\epsilon)}) \log G} = \text{poly}(\log N) \tag{14.69}$$

and so its final size will be $N \,\text{poly}(\log N)$, which is only polylogarithmically larger.

Opinions as to actual values of this error threshold have varied widely over the years. Initially, error rates per gate of around $10^{-4}$–$10^{-7}$ were thought necessary, but the threshold has steadily been climbing [31, 291, 350, 400, 479, 484, 486]. The truth is, although we have talked about *the* threshold, in reality it is not unique: one can obtain different thresholds if one specializes the theory to different quantum computer architectures. Such considerations take into account the specific characteristics of different physical embodiments of quantum information processing wherein some error mechanisms are more prevalent than others. If one does this, one can obtain different assessments of the error rate per gate operation needed to sustain quantum computations of arbitrary length. Some recent studies suggest in certain architectures and schemes, the threshold could be as high as 3% [291].

## 14.10 Errors as Allies: Noise-Assisted Quantum Computing

We end this chapter with an observation. The prevailing opinion of quantum computer scientists is that quantum error correction is essential to achieving a useful quantum computer. However, is this necessarily true? For certain computations, such as factoring composite integers, where we seek an exact solution that is either plainly right or plainly wrong, we are indeed obliged to imbue our quantum computations with the ability to either avoid errors (e.g., using decoherence-free or topological encodings we shall describe in Chap. 15) or undo errors (e.g., using quantum error correcting codes). But there are many other computations in which we seek not a right or wrong answer, but instead, a "pretty good" answer. For example, in a maximum satisfiability problem, an ideal solution is one that satisfies the greatest number of constraints. However, in practice, we might be content with a solution that comes close to this ideal but not quite. In this case, the pragmatic measure of whether a quantum computer is better than a classical computer, is whether

it finds an equally good, i.e., equally sub-optimal, solution in less time, or whether it finds a better, albeit still sub-optimal, solution in the same time as required by a classical computer. Given the relative importance and ubiquity of such problems in comparison to integer factorization, a greater degree of investigation is warranted.

Moreover, surprisingly, there are a handful of results that suggest that noise, dissipation and decoherence can sometimes be an ally of quantum computation! For example, noise can be harnessed productively in entangled state preparation [342, 551], to effect quantum gate operations [39–41], and to enhance quantum transport in networks including, e.g., the light harvesting structures in plants [105, 359, 394, 413, 414]. It seems worthwhile to pursue such avenues to determine whether there is a strategy for quantum computation that makes noise a friend rather than an enemy.

## 14.11 Summary

There is an inherent contradiction amongst the ideal requirements for a quantum computing device. On the one hand the machine needs to be well isolated from the external world to permit it to evolve unitarily while executing some desired quantum computation. On the other hand, the machine needs to be strongly coupled to the external world to allow us to initialize it in an arbitrary starting state, or command it to perform a particular sequence of unitary gate operations. Switching the interaction with the external world on and off cleanly is extremely challenging experimentally. Hence errors are likely to arise in real quantum computing hardware.

In this chapter we have looked at several approaches to dealing with errors in quantum computations. We found that it is not as easy to detect an error in a quantum computation as it is in a classical computation because errors may exist along a continuum of possibilities and our ability and we are not even allowed to read a corrupted state directly, because such direct observations would make matters worse rather than better.

In the early days of quantum computing it was felt that such obstacles appeared to preclude the possibility of error correcting codes for quantum information. However, it turns out that quantum error correcting codes are possible. The trick is to entangle the qubit whose state we want to protect (the logical qubit) with several other physical qubits (i.e., ancillae) in such a manner that subsequent measurements on the ancillae qubits will reveal what error has afflicted the encoded data, and hence the corrective action needed to restore the logical qubit to its correct state. Crucially, these measurements on the ancillae only reveal information about the error and nothing about the state we wish to protect. Once the error is known it can be undone using the appropriate inverse unitary operation.

Various families of quantum codes are now known. We can estimate the tradeoffs different codes make regarding the number of logical qubits protected, the number of physical qubits into which they are encoded, and the maximum number of errors that can be corrected by way of the quantum Hamming, Singleton and Gilbert-Varsharmov bounds. Sometimes tighter bounds have now been determined for many codes using more sophisticated methods. A database of known results is maintained

by Markus Grassl. The best code able to protect a logical qubit against a single bit-flip, phase-flip, or joint bit flip and phase flip, is the Laflamme-Miquel-Paz-Zurek 5-qubit code. This code saturates the quantum Hamming bound and is optimal. We gave complete circuits for the encoding and decoding stages of the 5-qubit code. We also showed the codewords used for less efficient codes that were discovered before the 5-qubit code.

In our original formulation of the 5-qubit code, the encoded qubit had to be mapped back to the unencoded (logical) basis periodically in order for he the error correction to be performed. This exposes the logical qubit to uncorrectable errors while it is back in the unencoded basis. We described how the stabilizer formalism can combat this by performing error correction while staying entirely within the encoded basis.

An obvious issue with quantum error correction is that the error-correcting circuitry may itself introduce more errors. For quantum error correction to be truly viable, we need to be able to use imperfect error correction to achieve perfect computation. Fortunately, through a combination of fault-tolerant circuit design principles, and the use of concatenated coding, we showed that coding schemes can be devised that, in principle, permit error-correctable quantum computations of arbitrary length. We showed that to achieve such concatenated coding schemes the error probability per qubit per gate operation needs to be below a critical threshold. This threshold is sensitive to the error model, architecture, and physical embodiment used. However, schemes now exist that suggest error rates as high as 3% might be tolerable.

Two relatively new directions for handling errors in quantum computing are the use of noise sources as an ally in quantum computation, and the use of decoherence-free subspaces and topological quantum effects to make quantum hardware that is immune to errors. We shall examine such topics in the next chapter in the context of alternative models of quantum computation.

## 14.12 Exercises

**14.1** Prove that any $2 \times 2$ matrix can be written as a weighted sum of Pauli matrices according to:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \frac{a+d}{2}\mathbb{1} + \frac{b+c}{2}X + \frac{i\,(b-c)}{2}Y + \frac{a-d}{2}Z \tag{14.70}$$

See Sect. 2.4.1.1 for a definition of the Pauli matrices.

**14.2** Write down the operators that describe the following errors afflicting a 5-qubit state.

(a) A bit-flip on the first qubit.
(b) A phase-flip and the fifth qubit.

(c) A joint bit-flip and phase-flip and the third qubit.
(d) A bit flip on the second qubit and phase-flip on the fourth qubit.

**14.3** Consider a single logical qubit in a state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ that interacts with an environment describable using just two qubits—a simplification indeed. Equation (14.31) says that the joint state of the qubit and its environment evolve as follows:

$$\begin{aligned} U|\psi\rangle|E\rangle = (\alpha|0\rangle + \beta|1\rangle) \otimes \frac{|E_{00}\rangle + |E_{11}\rangle}{2} \quad &\text{(no error)} \\ + (\alpha|0\rangle - \beta|1\rangle) \otimes \frac{|E_{00}\rangle - |E_{11}\rangle}{2} \quad &\text{(phase flip)} \\ + (\alpha|1\rangle + \beta|0\rangle) \otimes \frac{|E_{01}\rangle + |E_{10}\rangle}{2} \quad &\text{(bit flip)} \\ + (\alpha|1\rangle - \beta|0\rangle) \otimes \frac{|E_{01}\rangle - |E_{10}\rangle}{2} \quad &\text{(joint phase flip \& bit flip)} \end{aligned}$$

Assuming the states $|E_{00}\rangle$, $|E_{01}\rangle$, $|E_{10}\rangle$, and $|E_{11}\rangle$ are orthonormal:

(a) Prove that the states of the environment $\frac{|E_{00}\rangle+|E_{11}\rangle}{2}$, $\frac{|E_{00}\rangle-|E_{11}\rangle}{2}$, $\frac{|E_{01}\rangle+|E_{10}\rangle}{2}$, and $\frac{|E_{01}\rangle-|E_{10}\rangle}{2}$ are orthonormal. What is the significance of this in terms of error detection?
(b) Prove that the three qubit state $U|\psi\rangle|E\rangle$ is entangled? What is the significance of this in terms of error determination?

**14.4** The quantum circuit that encodes a single logical qubit in state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ within a 5-qubit entangled state according to Braunstein and Smolin's version of the Laflamme-Miquel-Paz-Zurek code is show in Fig. 14.3.

(a) Use the encoding circuit together with the fact that $L = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$ to compute the unitary matrix corresponding to the Laflamme-Miquel-Paz-Zurek encoding circuit.
(b) Verify that the circuit acting on input $|\psi\rangle|0000\rangle$ produces the state $|\psi\rangle_L = \alpha|0\rangle_L + \beta|1\rangle_L$ where the quantum codewords $|0\rangle_L$ and $|1\rangle_L$ are as given by (14.40).
(c) Write down the state that results from a joint bit-flip and phase-flip on the fourth qubit in the encoded state.
(d) Compute the unitary matrix corresponding to the Laflamme-Miquel-Paz-Zurek decoding circuit. Note, this is the inverse of the encoding circuit.
(e) Use the decoding matrix of part (d) to error-correct the error-afflicted state of part (c). What is the resulting state?

**14.5** The quantum circuit that encodes a single logical qubit in state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ within a 5-qubit entangled state according to Braunstein and Smolin's version of the Laflamme-Miquel-Paz-Zurek code is show in Fig. 14.3. This circuit protects

against a single general error afflicting any of the five qubits in the encoded state. However, to know a single general error has occurred we have to measure the error syndrome. Prior to such measurements no error has yet occurred. Hence, so long as we are not looking, multiple single qubit errors can afflict our state, but which one is actually realized only becomes definite after we measure the error syndrome! This is one of the amazing facts of quantum mechanics. This exercise will help you appreciate this subtlety of quantum error correction:

(a) Determine the unitary matrix for Braunstein and Smolin's version of the Laflamme-Miquel-Paz-Zurek code.
(b) Use the unitary matrix of part (a) to compute the encoded form of the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, i.e., $|\psi\rangle_L$.
(c) Define two representative single qubit error operators as follows: Let $\mathcal{E}_{\mathrm{B1}}$ be the operator representing a bit-flip on the first qubit in the encoded state, and let $\mathcal{E}_{\mathrm{P3}}$ be the operator representing a phase shift on the third qubit in the encoded state. Compute the state that results when *both* errors afflict the encoded qubit equally.
(d) How many errors have afflicted the encoded qubit at this point? [Think carefully before you answer].
(e) Now run the buggy state through the Laflamme-Miquel-Paz-Zurek decoding circuit. What are the possible values for the error syndrome you can obtain?
(f) How does the measurement of the error syndrome affect the state of the unmeasured (top) qubit?
(g) Is it fair to say that the Laflamme-Miquel-Paz-Zurek code can correct multiple errors? [Think carefully before you answer].

**14.6** In the Laflamme-Miquel-Paz-Zurek 5-qubit code we allow joint bit flip and phase flip errors to afflict a given qubit. However, does it make a difference whether the bit flip or phase flip occurs first? To investigate this, answer the following questions:

(a) Show that a bit flip followed by a phase flip yields a strictly different result from a phase flip followed by a bit flip.
(b) Show, however, that the error syndrome corresponding to a bit flip followed by a phase flip on qubit $i$ is the same as the error syndrome corresponding to a phase flip followed by a bit flip on qubit $i$.
(c) Even though the error syndromes are the same, are the corrective actions needed to restore the buggy qubit to its original state, the same? If not, does this means that we cannot really error correct the qubit because we cannot distinguish between a bit flip followed by a phase flip from a phase flip followed by a bit flip?
(d) Is there any measurement you could do that would tell you whether you had restored a qubit to its original state $|\psi\rangle$ or whether you had restored it to $-|\psi\rangle$?

**14.7** Prove that the asymptotic form for the quantum Hamming bound (14.61) is given by:

$$\frac{k}{n} \leq \left(1 - \frac{t}{n}\log_2 3 - H\left(\frac{t}{n}\right)\right) \tag{14.71}$$

where $H$ is the entropy $H(x) = -x\log_2 x - (1-x)\log_2(1-x)$.

**14.8** Prove that the asymptotic form for the quantum Gilbert-Varshamov bound (14.64) is given by:

$$\frac{k}{n} \geq \left(1 - \frac{2t}{n}\log_2 3 - H\left(\frac{2t}{n}\right)\right) \tag{14.72}$$

where $H$ is the entropy $H(x) = -x\log_2 x - (1-x)\log_2(1-x)$.

**14.9** Which of the following $[\![n, k, d]\!]$ codes are ruled out by the quantum Hamming, quantum Gilbert-Varshamov, or quantum Singleton bounds?

(a) A $[\![5, 2, 3]\!]$ code
(b) A $[\![7, 2, 3]\!]$ code
(c) A $[\![10, 1, 5]\!]$ code
(d) A $[\![10, 5, 3]\!]$ code
(e) A $[\![19, 1, 7]\!]$ code
(f) A $[\![20, 1, 11]\!]$ code

**14.1 Problem** What is the "minimum length", i.e., minimum value of $n$, of a $k = 1$ quantum error-correcting code that corrects $t = 1, 2, 3, 4, 5, 6, 7, 8, 9$ errors?