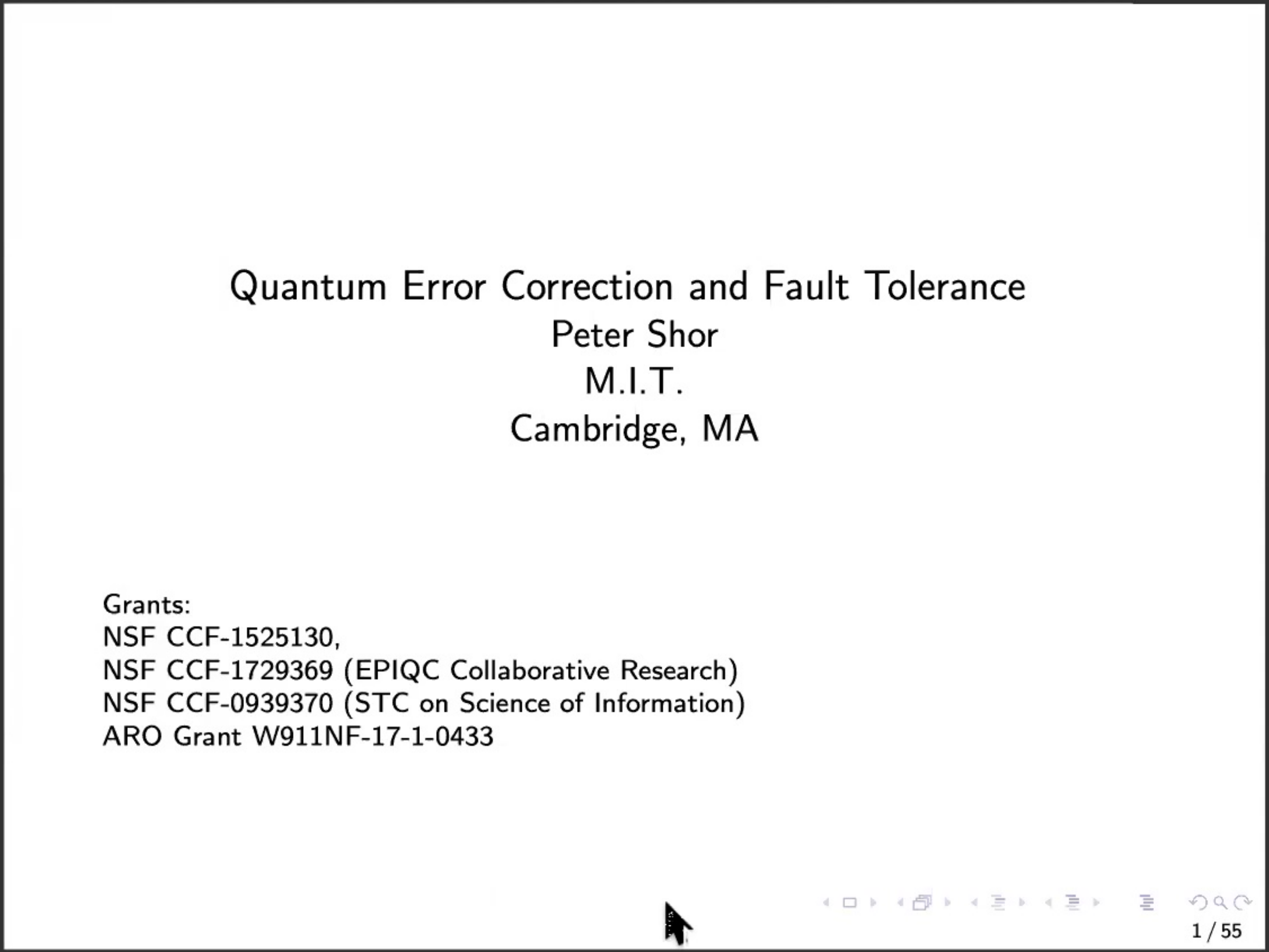
Abstract: Quantum computers need to manipulate quantum states, and the only ways we know of doing this are inherently noisy. Thus, if we are ever to be able to do long computations on quantum computers, we need to make them fault tolerant. The way we know how to do this currently is to use quantum error correcting codes. We will introduce error correcting codes and explain how they can be used to provide fault tolerance for quantum computers.
&nbsp;

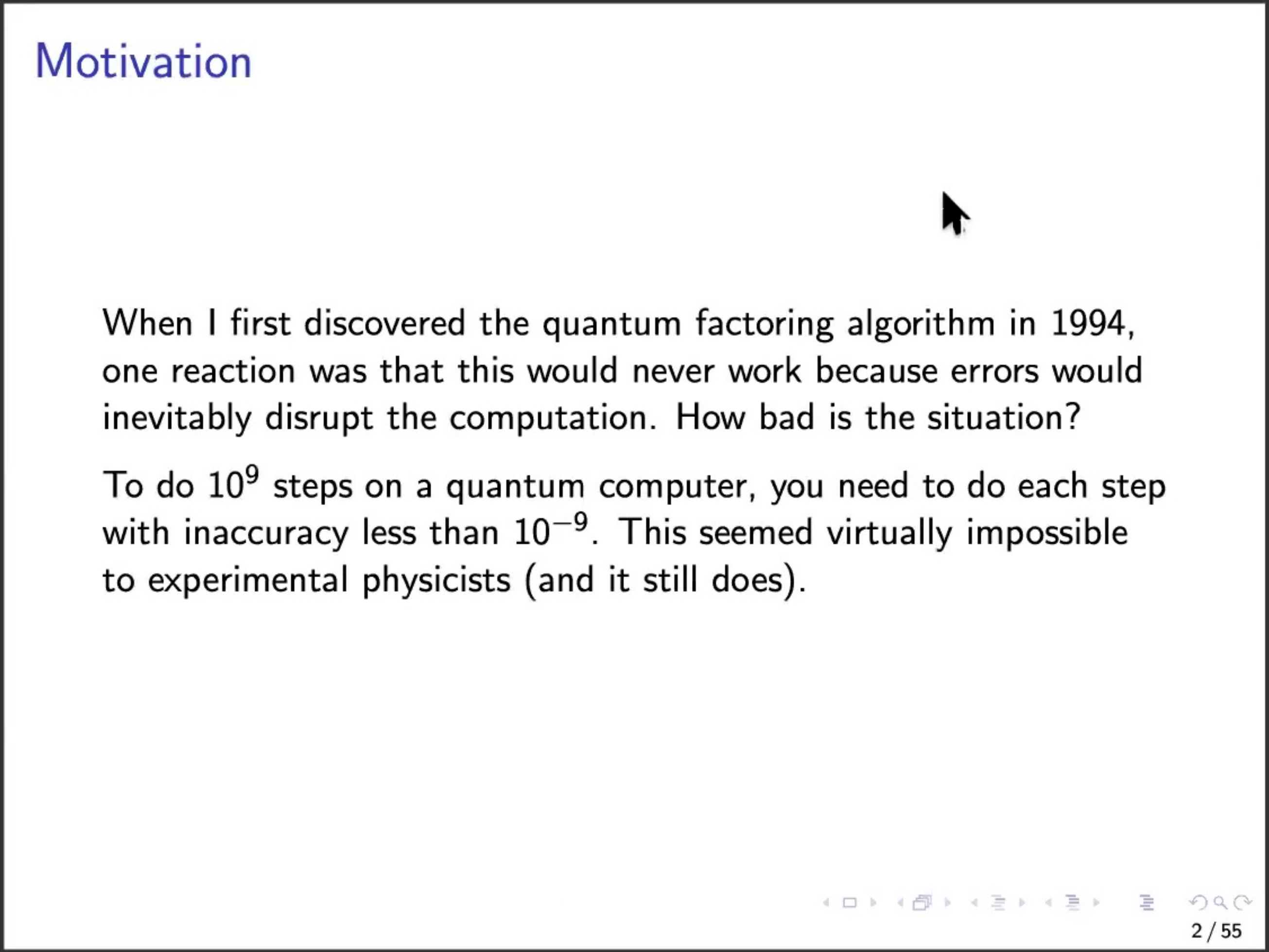


Quantum Error Correction and Fault Tolerance

Peter Shor
M.I.T.
Cambridge, MA

Grants:
NSF CCF-1525130,
NSF CCF-1729369 (EPIQC Collaborative Research)
NSF CCF-0939370 (STC on Science of Information)
ARO Grant W911NF-17-1-0433

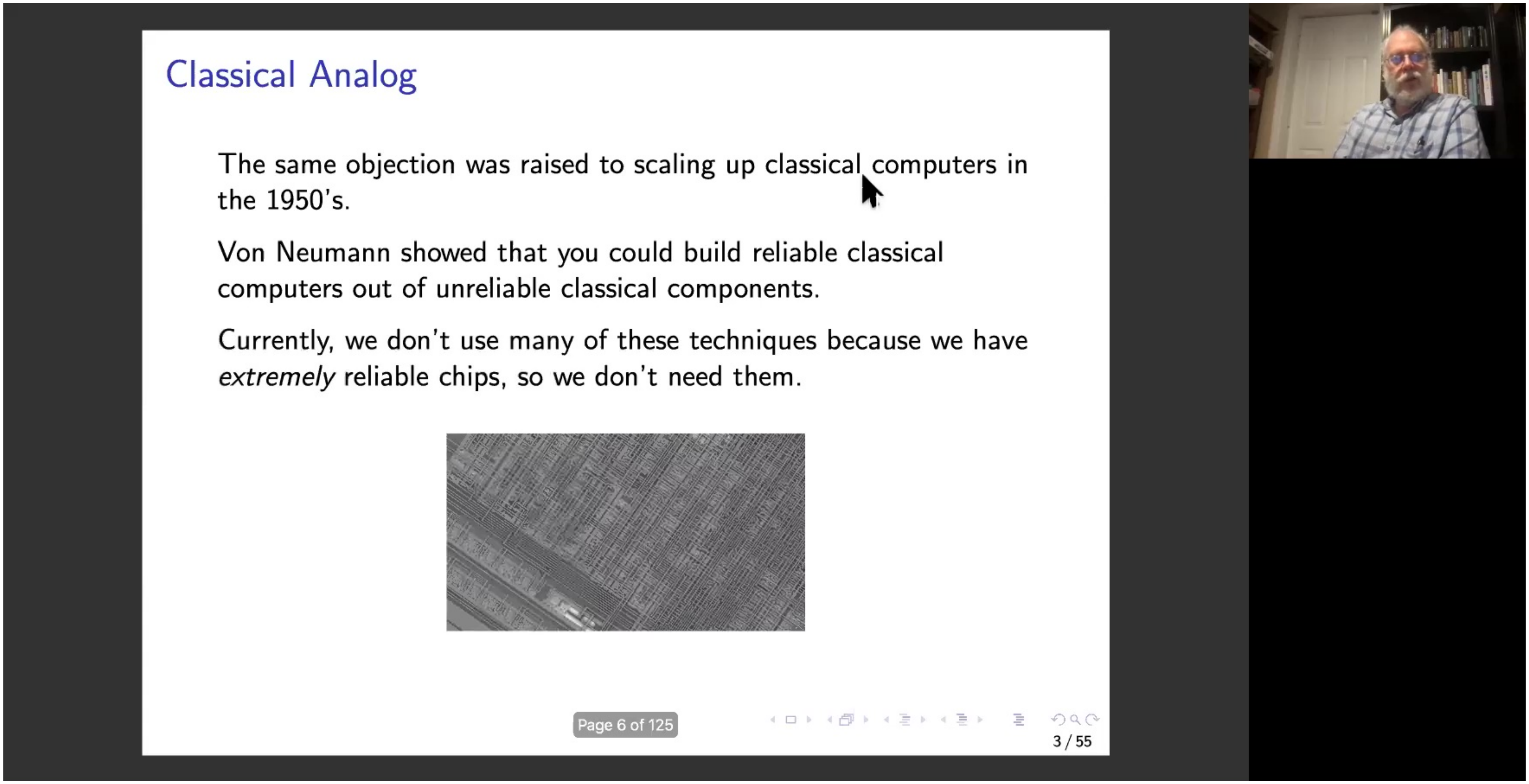
1 / 55



Motivation

When I first discovered the quantum factoring algorithm in 1994, one reaction was that this would never work because errors would inevitably disrupt the computation. How bad is the situation?

To do 10^9 steps on a quantum computer, you need to do each step with inaccuracy less than 10^{-9}. This seemed virtually impossible to experimental physicists (and it still does).

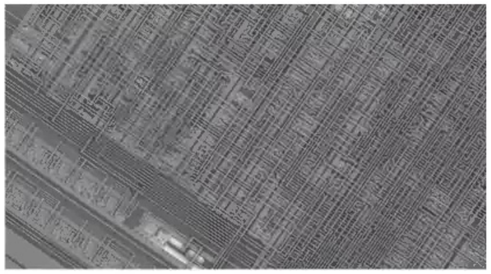


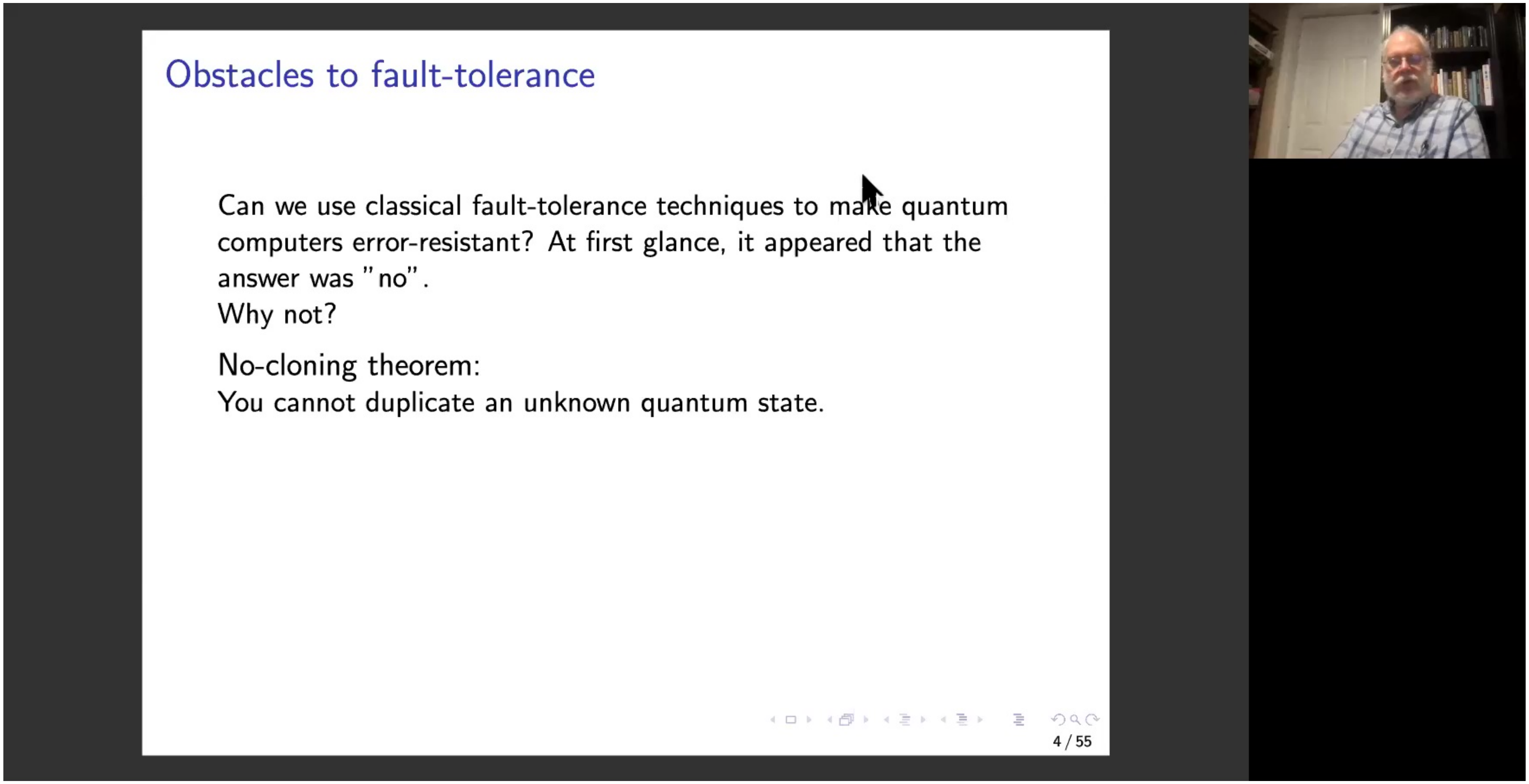
Classical Analog

The same objection was raised to scaling up classical computers in the 1950's.

Von Neumann showed that you could build reliable classical computers out of unreliable classical components.

Currently, we don't use many of these techniques because we have *extremely* reliable chips, so we don't need them.



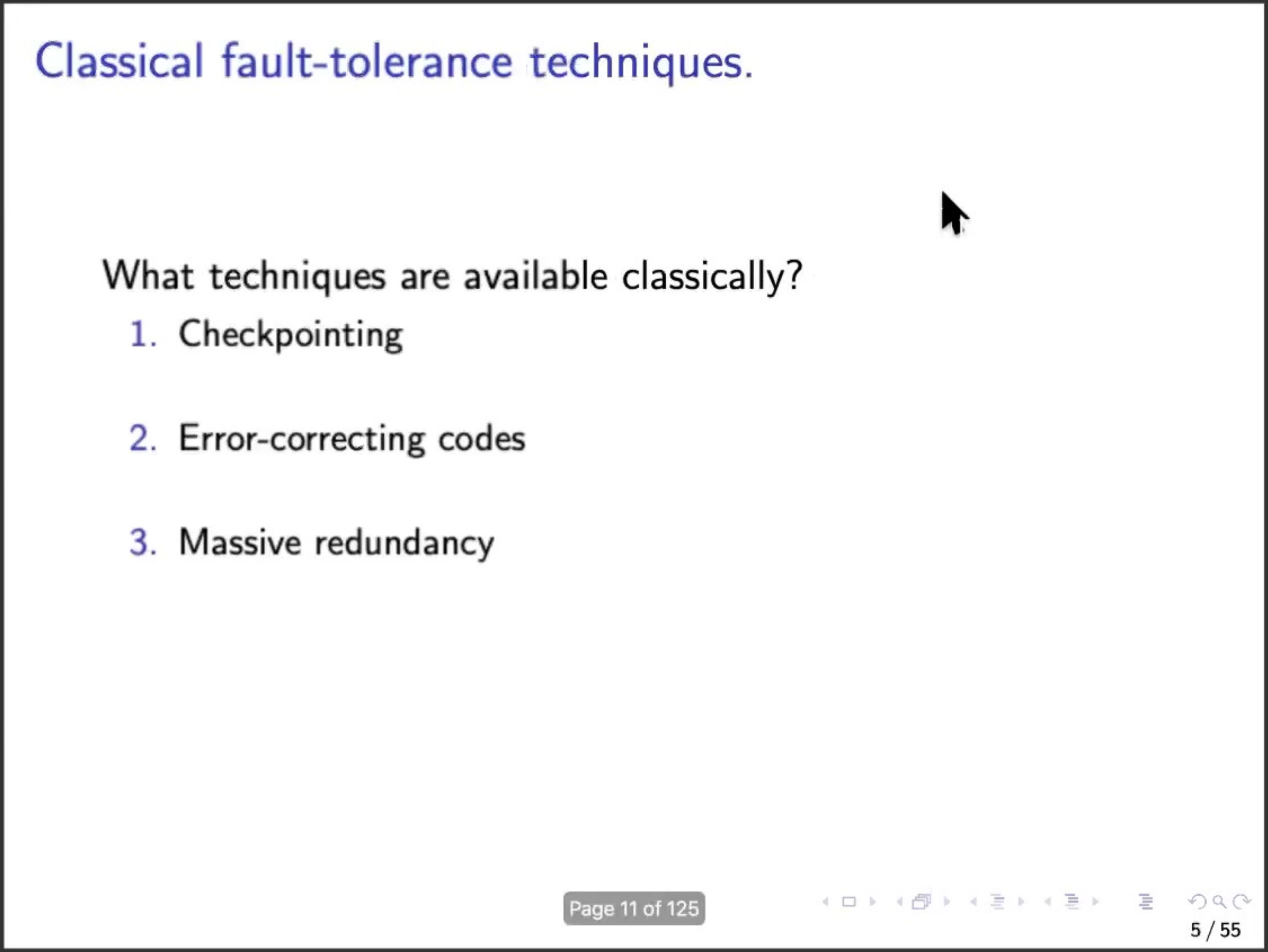


Obstacles to fault-tolerance

Can we use classical fault-tolerance techniques to make quantum computers error-resistant? At first glance, it appeared that the answer was "no".
Why not?

No-cloning theorem:
You cannot duplicate an unknown quantum state.

4 / 55



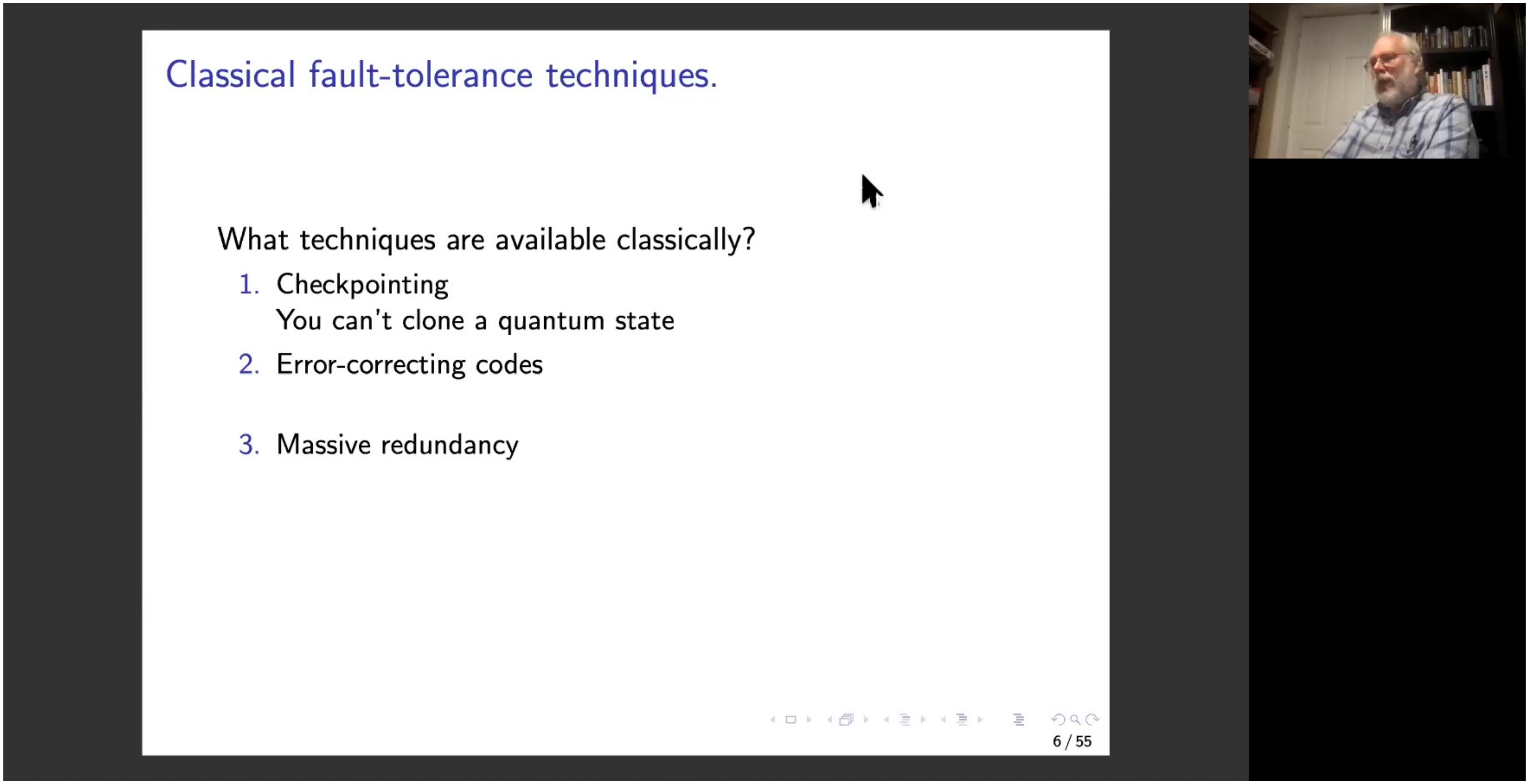
Classical fault-tolerance techniques.

What techniques are available classically?

1. Checkpointing
2. Error-correcting codes
3. Massive redundancy

Page 11 of 125

5 / 55



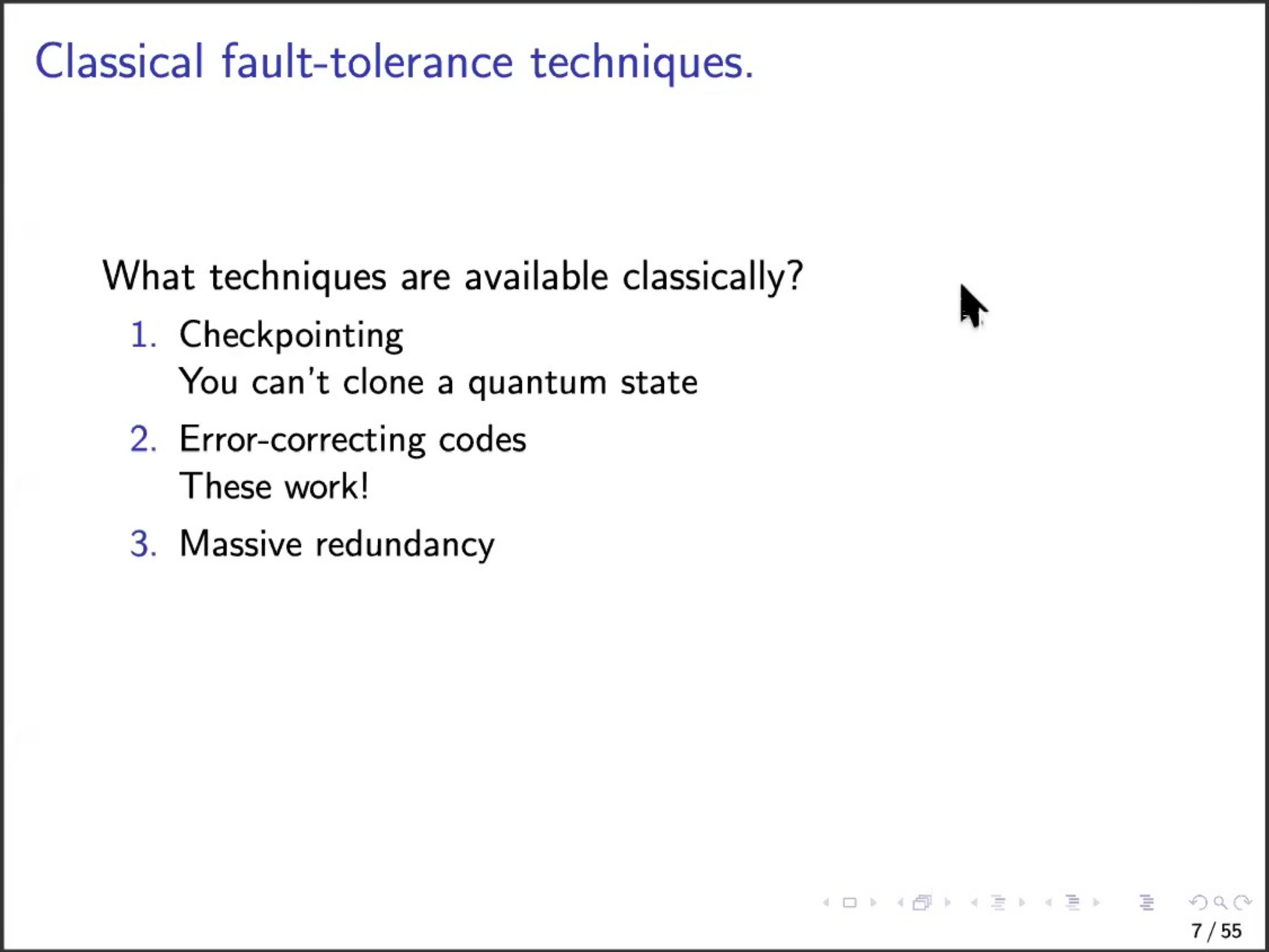
Classical fault-tolerance techniques.

What techniques are available classically?

1. Checkpointing
   You can't clone a quantum state
2. Error-correcting codes

3. Massive redundancy

6 / 55

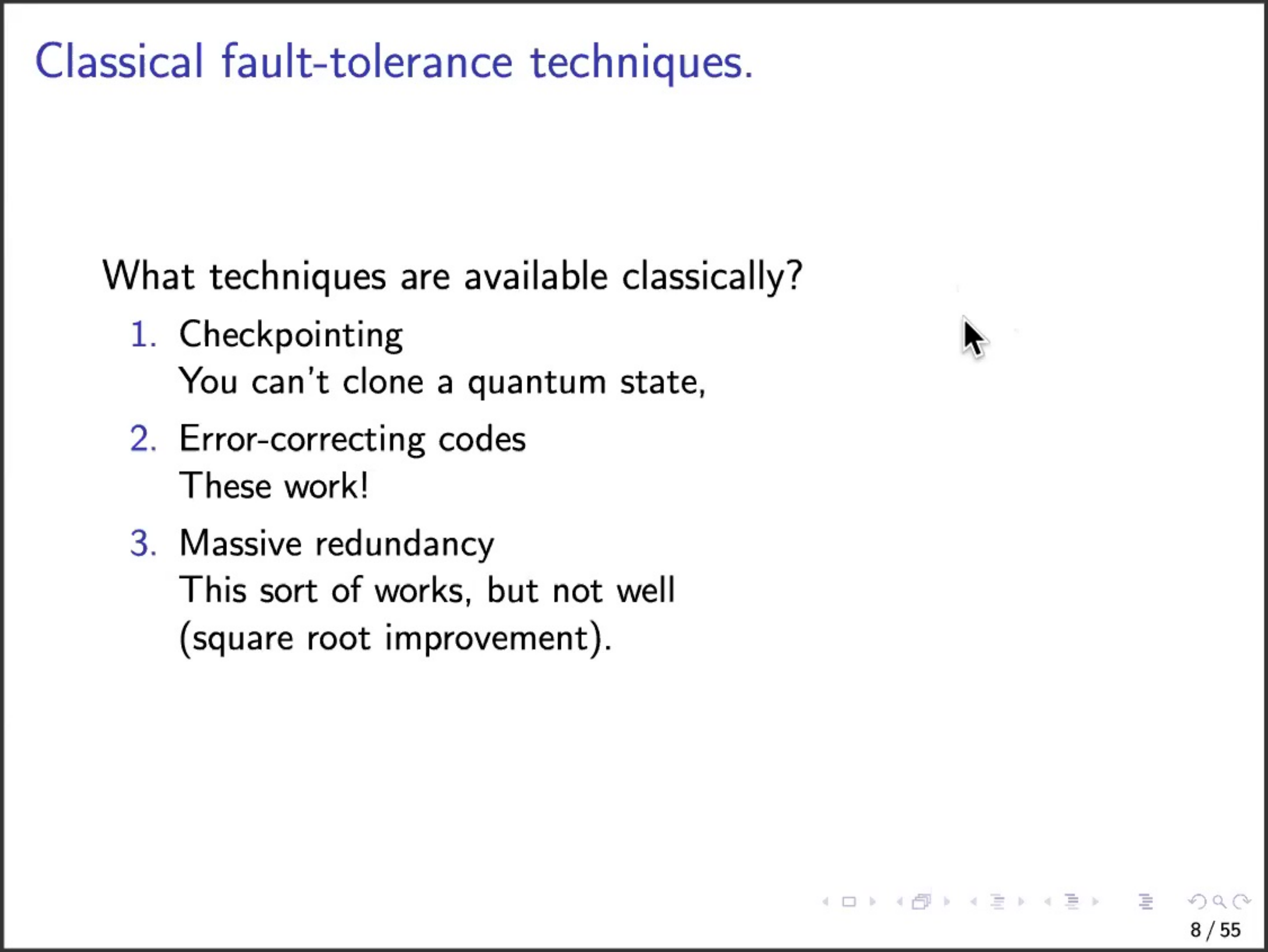


Classical fault-tolerance techniques.

What techniques are available classically?

1. Checkpointing
   You can't clone a quantum state
2. Error-correcting codes
   These work!
3. Massive redundancy

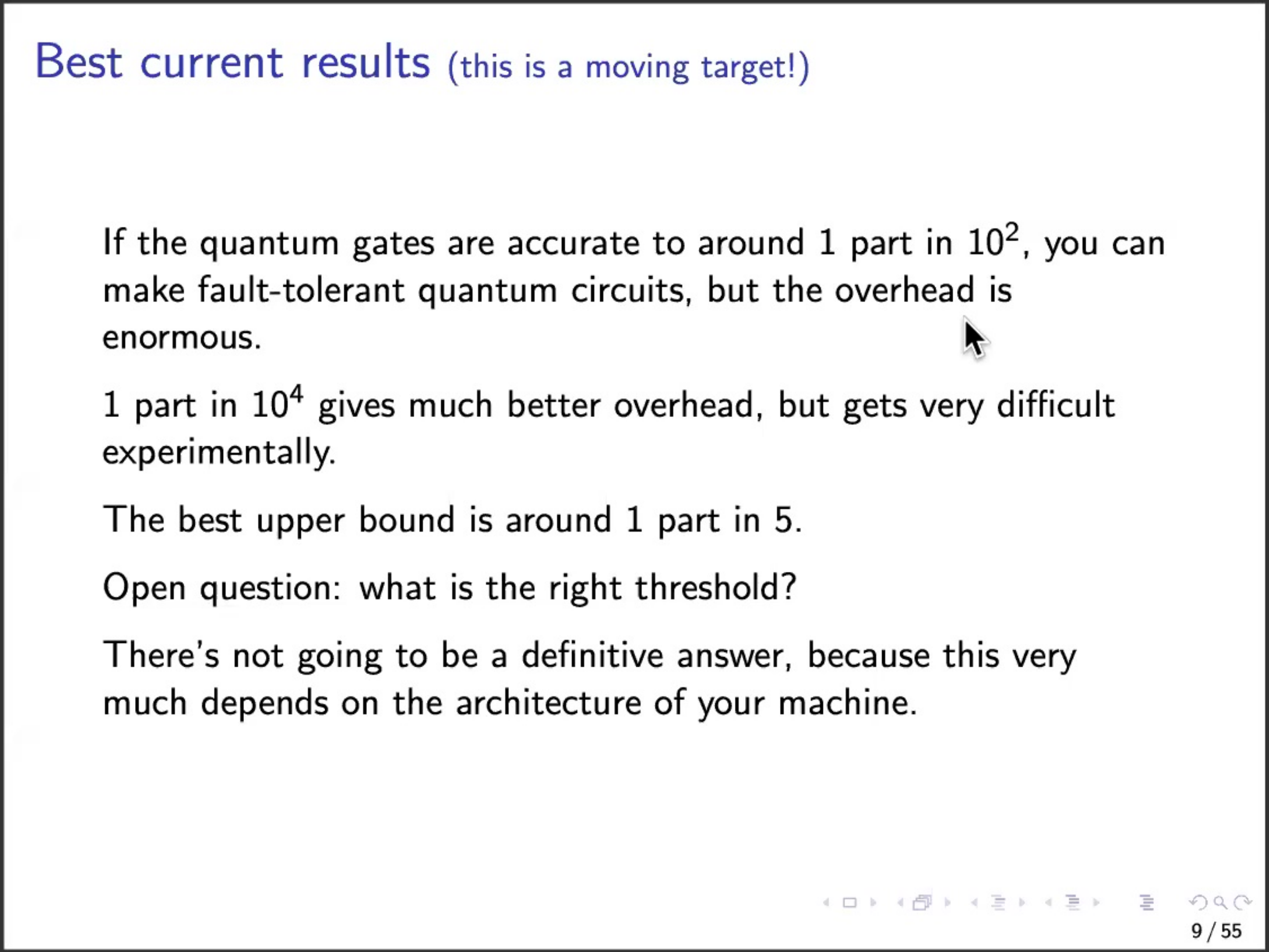
7 / 55



Classical fault-tolerance techniques.

What techniques are available classically?

1. Checkpointing
   You can't clone a quantum state,
2. Error-correcting codes
   These work!
3. Massive redundancy
   This sort of works, but not well
   (square root improvement).



Best current results (this is a moving target!)

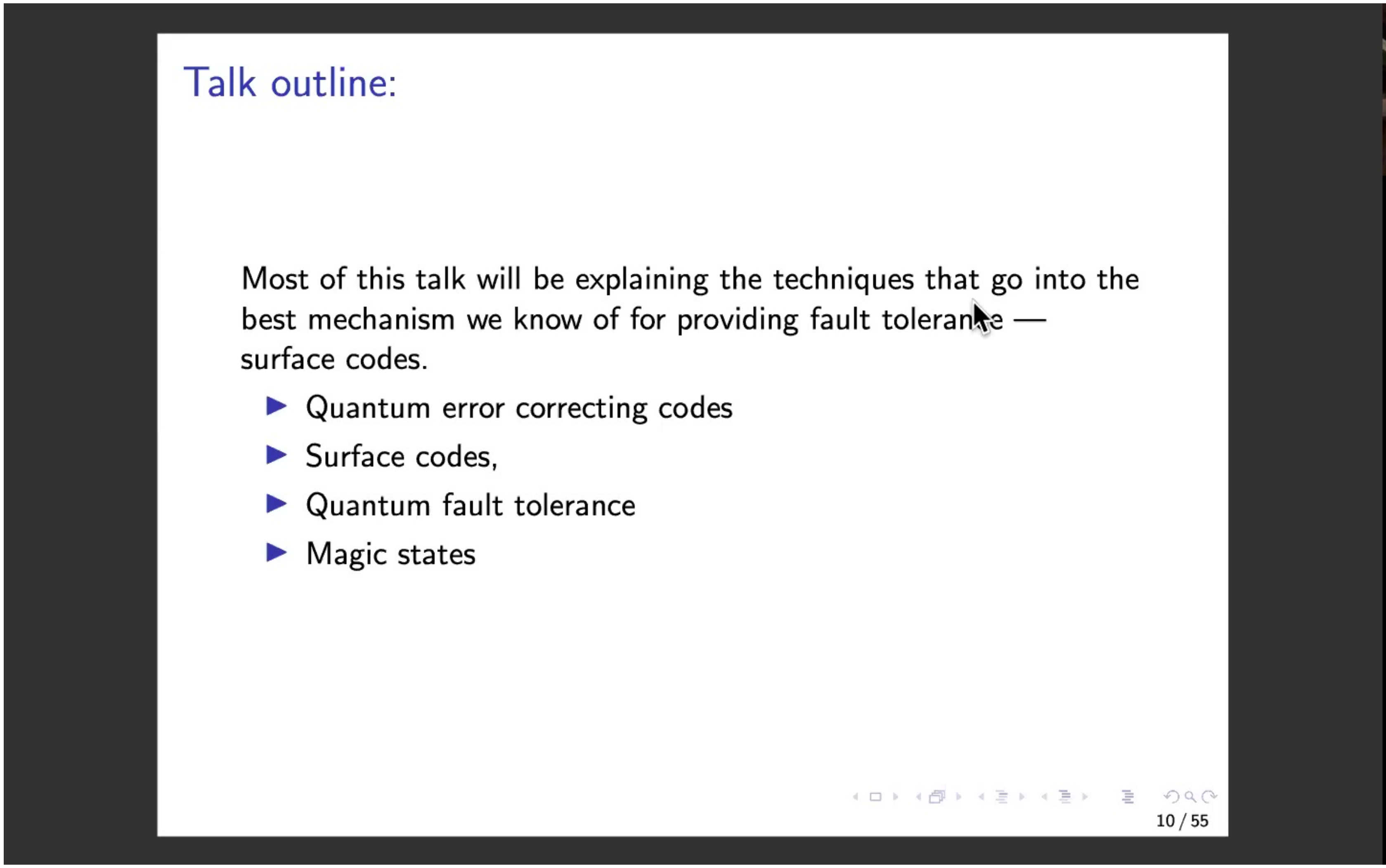
If the quantum gates are accurate to around 1 part in 10^2, you can make fault-tolerant quantum circuits, but the overhead is enormous.

1 part in 10^4 gives much better overhead, but gets very difficult experimentally.

The best upper bound is around 1 part in 5.

Open question: what is the right threshold?

There's not going to be a definitive answer, because this very much depends on the architecture of your machine.

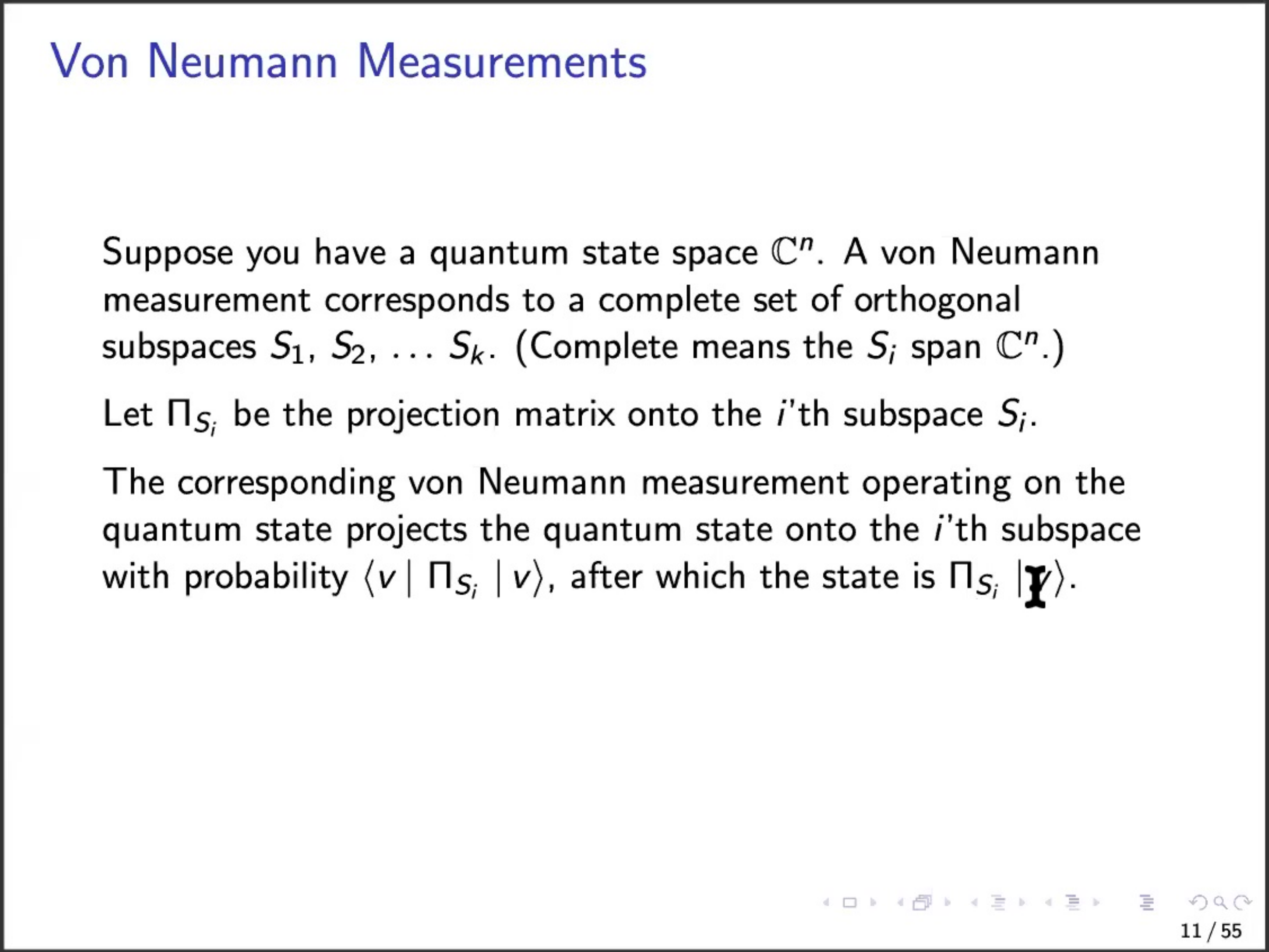


Talk outline:

Most of this talk will be explaining the techniques that go into the best mechanism we know of for providing fault tolerance — surface codes.

- Quantum error correcting codes
- Surface codes,
- Quantum fault tolerance
- Magic states

10 / 55

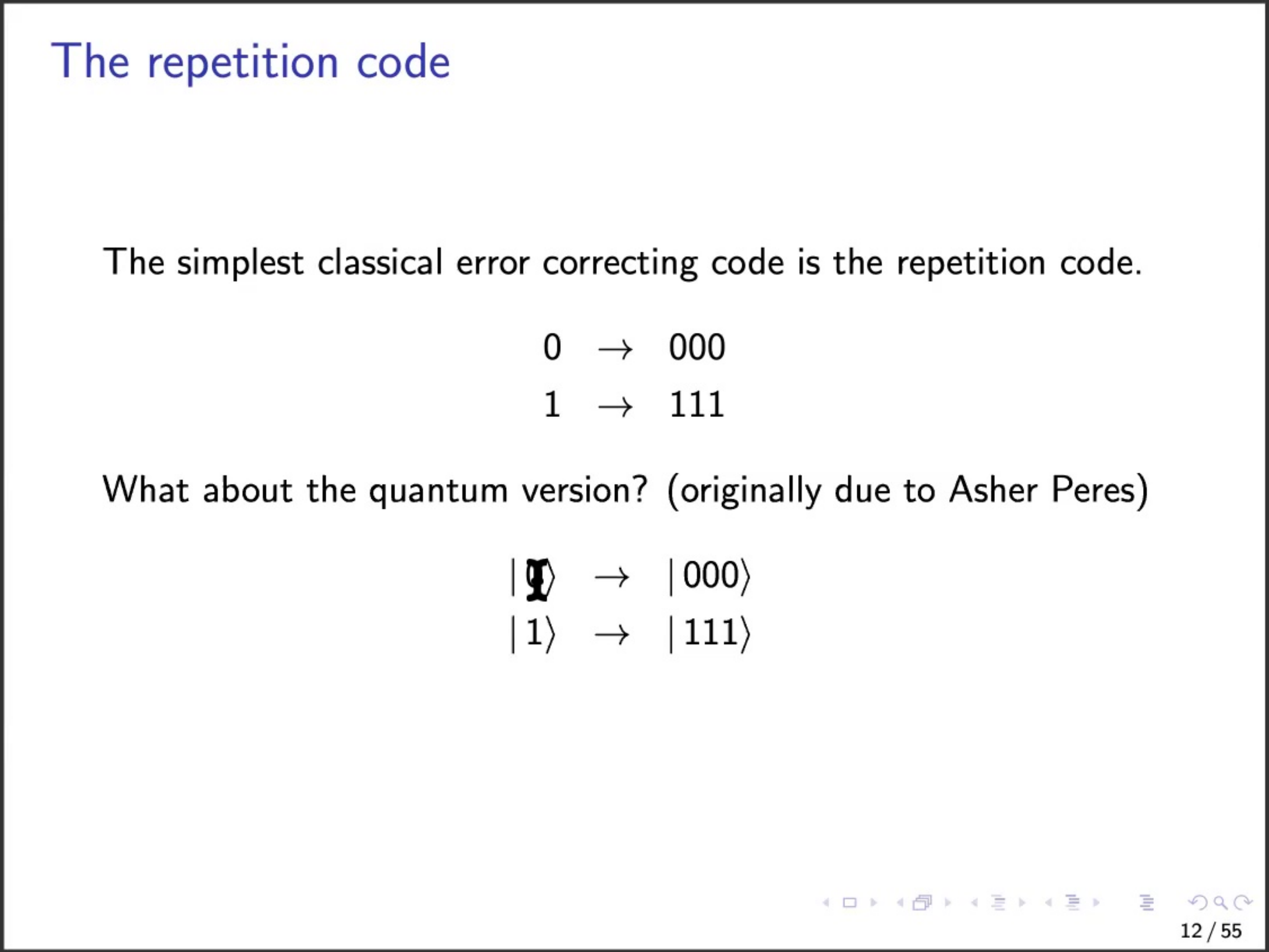


Von Neumann Measurements

Suppose you have a quantum state space $\mathbb{C}^n$. A von Neumann measurement corresponds to a complete set of orthogonal subspaces S_1, S_2, ... S_k. (Complete means the S_i span $\mathbb{C}^n$.)

Let Π_{S_i} be the projection matrix onto the i'th subspace S_i.

The corresponding von Neumann measurement operating on the quantum state projects the quantum state onto the i'th subspace with probability $\langle v \mid \Pi_{S_i} \mid v\rangle$, after which the state is $\Pi_{S_i} \mid v\rangle$.



The repetition code

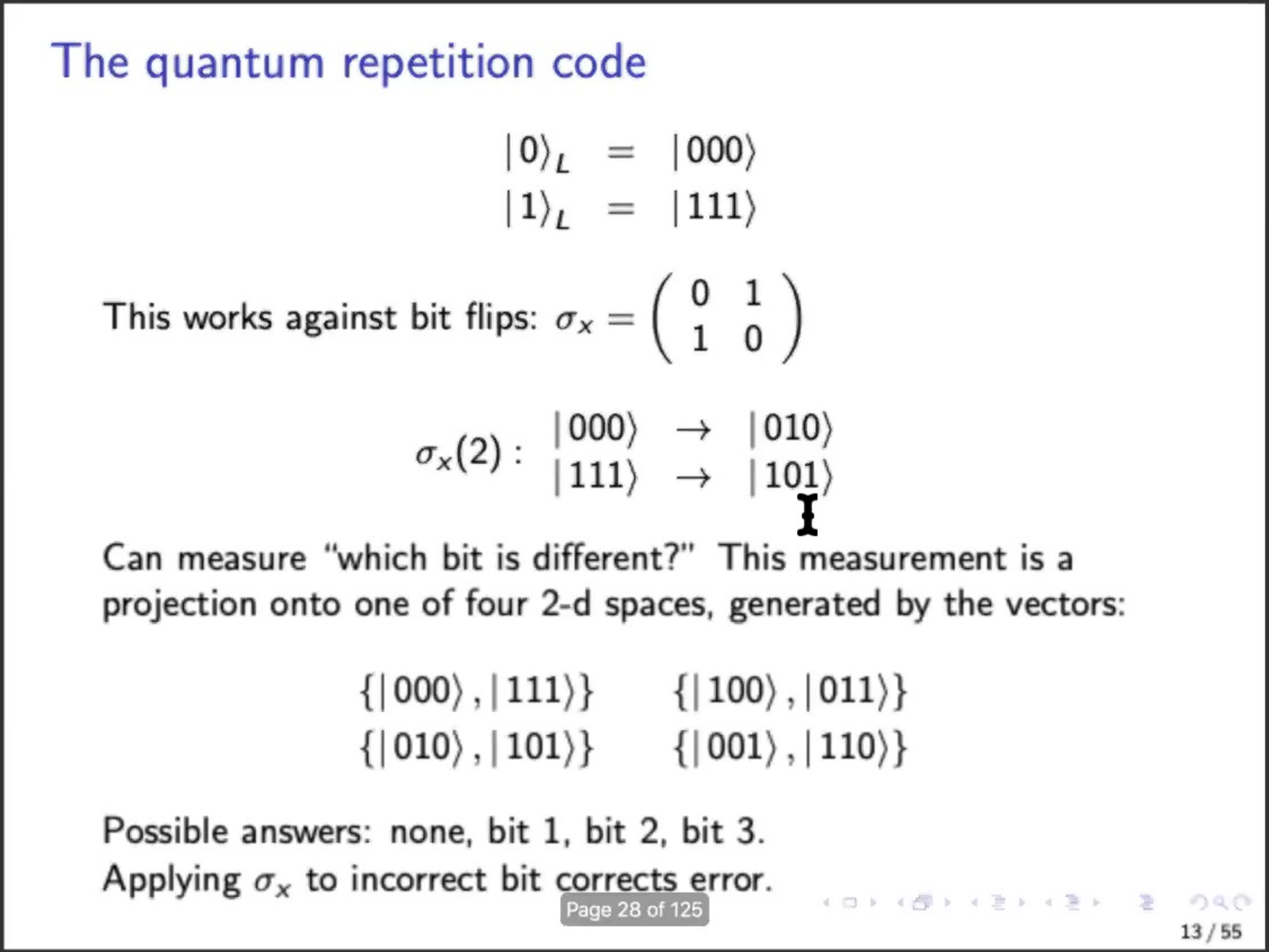
The simplest classical error correcting code is the repetition code.

$$\begin{aligned} 0 &\rightarrow 000 \\ 1 &\rightarrow 111 \end{aligned}$$

What about the quantum version? (originally due to Asher Peres)

$$\begin{aligned} |0\rangle &\rightarrow |000\rangle \\ |1\rangle &\rightarrow |111\rangle \end{aligned}$$

12 / 55



The quantum repetition code

$$\begin{aligned} |0\rangle_L &= |000\rangle \\ |1\rangle_L &= |111\rangle \end{aligned}$$

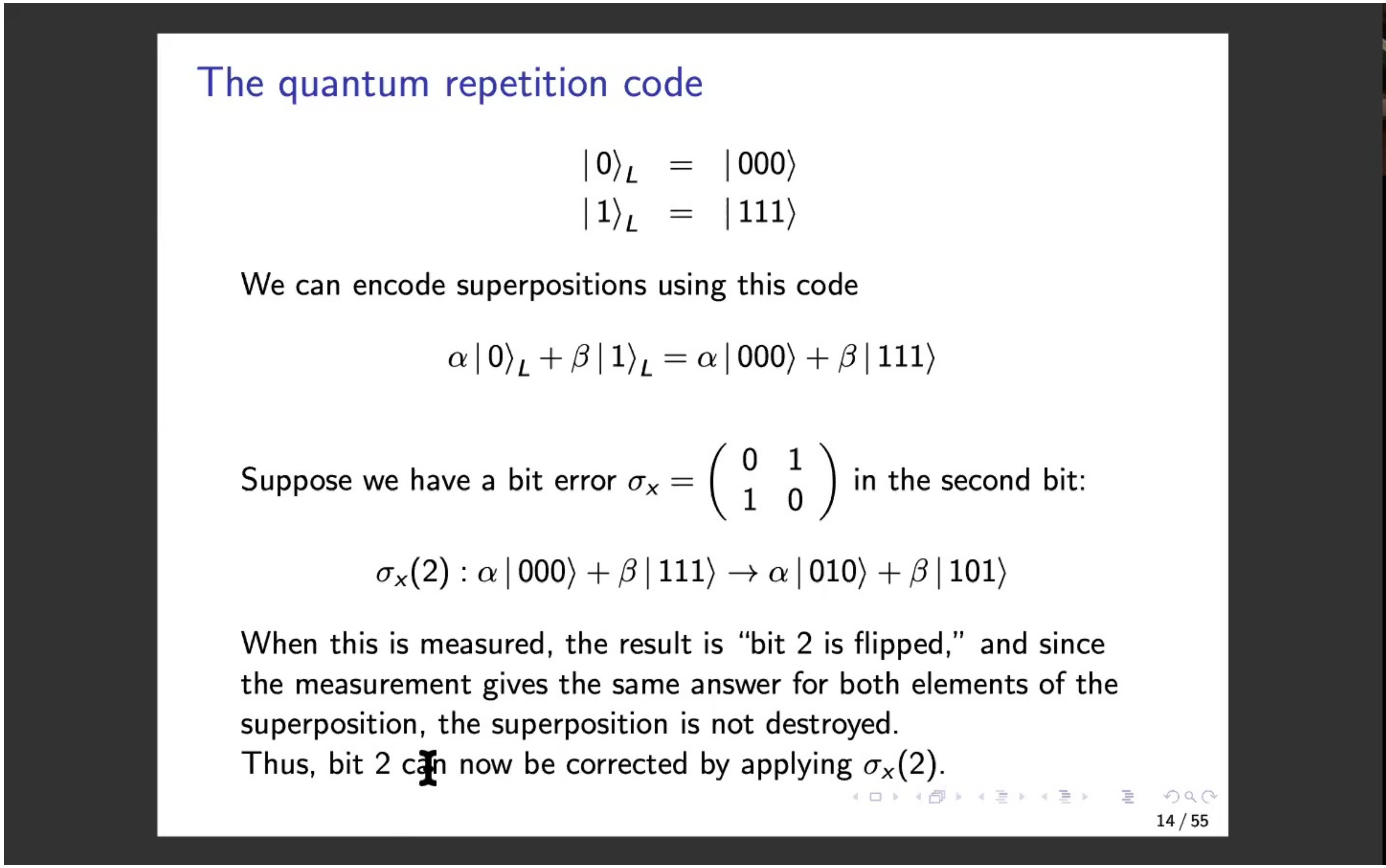
This works against bit flips: $\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

$$\sigma_x(2): \begin{aligned} |000\rangle &\rightarrow |010\rangle \\ |111\rangle &\rightarrow |101\rangle \end{aligned}$$

Can measure "which bit is different?" This measurement is a projection onto one of four 2-d spaces, generated by the vectors:

$$\begin{array}{cc} \{|000\rangle, |111\rangle\} & \{|100\rangle, |011\rangle\} \\ \{|010\rangle, |101\rangle\} & \{|001\rangle, |110\rangle\} \end{array}$$

Possible answers: none, bit 1, bit 2, bit 3.
Applying σ_x to incorrect bit corrects error.



The quantum repetition code

$$\begin{aligned} |0\rangle_L &= |000\rangle \\ |1\rangle_L &= |111\rangle \end{aligned}$$

We can encode superpositions using this code

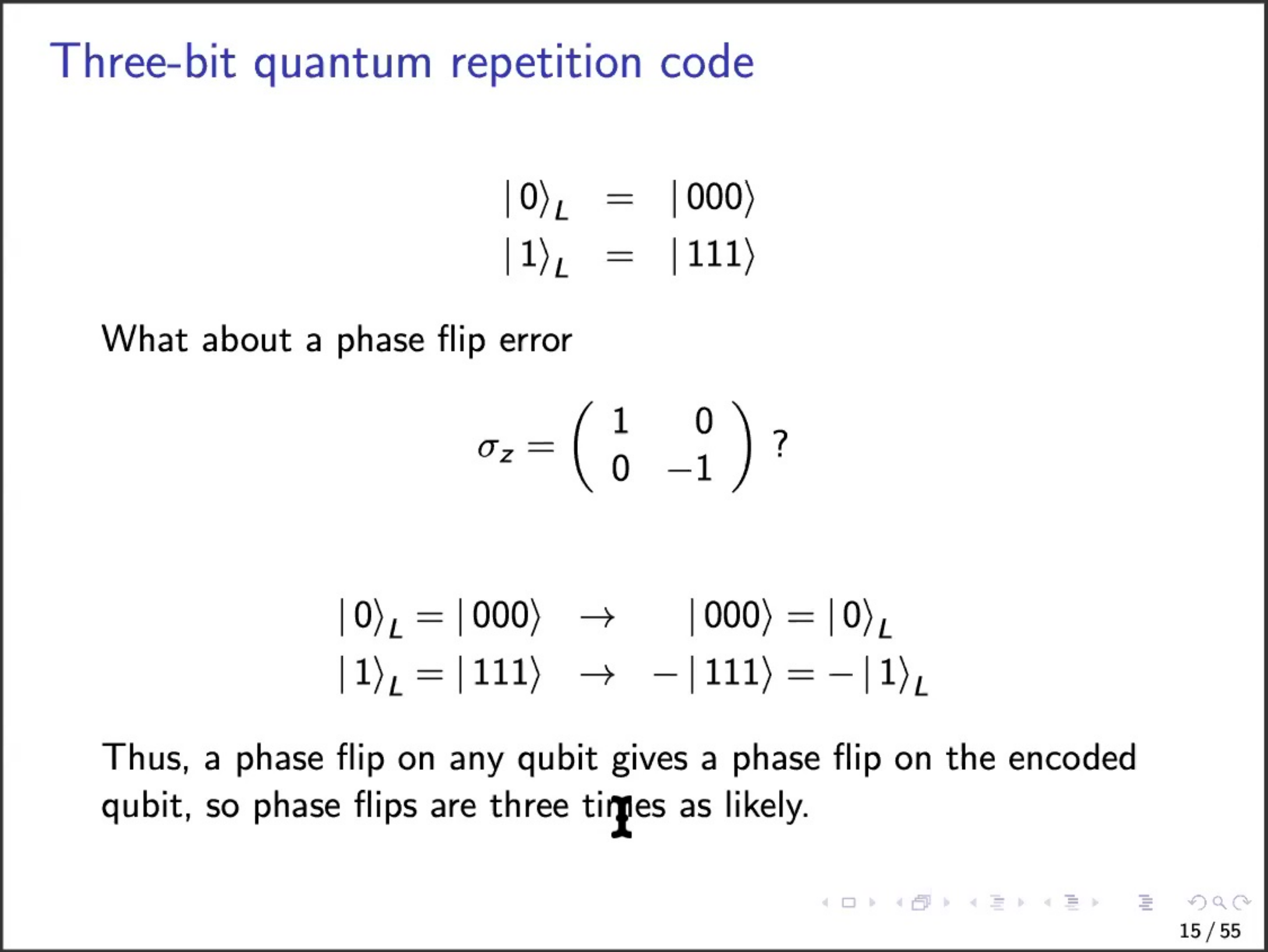
$$\alpha|0\rangle_L + \beta|1\rangle_L = \alpha|000\rangle + \beta|111\rangle$$

Suppose we have a bit error $\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ in the second bit:

$$\sigma_x(2) : \alpha|000\rangle + \beta|111\rangle \rightarrow \alpha|010\rangle + \beta|101\rangle$$

When this is measured, the result is “bit 2 is flipped,” and since the measurement gives the same answer for both elements of the superposition, the superposition is not destroyed.
Thus, bit 2 can now be corrected by applying $\sigma_x(2)$.

14 / 55



Three-bit quantum repetition code

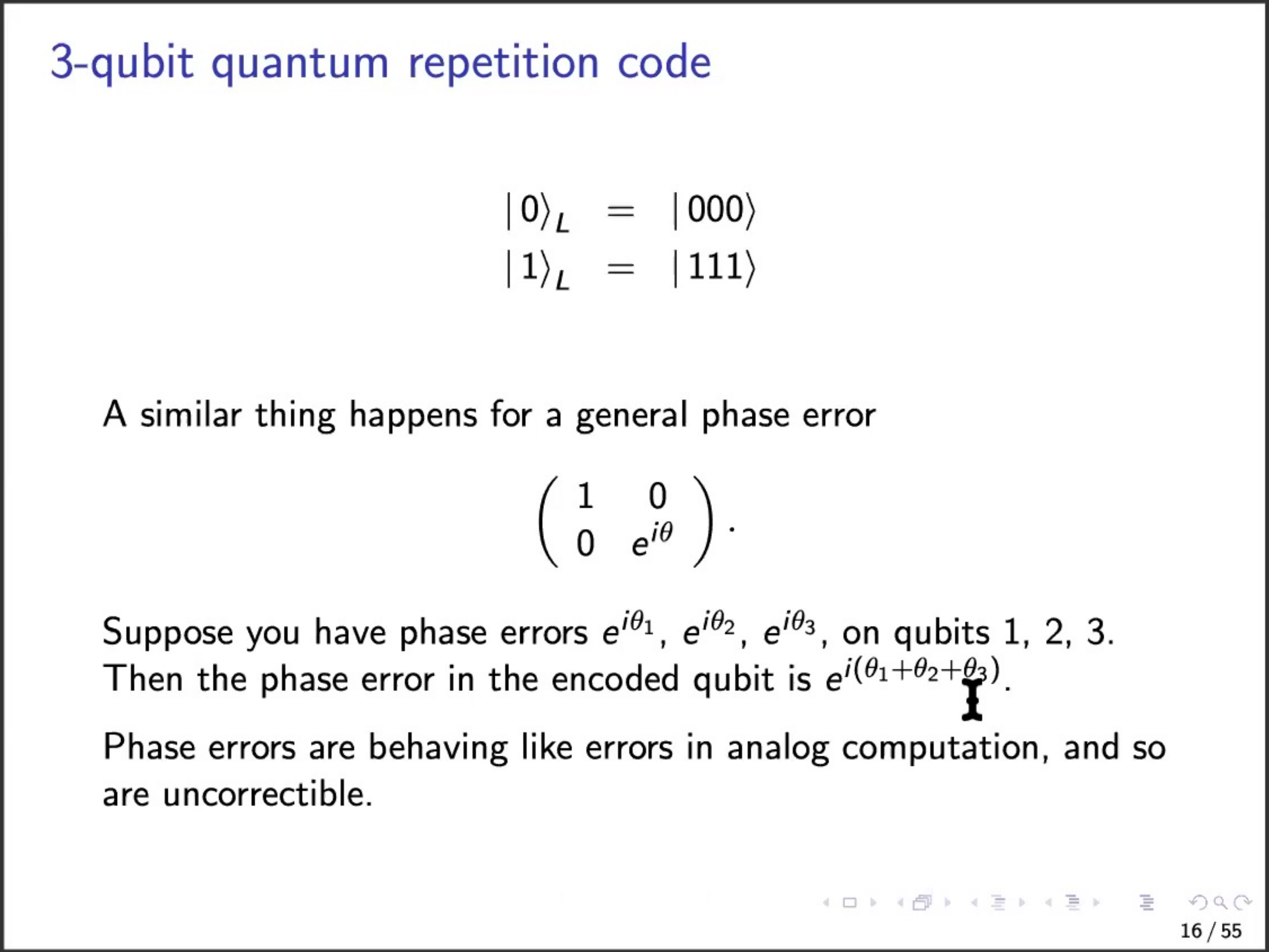
$$\begin{aligned} |0\rangle_L &= |000\rangle \\ |1\rangle_L &= |111\rangle \end{aligned}$$

What about a phase flip error

$$\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} ?$$

$$\begin{aligned} |0\rangle_L = |000\rangle &\to \quad |000\rangle = |0\rangle_L \\ |1\rangle_L = |111\rangle &\to -|111\rangle = -|1\rangle_L \end{aligned}$$

Thus, a phase flip on any qubit gives a phase flip on the encoded qubit, so phase flips are three times as likely.



3-qubit quantum repetition code

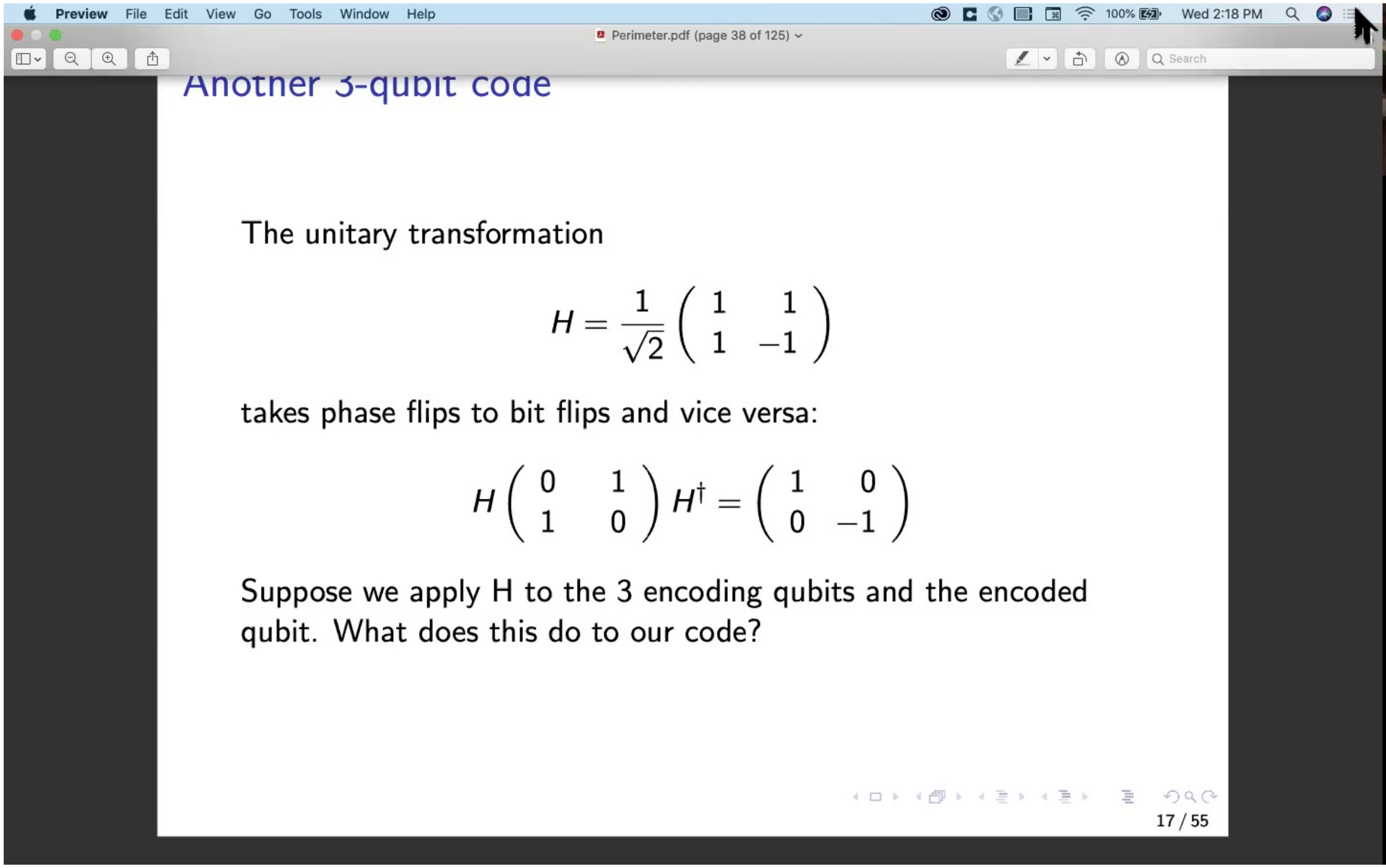
$$\begin{aligned} |0\rangle_L &= |000\rangle \\ |1\rangle_L &= |111\rangle \end{aligned}$$

A similar thing happens for a general phase error

$$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}.$$

Suppose you have phase errors $e^{i\theta_1}$, $e^{i\theta_2}$, $e^{i\theta_3}$, on qubits 1, 2, 3. Then the phase error in the encoded qubit is $e^{i(\theta_1+\theta_2+\theta_3)}$.

Phase errors are behaving like errors in analog computation, and so are uncorrectible.


Preview File Edit View Go Tools Window Help
Wed 2:18 PM
Perimeter.pdf (page 38 of 125)

Another 3-qubit code

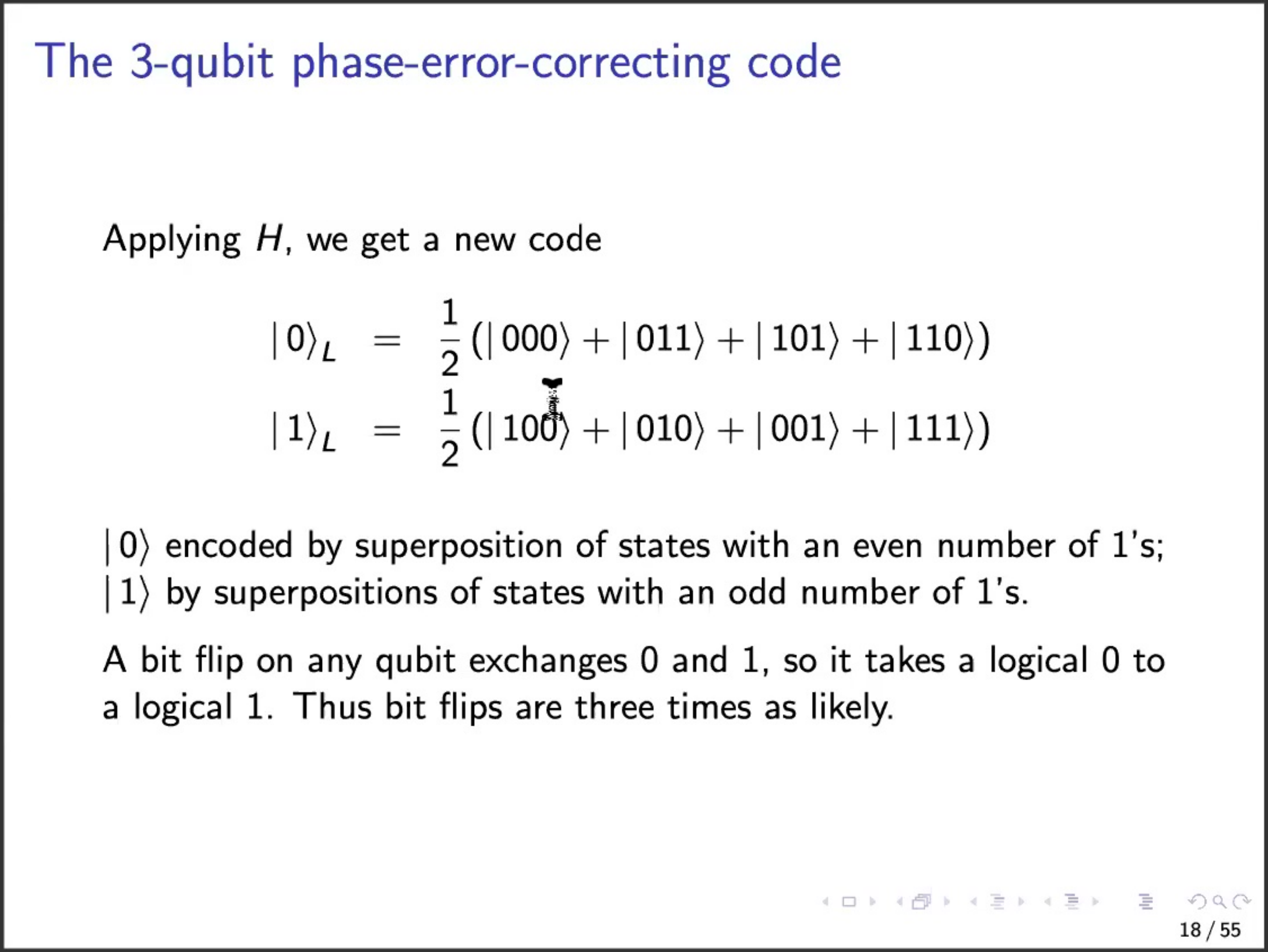
The unitary transformation

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

takes phase flips to bit flips and vice versa:

$$H \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} H^{\dagger} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Suppose we apply H to the 3 encoding qubits and the encoded qubit. What does this do to our code?



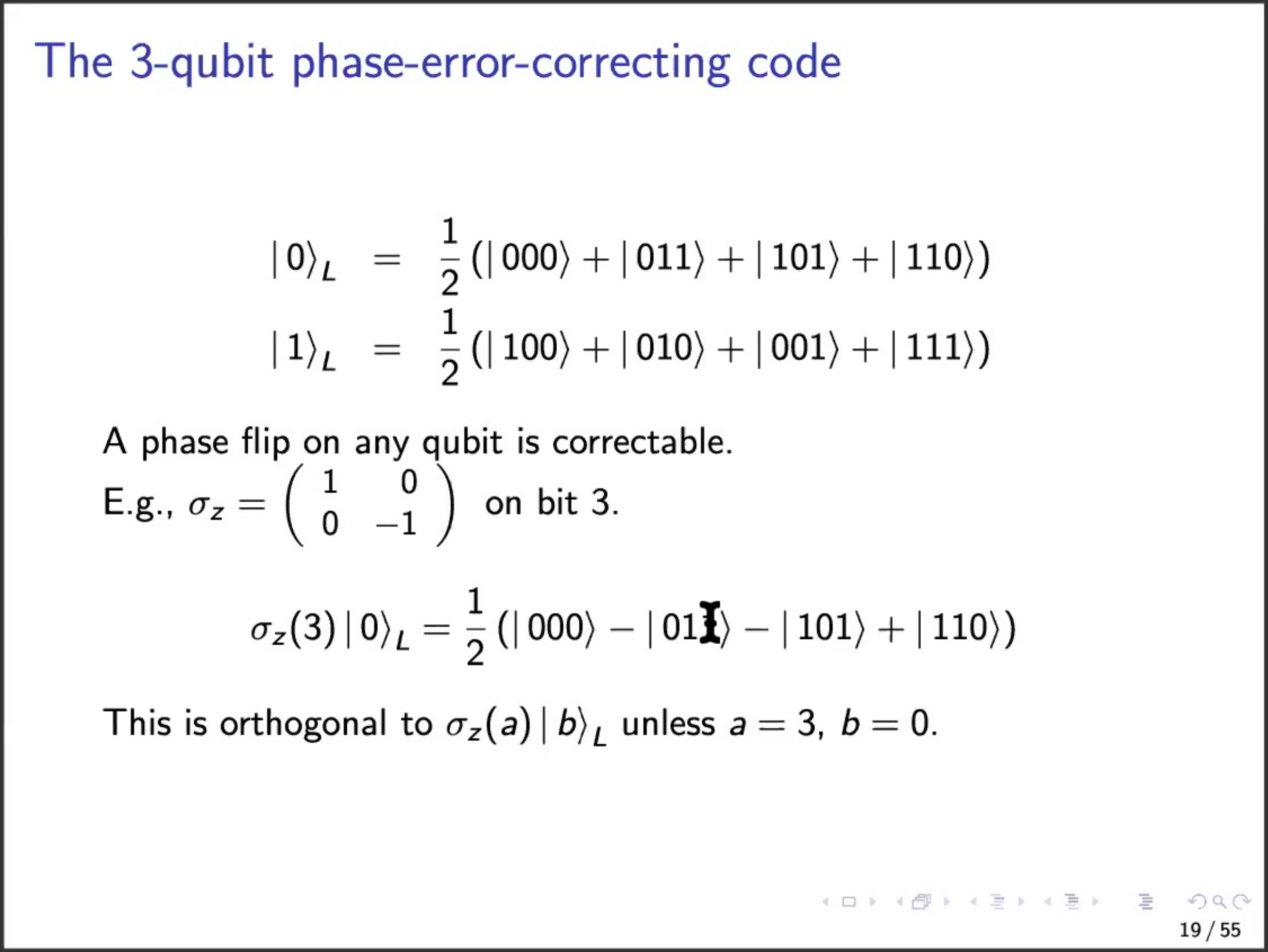
The 3-qubit phase-error-correcting code

Applying H, we get a new code

$$\begin{aligned} |\,0\rangle_L &= \frac{1}{2}(|\,000\rangle + |\,011\rangle + |\,101\rangle + |\,110\rangle) \\ |\,1\rangle_L &= \frac{1}{2}(|\,100\rangle + |\,010\rangle + |\,001\rangle + |\,111\rangle) \end{aligned}$$

$|\,0\rangle$ encoded by superposition of states with an even number of 1's;
$|\,1\rangle$ by superpositions of states with an odd number of 1's.

A bit flip on any qubit exchanges 0 and 1, so it takes a logical 0 to a logical 1. Thus bit flips are three times as likely.



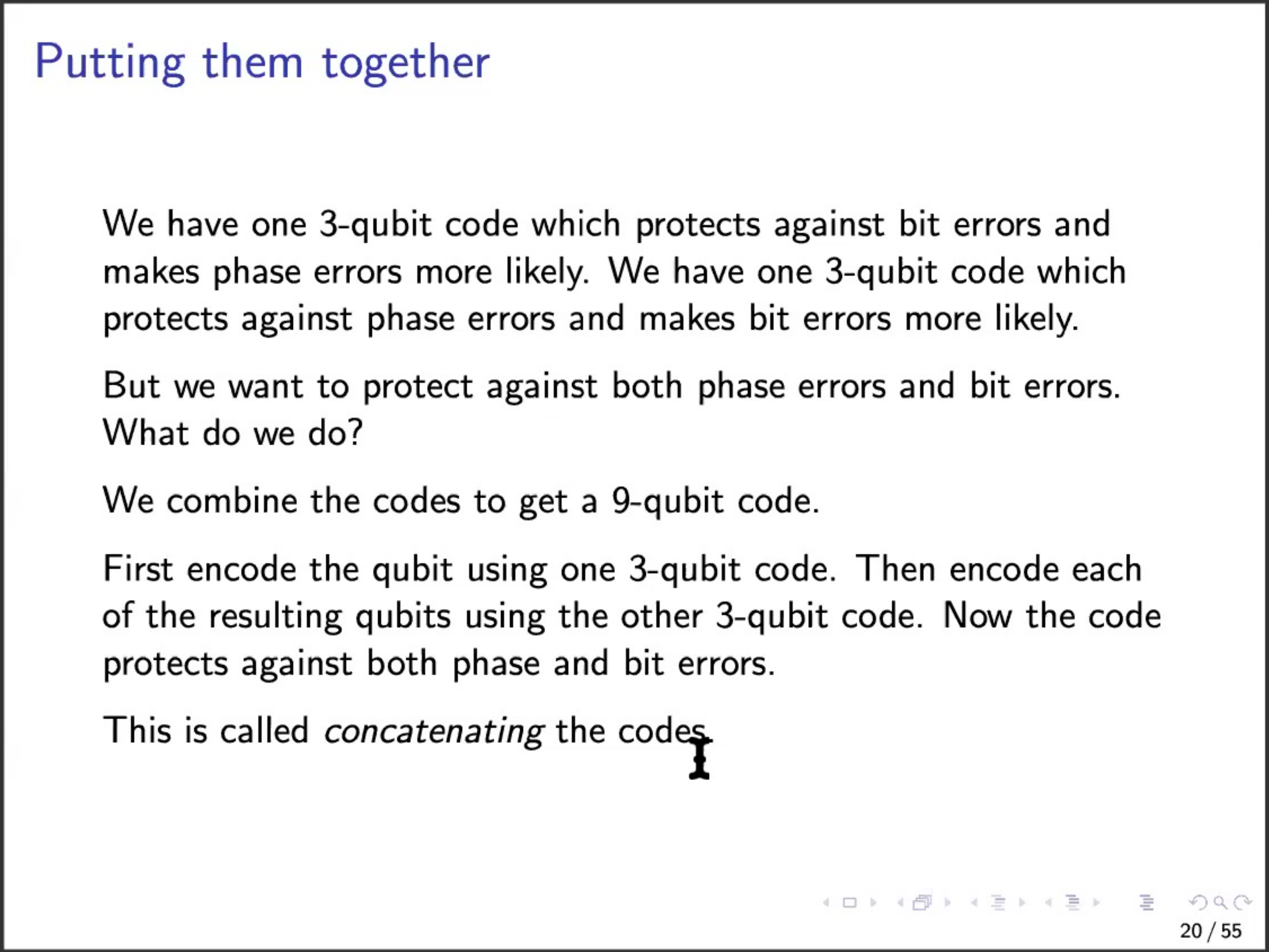
The 3-qubit phase-error-correcting code

$$
\begin{aligned}
|\,0\rangle_L &= \frac{1}{2}(|\,000\rangle + |\,011\rangle + |\,101\rangle + |\,110\rangle) \\
|\,1\rangle_L &= \frac{1}{2}(|\,100\rangle + |\,010\rangle + |\,001\rangle + |\,111\rangle)
\end{aligned}
$$

A phase flip on any qubit is correctable.
E.g., $\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ on bit 3.

$$\sigma_z(3)\,|\,0\rangle_L = \frac{1}{2}(|\,000\rangle - |\,011\rangle - |\,101\rangle + |\,110\rangle)$$

This is orthogonal to $\sigma_z(a)\,|\,b\rangle_L$ unless $a = 3,\ b = 0$.



Putting them together

We have one 3-qubit code which protects against bit errors and makes phase errors more likely. We have one 3-qubit code which protects against phase errors and makes bit errors more likely.

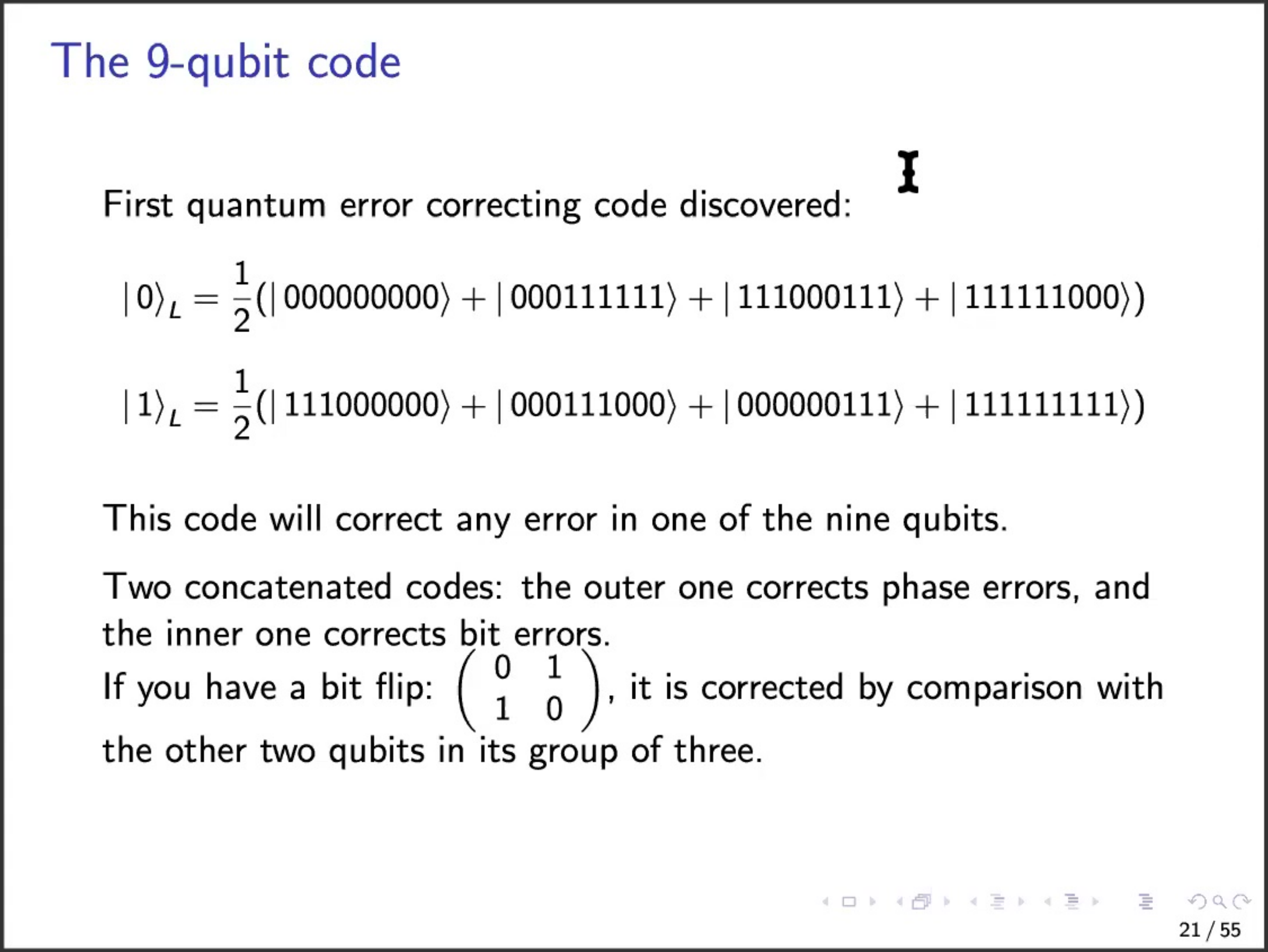
But we want to protect against both phase errors and bit errors. What do we do?

We combine the codes to get a 9-qubit code.

First encode the qubit using one 3-qubit code. Then encode each of the resulting qubits using the other 3-qubit code. Now the code protects against both phase and bit errors.

This is called *concatenating* the codes.

20 / 55



The 9-qubit code

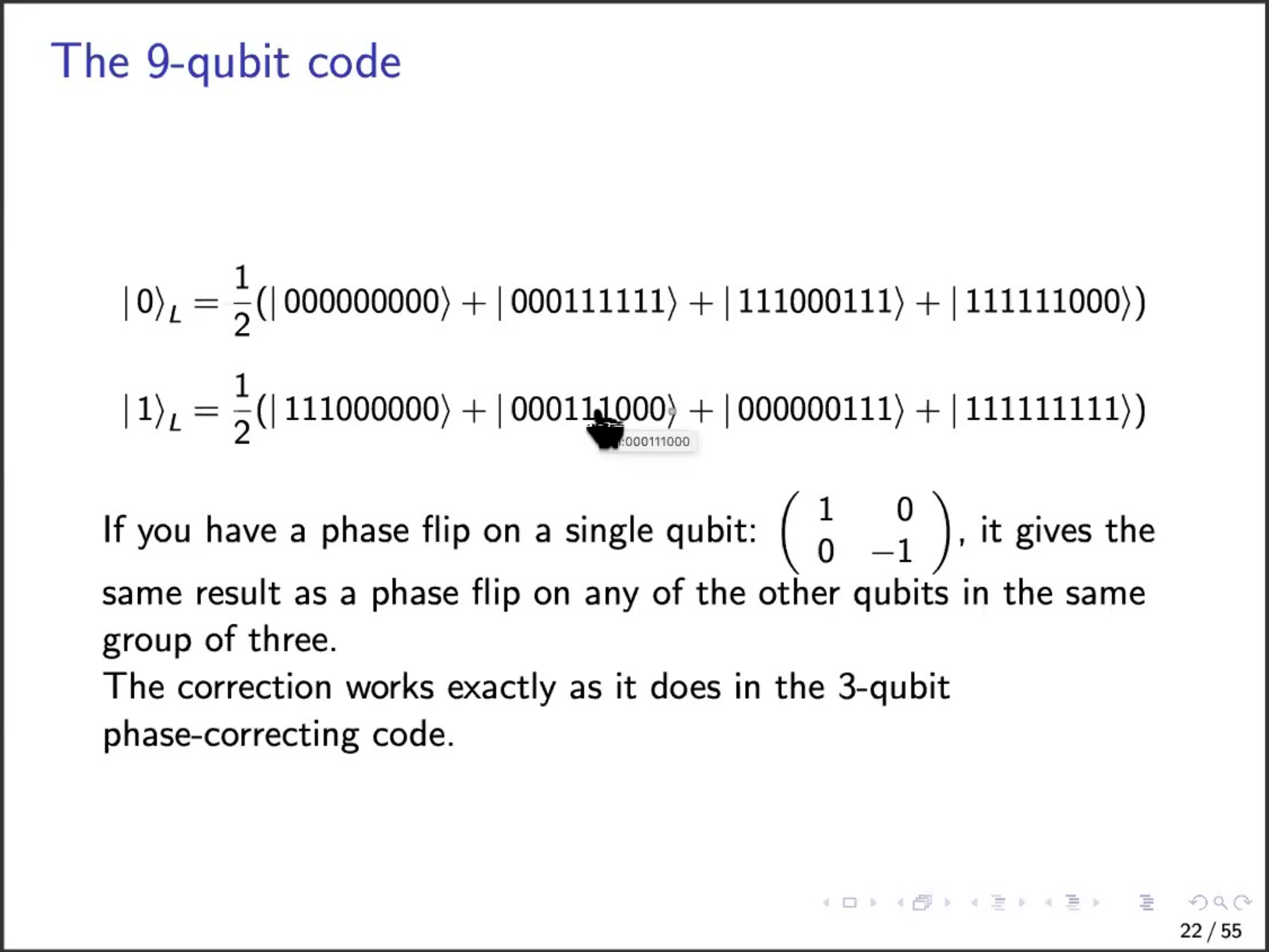
First quantum error correcting code discovered:

$$|0\rangle_L = \frac{1}{2}(|000000000\rangle + |000111111\rangle + |111000111\rangle + |111111000\rangle)$$

$$|1\rangle_L = \frac{1}{2}(|111000000\rangle + |000111000\rangle + |000000111\rangle + |111111111\rangle)$$

This code will correct any error in one of the nine qubits.

Two concatenated codes: the outer one corrects phase errors, and the inner one corrects bit errors.
If you have a bit flip: $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, it is corrected by comparison with the other two qubits in its group of three.

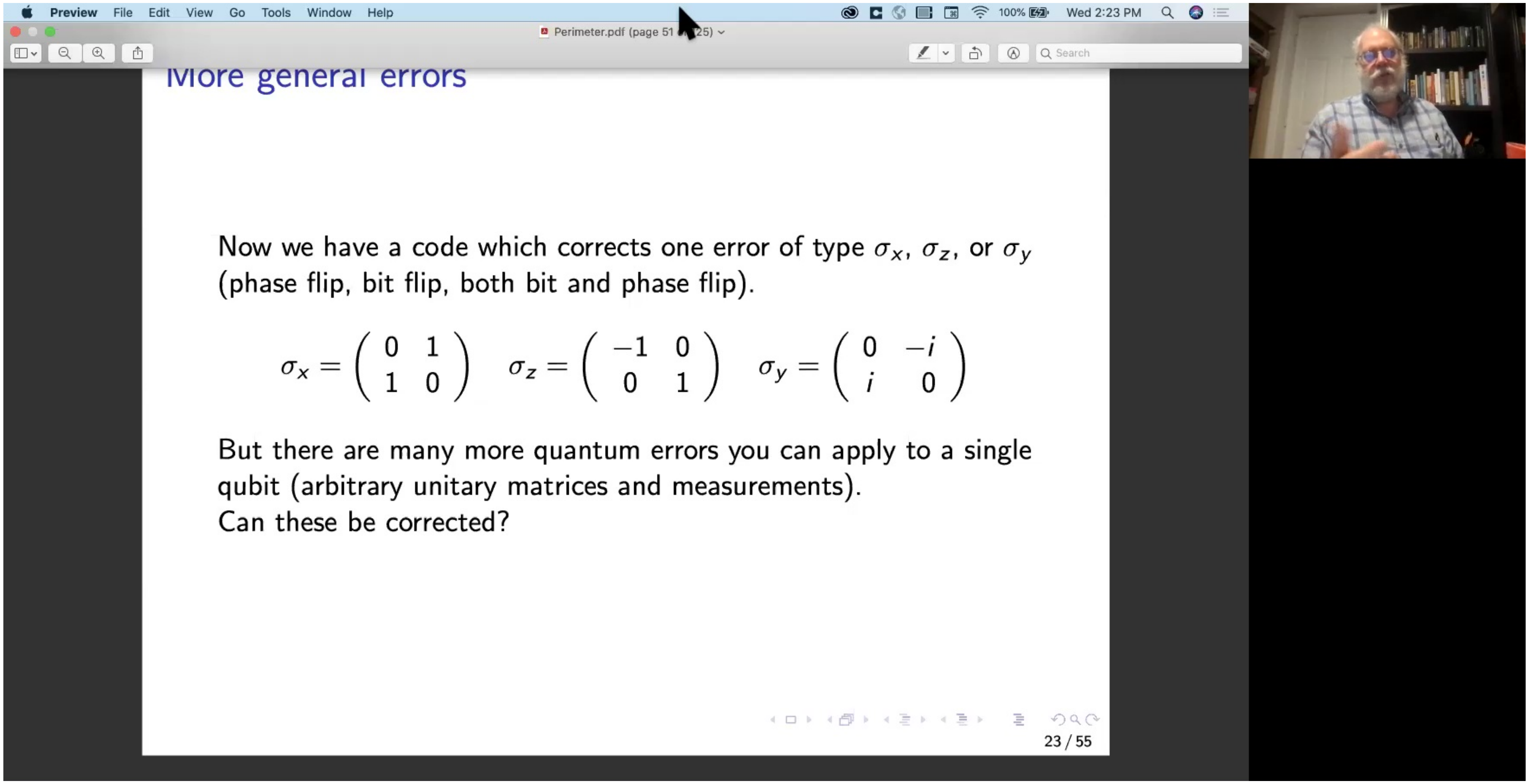


The 9-qubit code

$$|0\rangle_L = \frac{1}{2}(|000000000\rangle + |000111111\rangle + |111000111\rangle + |111111000\rangle)$$

$$|1\rangle_L = \frac{1}{2}(|111000000\rangle + |000111000\rangle + |000000111\rangle + |111111111\rangle)$$

If you have a phase flip on a single qubit: $\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$, it gives the same result as a phase flip on any of the other qubits in the same group of three.
The correction works exactly as it does in the 3-qubit phase-correcting code.


Preview File Edit View Go Tools Window Help
Wed 2:23 PM
Perimeter.pdf (page 51 of 125)
Search

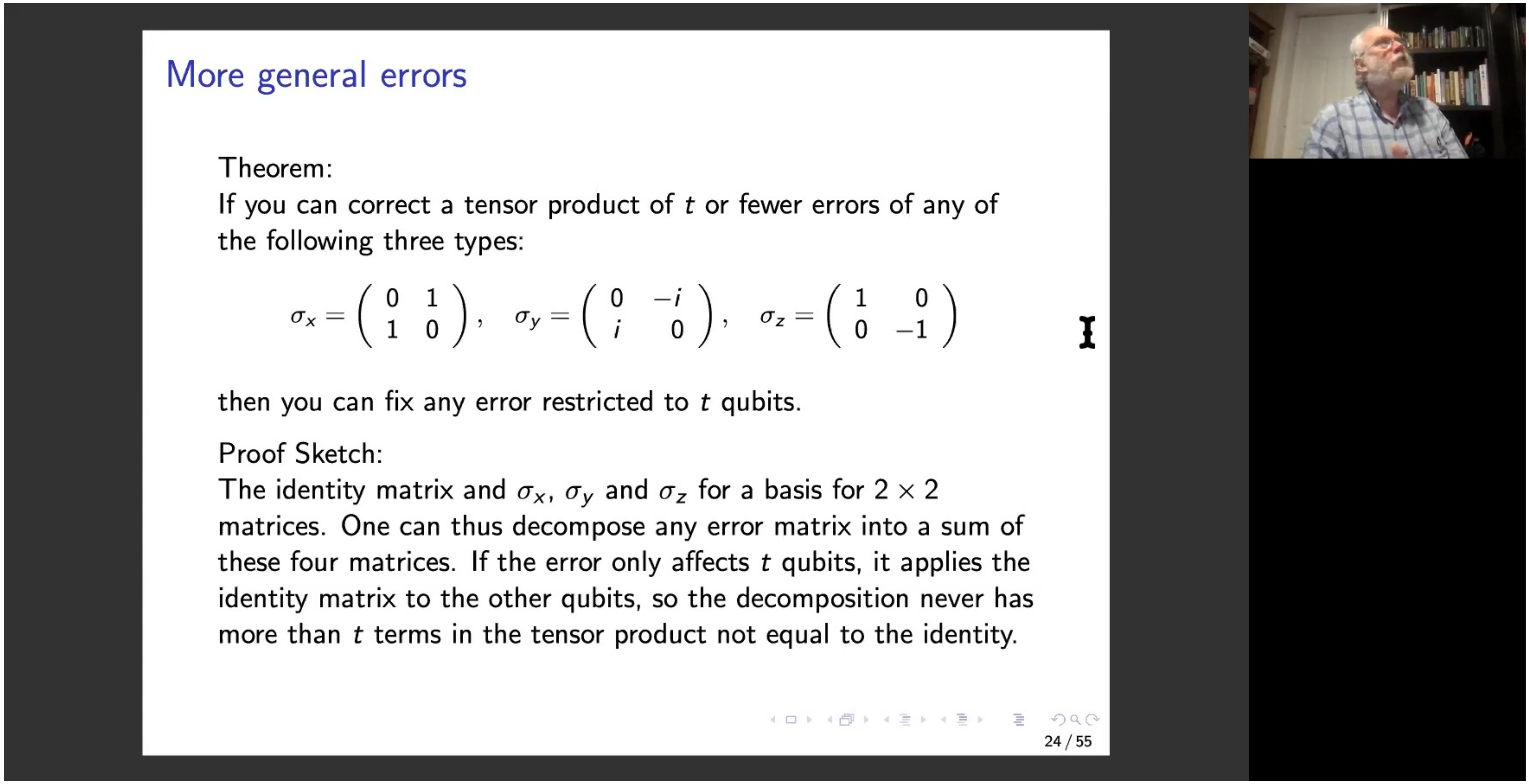
More general errors

Now we have a code which corrects one error of type σ_x, σ_z, or σ_y (phase flip, bit flip, both bit and phase flip).

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \sigma_z = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \quad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

But there are many more quantum errors you can apply to a single qubit (arbitrary unitary matrices and measurements).
Can these be corrected?

23 / 55



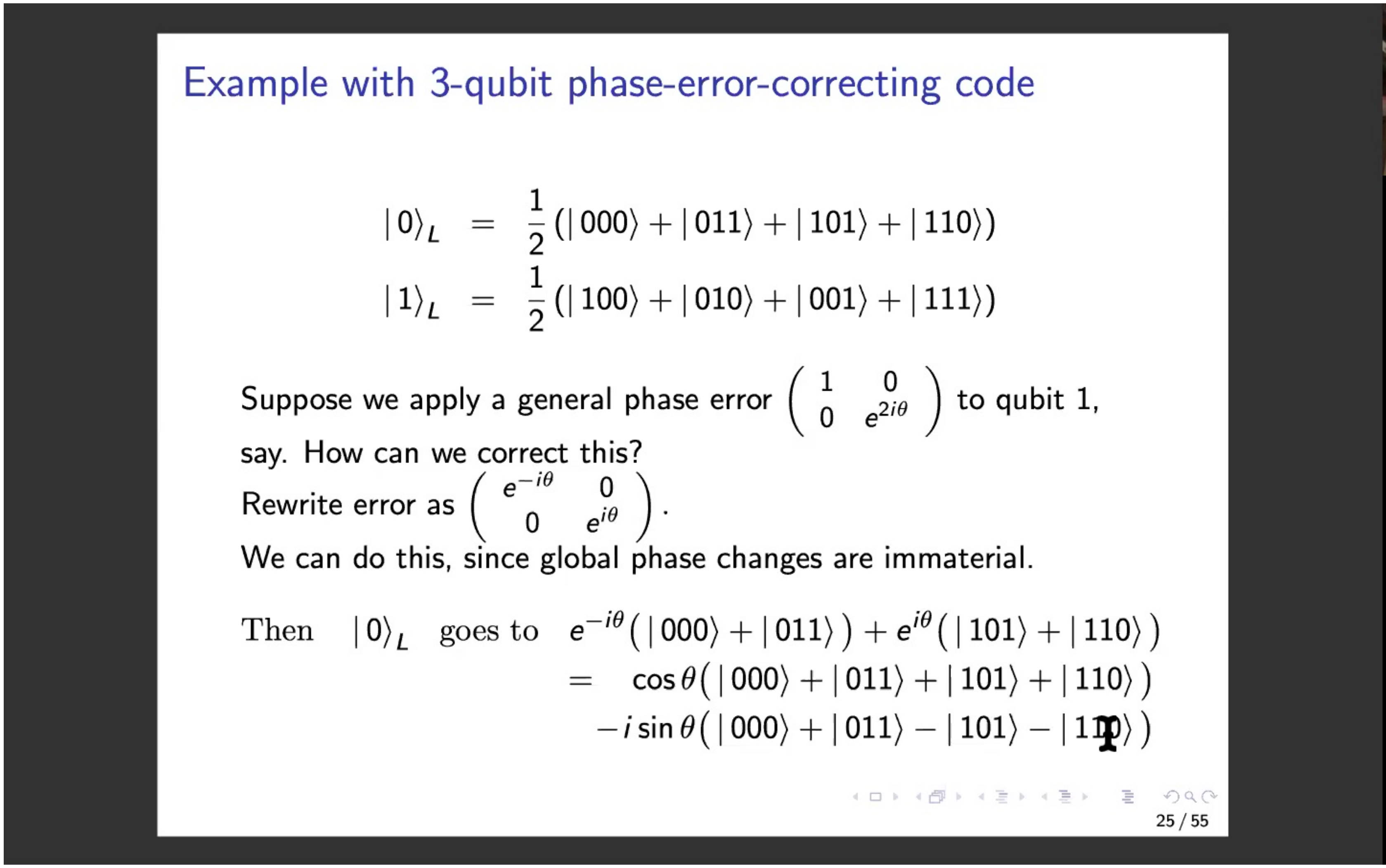
More general errors

Theorem:
If you can correct a tensor product of t or fewer errors of any of the following three types:

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

then you can fix any error restricted to t qubits.

Proof Sketch:
The identity matrix and σ_x, σ_y and σ_z for a basis for 2×2 matrices. One can thus decompose any error matrix into a sum of these four matrices. If the error only affects t qubits, it applies the identity matrix to the other qubits, so the decomposition never has more than t terms in the tensor product not equal to the identity.



Example with 3-qubit phase-error-correcting code

$$|0\rangle_L = \frac{1}{2}(|000\rangle + |011\rangle + |101\rangle + |110\rangle)$$
$$|1\rangle_L = \frac{1}{2}(|100\rangle + |010\rangle + |001\rangle + |111\rangle)$$

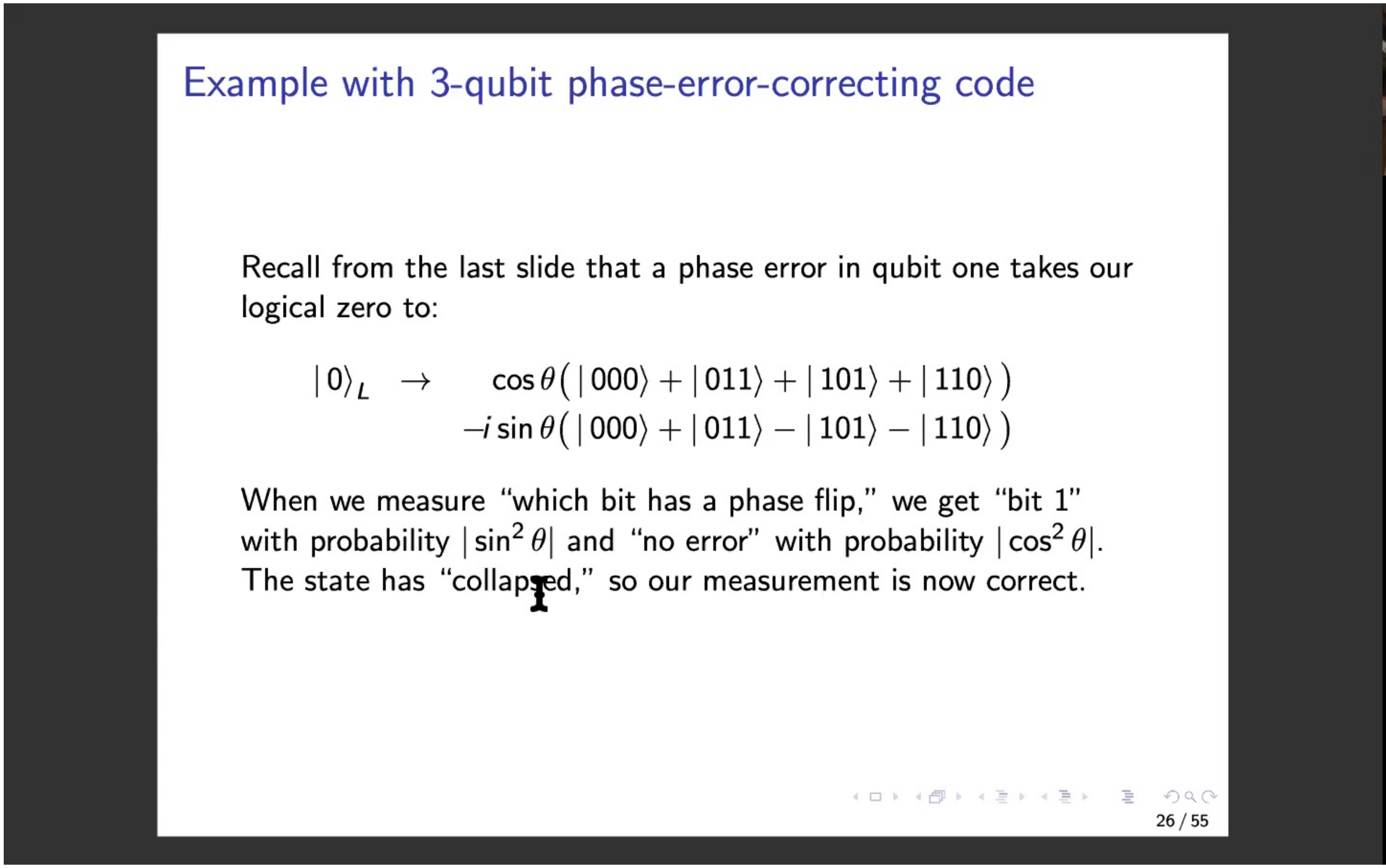
Suppose we apply a general phase error $\begin{pmatrix} 1 & 0 \\ 0 & e^{2i\theta} \end{pmatrix}$ to qubit 1, say. How can we correct this?

Rewrite error as $\begin{pmatrix} e^{-i\theta} & 0 \\ 0 & e^{i\theta} \end{pmatrix}$.

We can do this, since global phase changes are immaterial.

Then $|0\rangle_L$ goes to $e^{-i\theta}(|000\rangle + |011\rangle) + e^{i\theta}(|101\rangle + |110\rangle)$

$$= \cos\theta(|000\rangle + |011\rangle + |101\rangle + |110\rangle) - i\sin\theta(|000\rangle + |011\rangle - |101\rangle - |110\rangle)$$

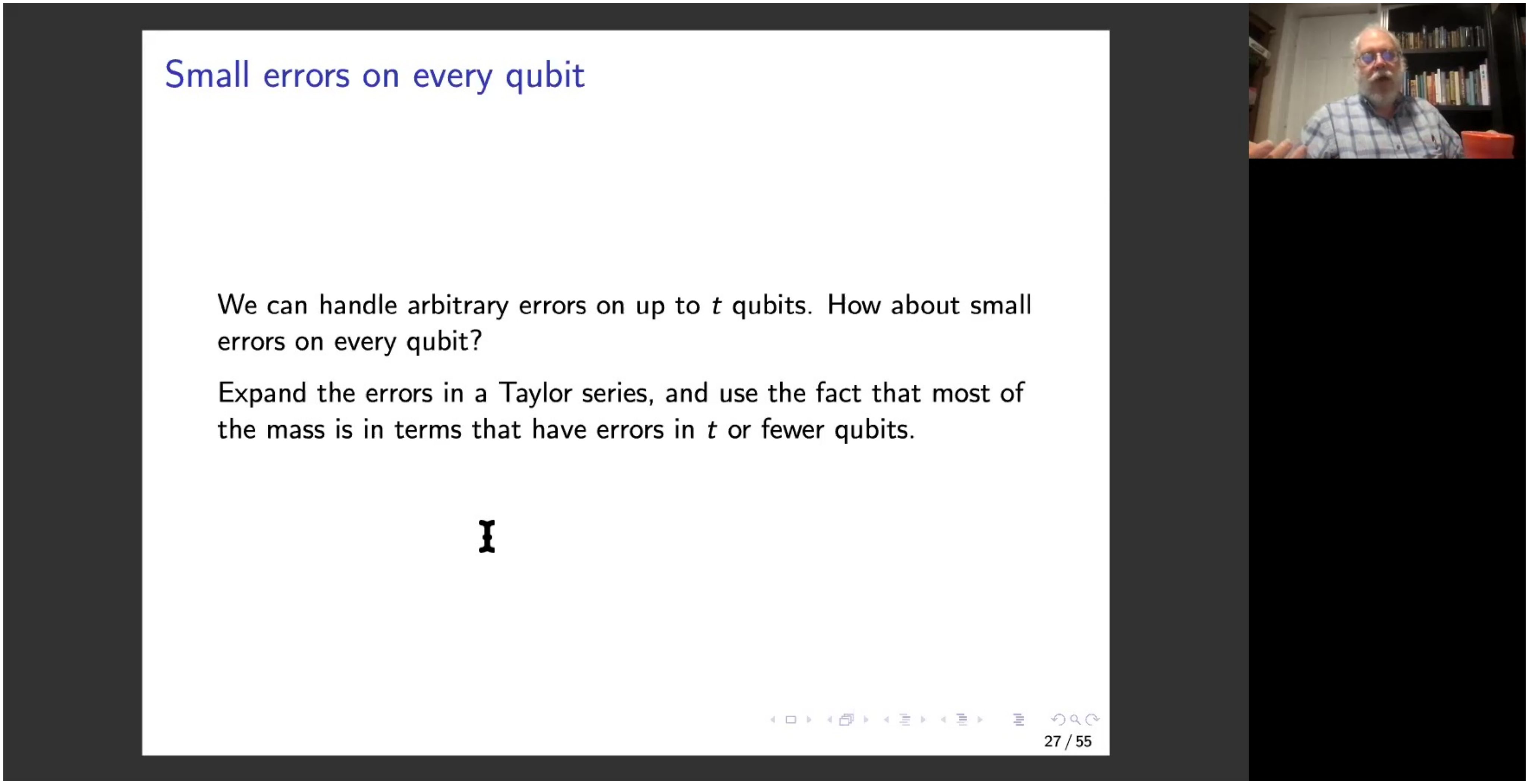


Example with 3-qubit phase-error-correcting code

Recall from the last slide that a phase error in qubit one takes our logical zero to:

$$\begin{aligned} |0\rangle_L \quad \rightarrow \quad & \cos\theta\big(|000\rangle + |011\rangle + |101\rangle + |110\rangle\big) \\ & -i\sin\theta\big(|000\rangle + |011\rangle - |101\rangle - |110\rangle\big) \end{aligned}$$

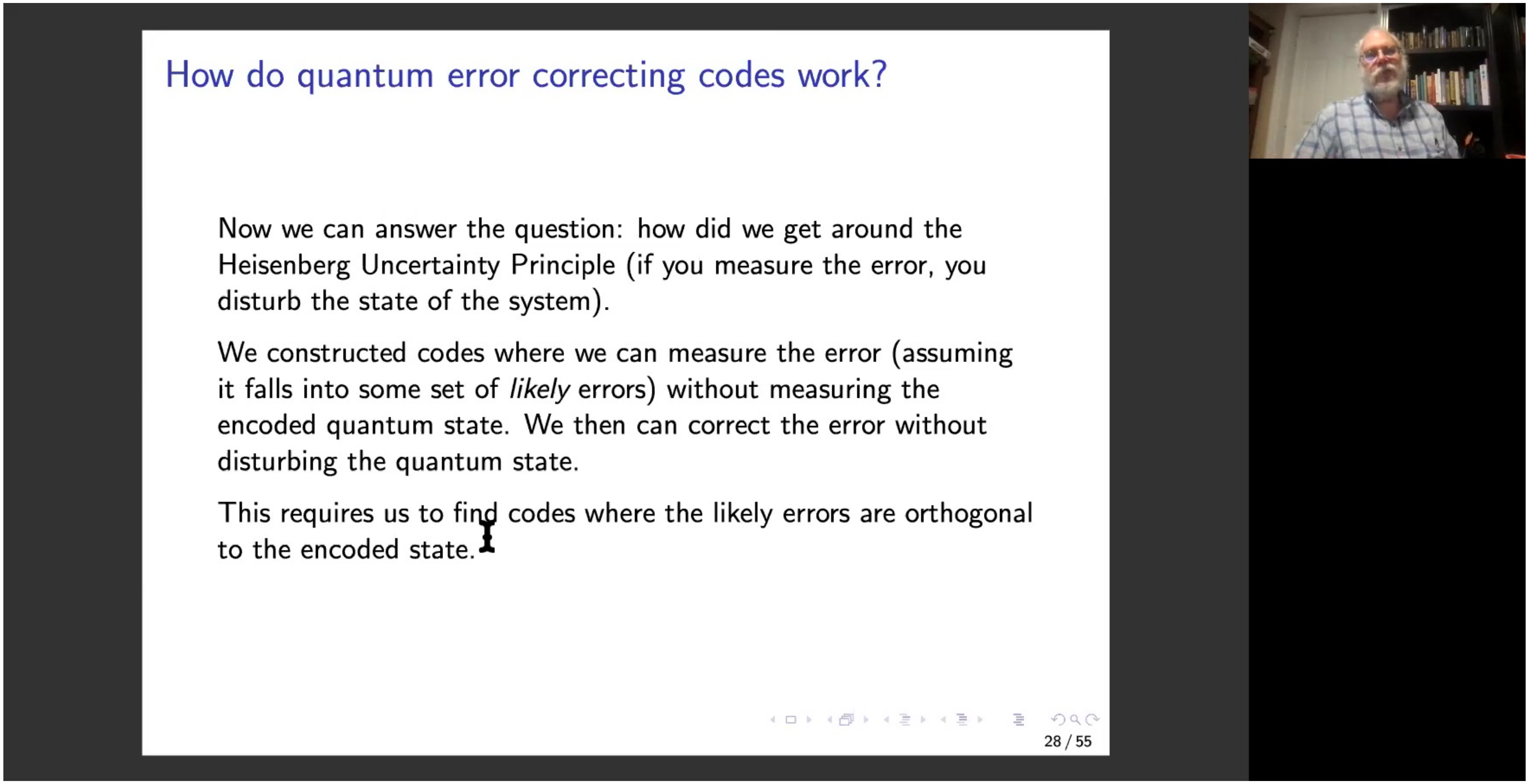
When we measure "which bit has a phase flip," we get "bit 1" with probability $|\sin^2\theta|$ and "no error" with probability $|\cos^2\theta|$. The state has "collapsed," so our measurement is now correct.



Small errors on every qubit

We can handle arbitrary errors on up to t qubits. How about small errors on every qubit?

Expand the errors in a Taylor series, and use the fact that most of the mass is in terms that have errors in t or fewer qubits.

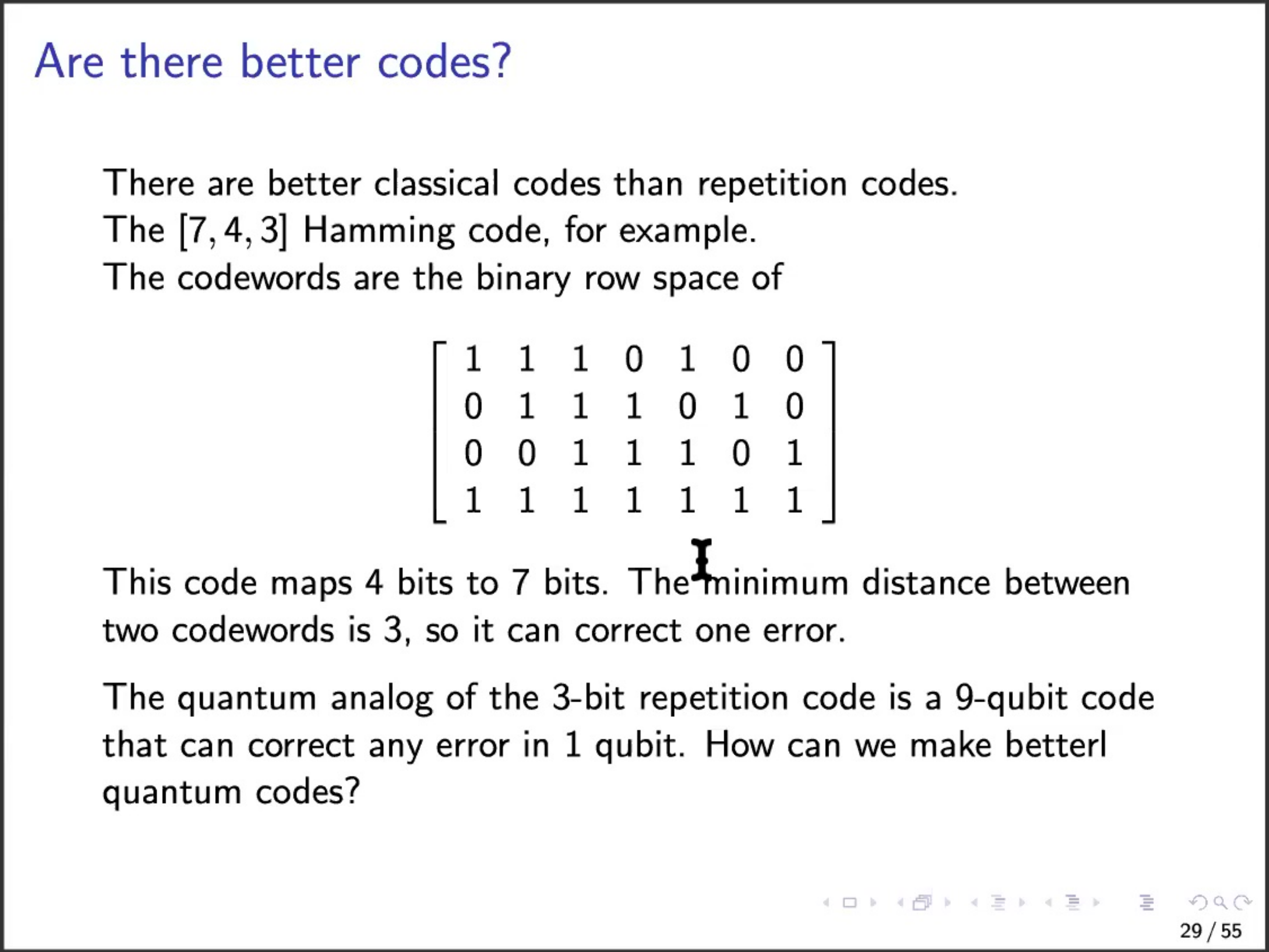


How do quantum error correcting codes work?

Now we can answer the question: how did we get around the Heisenberg Uncertainty Principle (if you measure the error, you disturb the state of the system).

We constructed codes where we can measure the error (assuming it falls into some set of *likely* errors) without measuring the encoded quantum state. We then can correct the error without disturbing the quantum state.

This requires us to find codes where the likely errors are orthogonal to the encoded state.



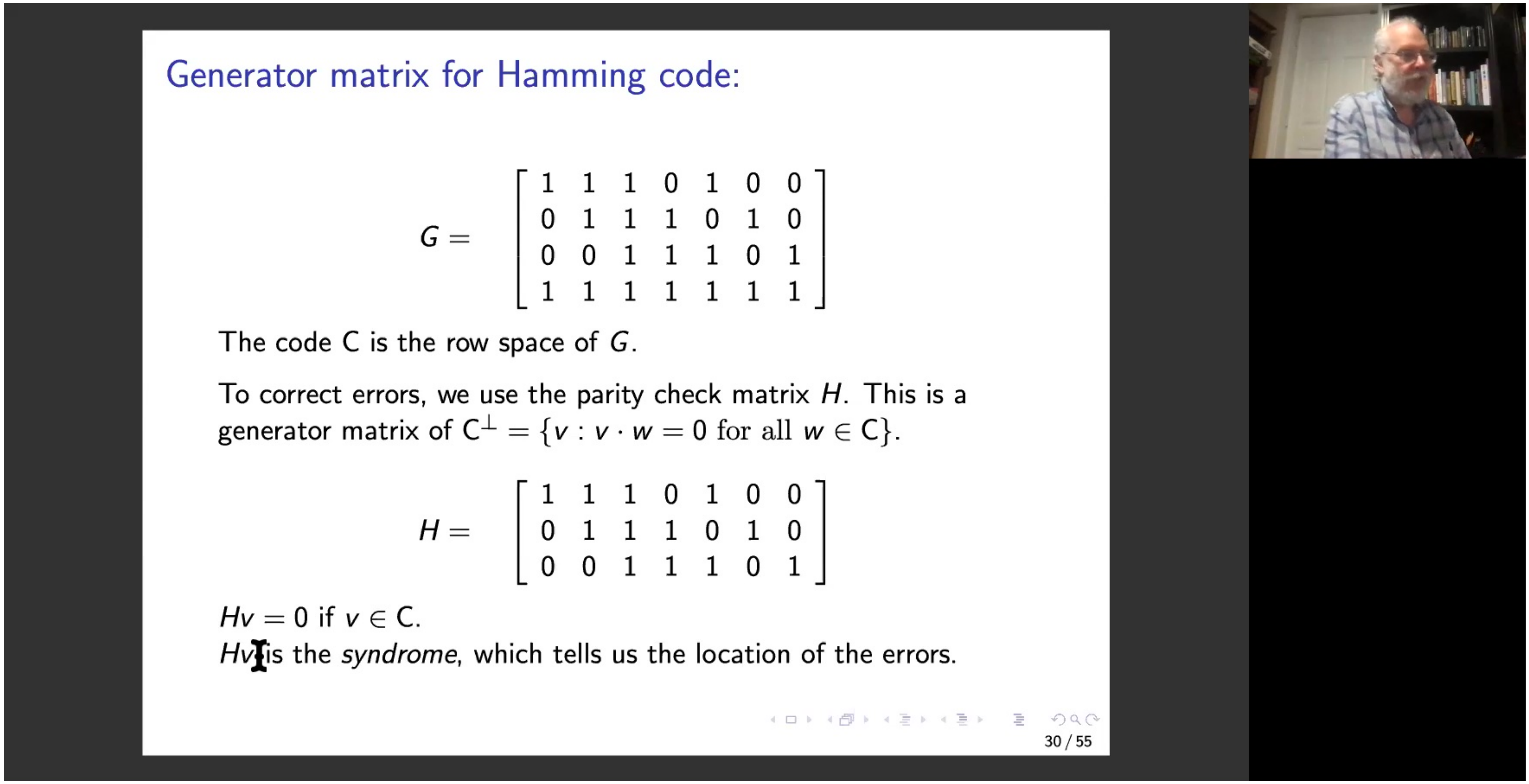
Are there better codes?

There are better classical codes than repetition codes.
The $[7, 4, 3]$ Hamming code, for example.
The codewords are the binary row space of

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

This code maps 4 bits to 7 bits. The minimum distance between two codewords is 3, so it can correct one error.

The quantum analog of the 3-bit repetition code is a 9-qubit code that can correct any error in 1 qubit. How can we make betterl quantum codes?



Generator matrix for Hamming code:

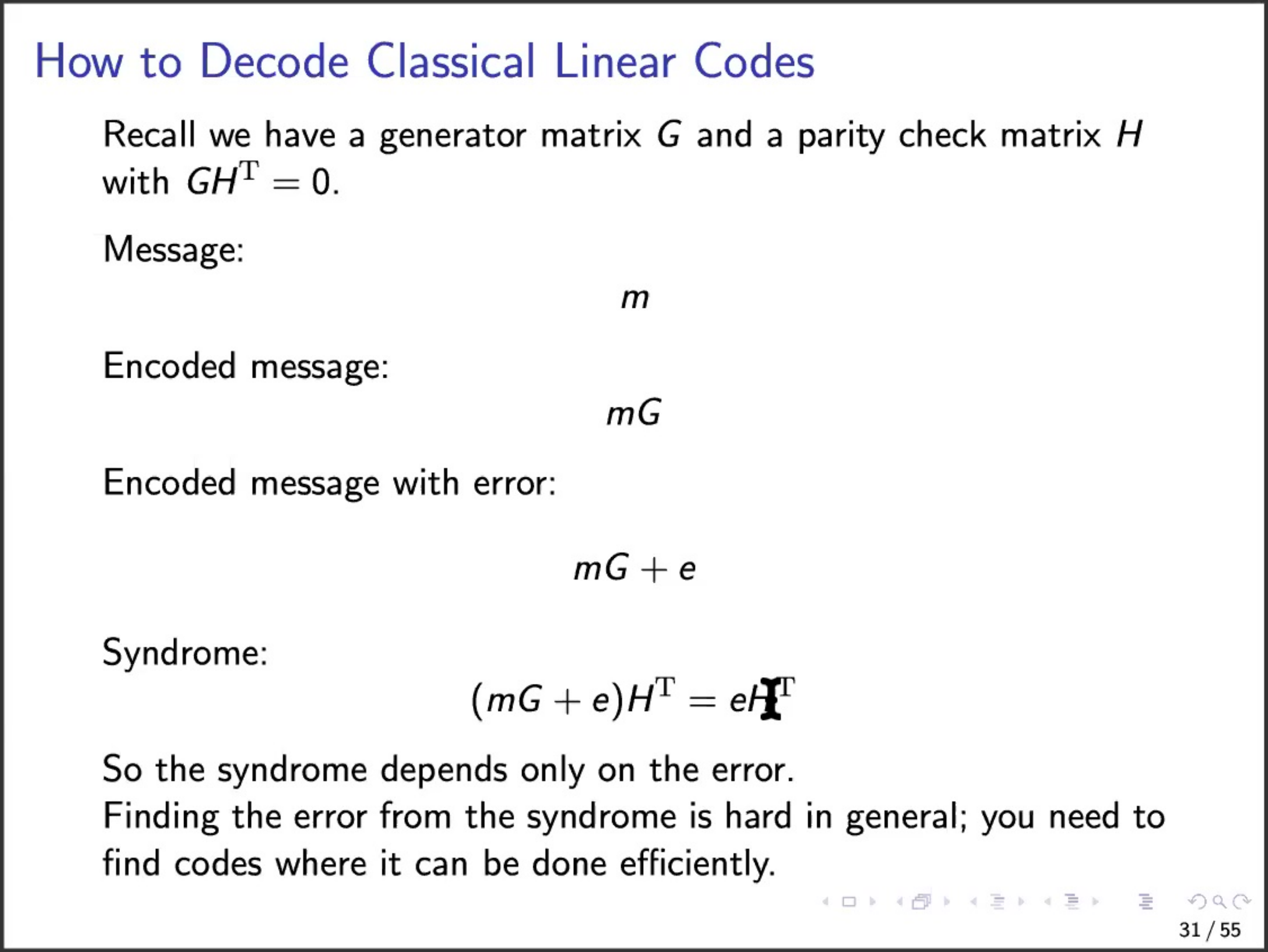
$$G = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

The code C is the row space of G.

To correct errors, we use the parity check matrix H. This is a generator matrix of $C^{\perp} = \{v : v \cdot w = 0 \text{ for all } w \in C\}$.

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

$Hv = 0$ if $v \in C$.
Hv is the *syndrome*, which tells us the location of the errors.



How to Decode Classical Linear Codes

Recall we have a generator matrix G and a parity check matrix H with $GH^{\mathrm{T}} = 0$.

Message:

$$m$$

Encoded message:

$$mG$$

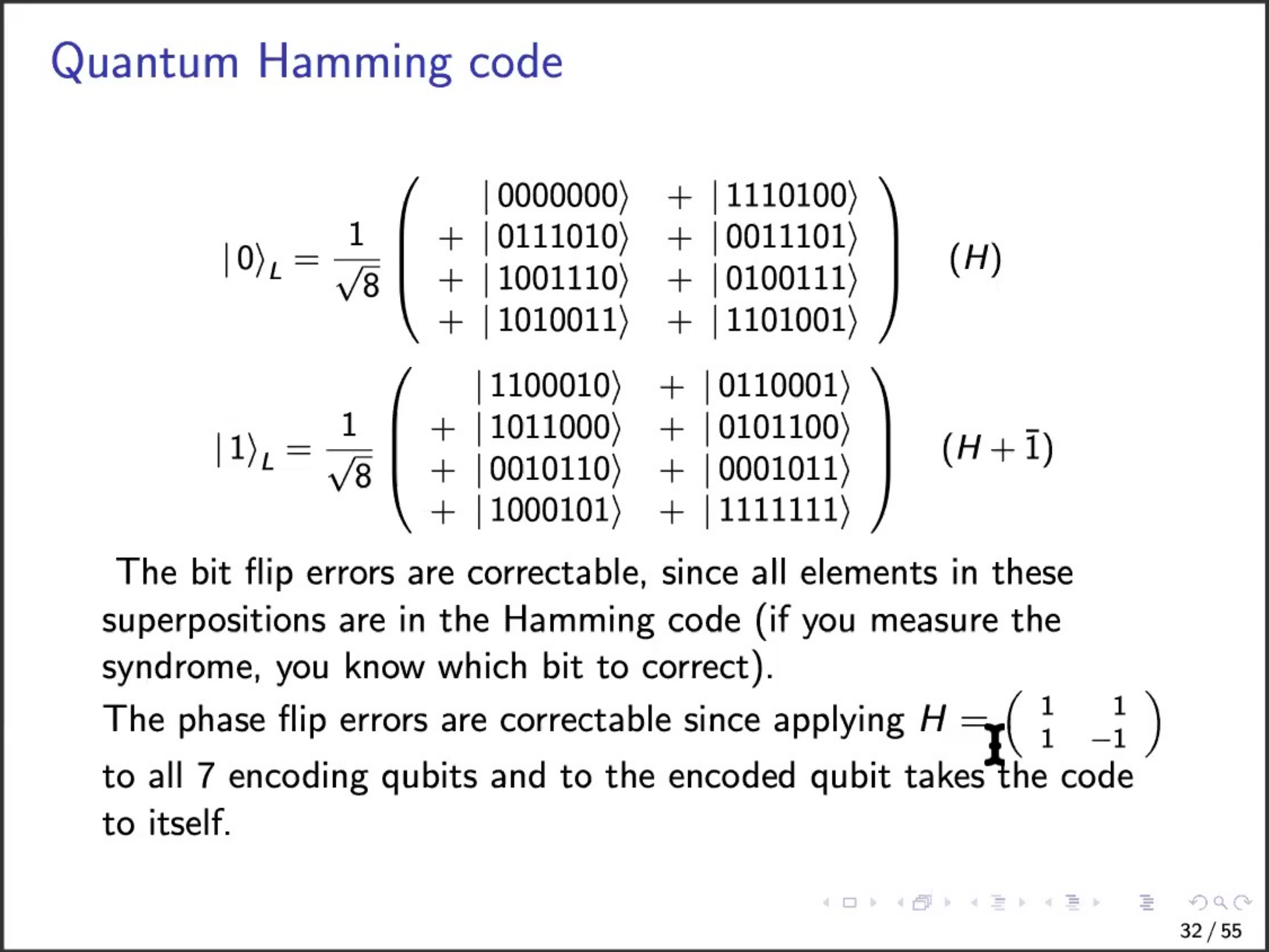
Encoded message with error:

$$mG + e$$

Syndrome:

$$(mG + e)H^{\mathrm{T}} = eH^{\mathrm{T}}$$

So the syndrome depends only on the error.
Finding the error from the syndrome is hard in general; you need to find codes where it can be done efficiently.

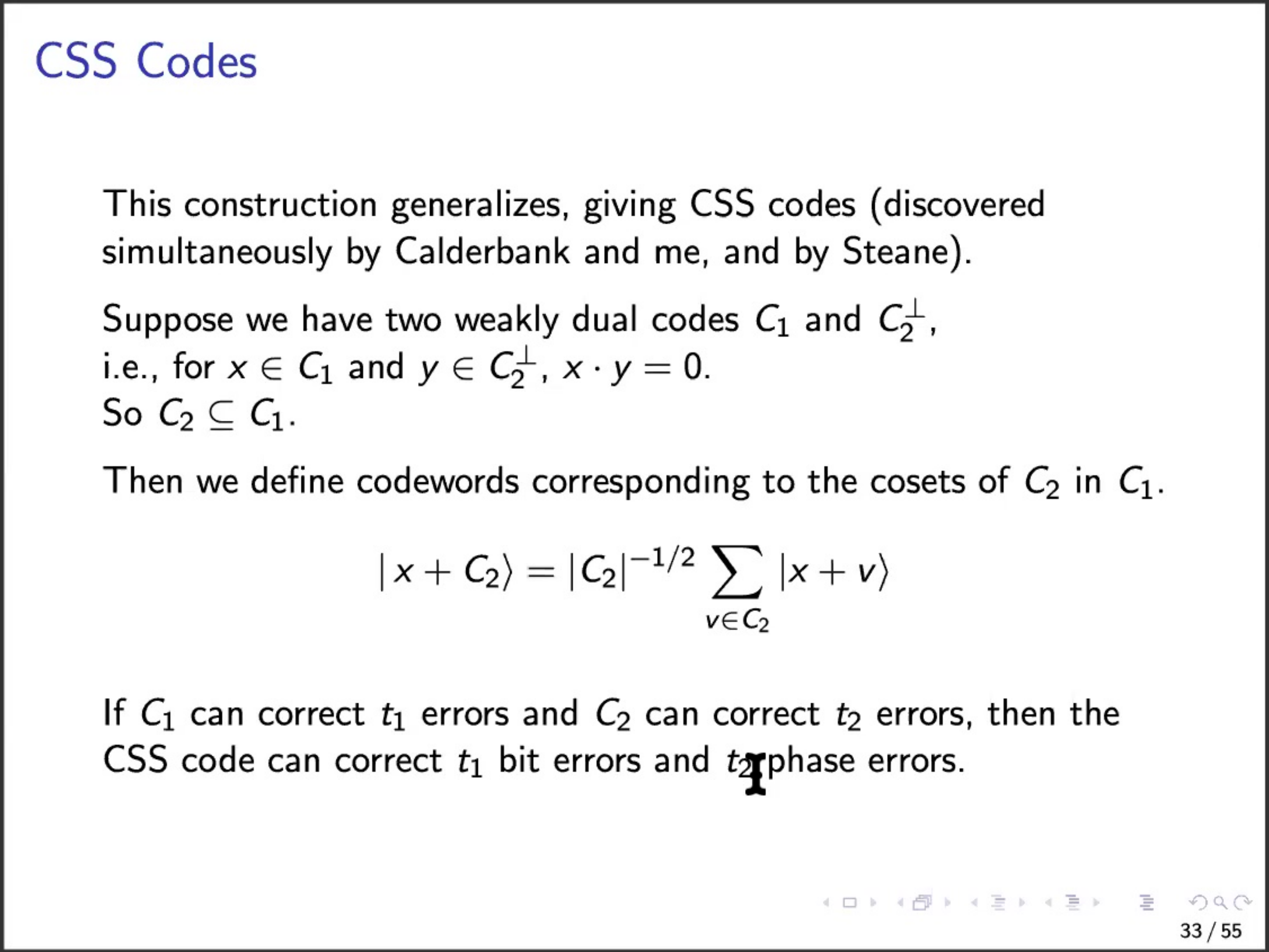


Quantum Hamming code

$$|0\rangle_L = \frac{1}{\sqrt{8}} \begin{pmatrix} |0000000\rangle & + |1110100\rangle \\ + |0111010\rangle & + |0011101\rangle \\ + |1001110\rangle & + |0100111\rangle \\ + |1010011\rangle & + |1101001\rangle \end{pmatrix} \quad (H)$$

$$|1\rangle_L = \frac{1}{\sqrt{8}} \begin{pmatrix} |1100010\rangle & + |0110001\rangle \\ + |1011000\rangle & + |0101100\rangle \\ + |0010110\rangle & + |0001011\rangle \\ + |1000101\rangle & + |1111111\rangle \end{pmatrix} \quad (H + \bar{1})$$

The bit flip errors are correctable, since all elements in these superpositions are in the Hamming code (if you measure the syndrome, you know which bit to correct).
The phase flip errors are correctable since applying $H = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ to all 7 encoding qubits and to the encoded qubit takes the code to itself.



CSS Codes

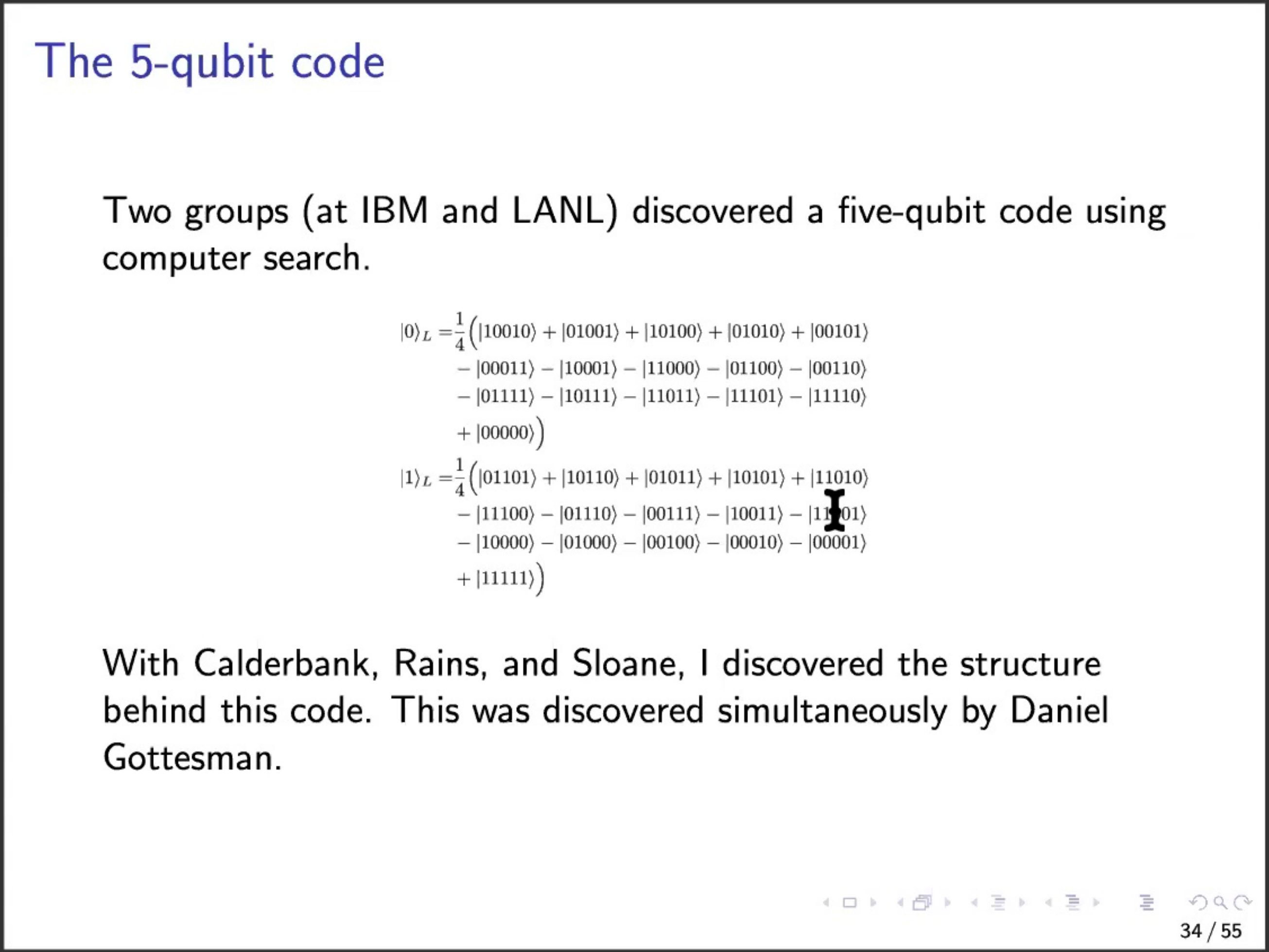
This construction generalizes, giving CSS codes (discovered simultaneously by Calderbank and me, and by Steane).

Suppose we have two weakly dual codes C_1 and $C_2^\perp$,
i.e., for $x \in C_1$ and $y \in C_2^\perp$, $x \cdot y = 0$.
So $C_2 \subseteq C_1$.

Then we define codewords corresponding to the cosets of C_2 in C_1.

$$|x + C_2\rangle = |C_2|^{-1/2} \sum_{v \in C_2} |x + v\rangle$$

If C_1 can correct t_1 errors and C_2 can correct t_2 errors, then the CSS code can correct t_1 bit errors and t_2 phase errors.

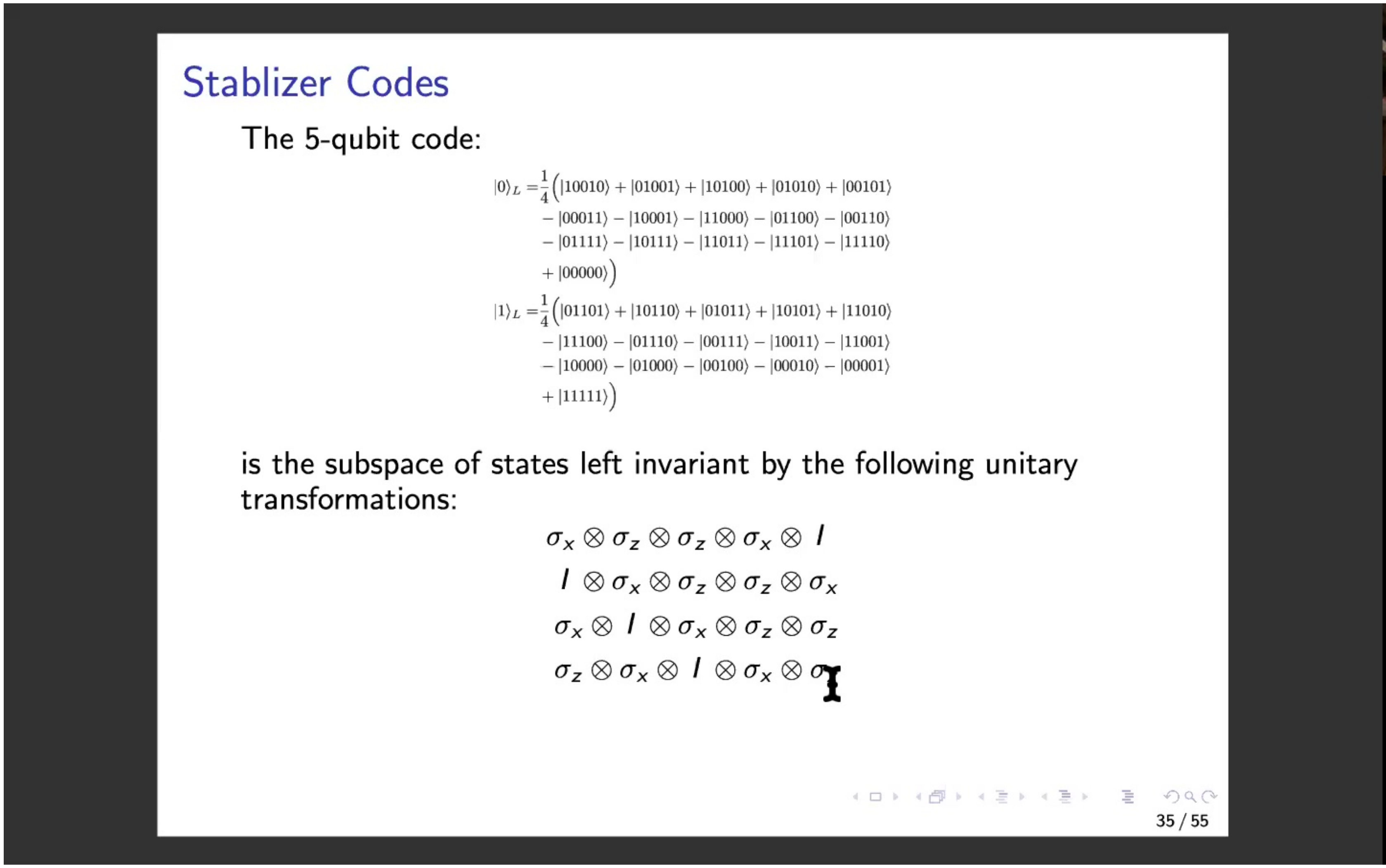


The 5-qubit code

Two groups (at IBM and LANL) discovered a five-qubit code using computer search.

$$
\begin{aligned}
|0\rangle_L = \frac{1}{4}\Big(& |10010\rangle + |01001\rangle + |10100\rangle + |01010\rangle + |00101\rangle \\
& - |00011\rangle - |10001\rangle - |11000\rangle - |01100\rangle - |00110\rangle \\
& - |01111\rangle - |10111\rangle - |11011\rangle - |11101\rangle - |11110\rangle \\
& + |00000\rangle \Big) \\
|1\rangle_L = \frac{1}{4}\Big(& |01101\rangle + |10110\rangle + |01011\rangle + |10101\rangle + |11010\rangle \\
& - |11100\rangle - |01110\rangle - |00111\rangle - |10011\rangle - |11001\rangle \\
& - |10000\rangle - |01000\rangle - |00100\rangle - |00010\rangle - |00001\rangle \\
& + |11111\rangle \Big)
\end{aligned}
$$

With Calderbank, Rains, and Sloane, I discovered the structure behind this code. This was discovered simultaneously by Daniel Gottesman.



Stablizer Codes

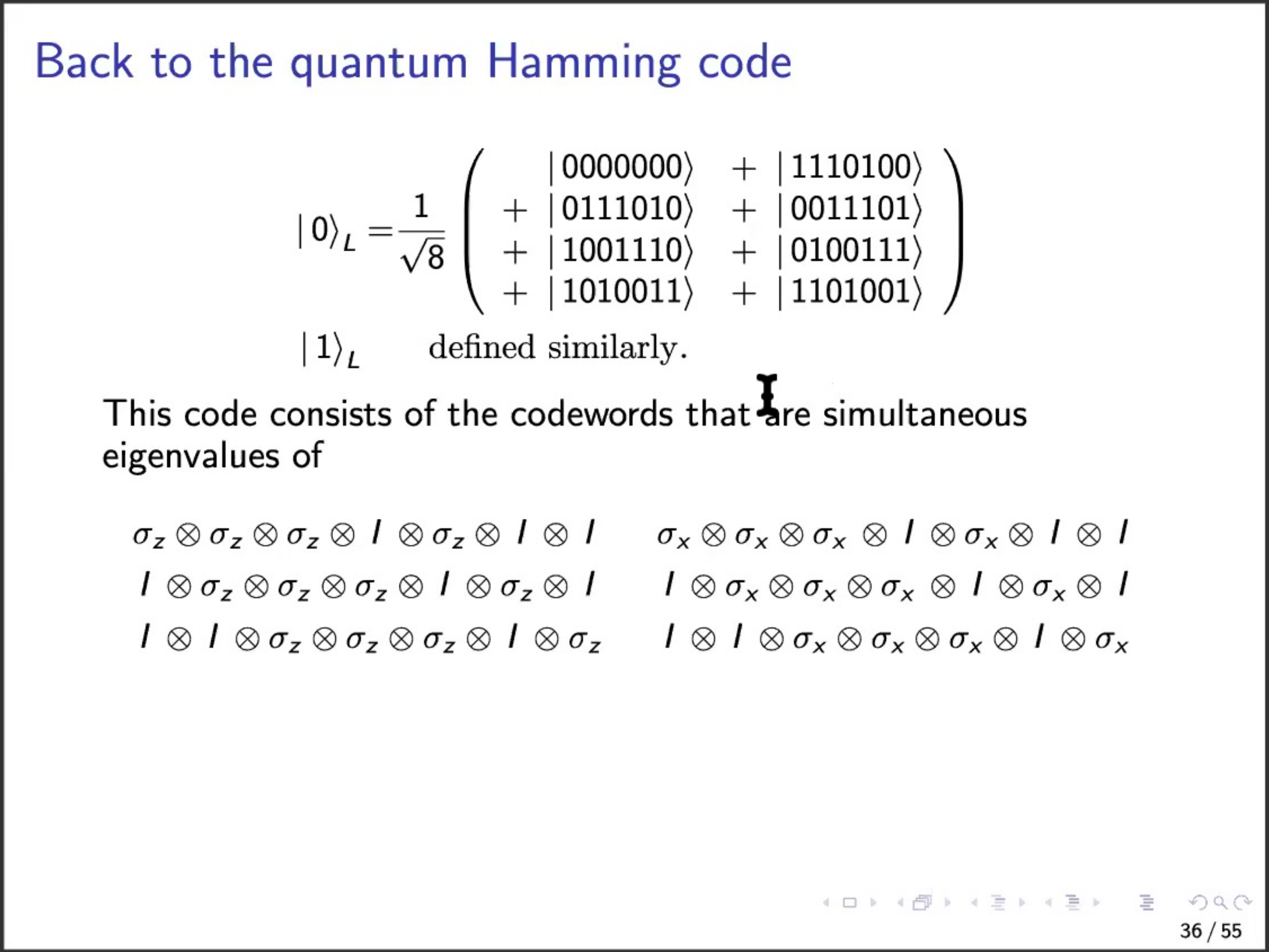
The 5-qubit code:

$$
\begin{aligned}
|0\rangle_L = \frac{1}{4}\Big(& |10010\rangle + |01001\rangle + |10100\rangle + |01010\rangle + |00101\rangle \\
& - |00011\rangle - |10001\rangle - |11000\rangle - |01100\rangle - |00110\rangle \\
& - |01111\rangle - |10111\rangle - |11011\rangle - |11101\rangle - |11110\rangle \\
& + |00000\rangle \Big) \\
|1\rangle_L = \frac{1}{4}\Big(& |01101\rangle + |10110\rangle + |01011\rangle + |10101\rangle + |11010\rangle \\
& - |11100\rangle - |01110\rangle - |00111\rangle - |10011\rangle - |11001\rangle \\
& - |10000\rangle - |01000\rangle - |00100\rangle - |00010\rangle - |00001\rangle \\
& + |11111\rangle \Big)
\end{aligned}
$$

is the subspace of states left invariant by the following unitary transformations:

$$
\begin{gathered}
\sigma_x \otimes \sigma_z \otimes \sigma_z \otimes \sigma_x \otimes I \\
I \otimes \sigma_x \otimes \sigma_z \otimes \sigma_z \otimes \sigma_x \\
\sigma_x \otimes I \otimes \sigma_x \otimes \sigma_z \otimes \sigma_z \\
\sigma_z \otimes \sigma_x \otimes I \otimes \sigma_x \otimes \sigma
\end{gathered}
$$

35 / 55



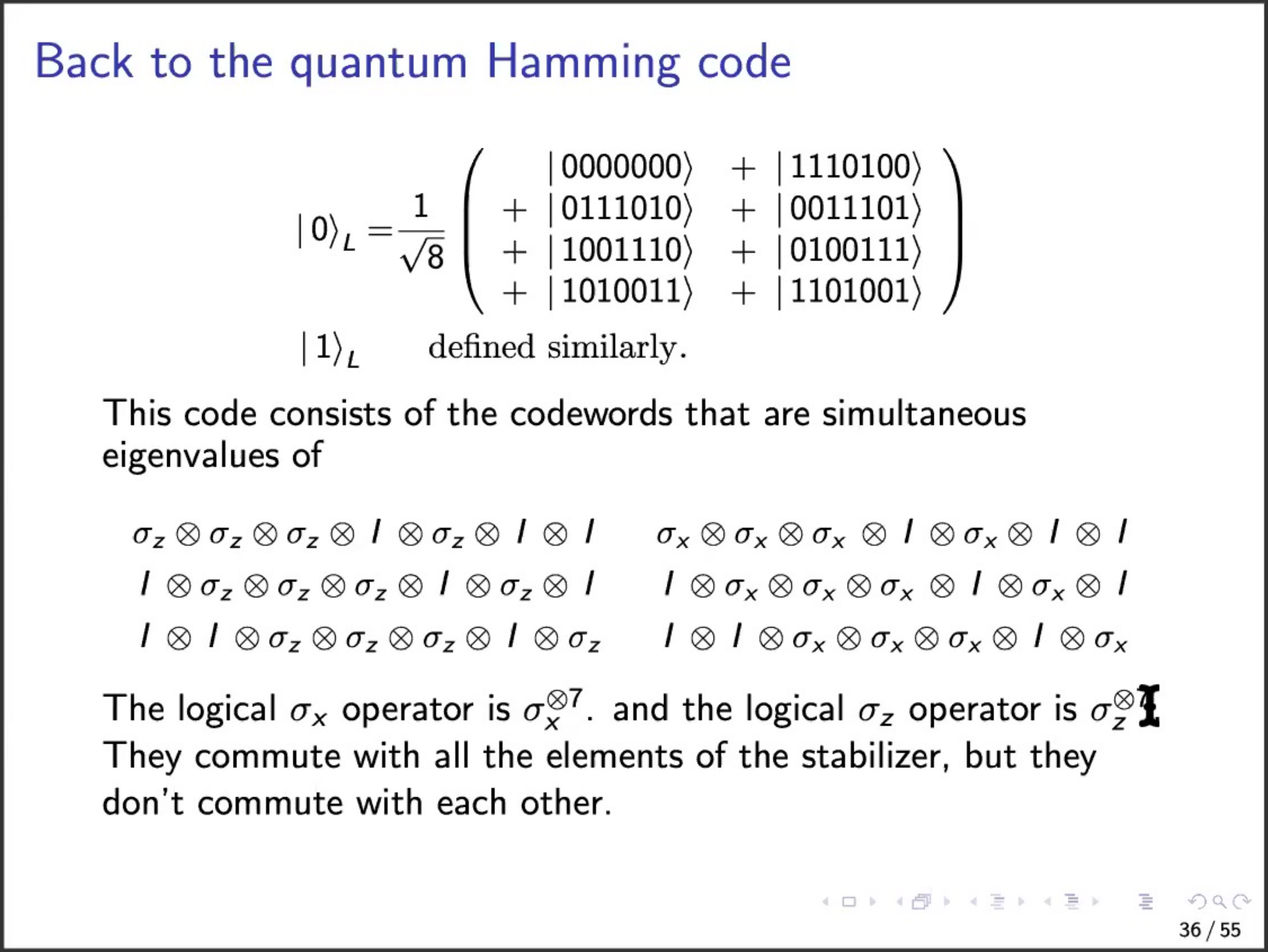
Back to the quantum Hamming code

$$|0\rangle_L = \frac{1}{\sqrt{8}} \begin{pmatrix} & |0000000\rangle & + |1110100\rangle \\ + & |0111010\rangle & + |0011101\rangle \\ + & |1001110\rangle & + |0100111\rangle \\ + & |1010011\rangle & + |1101001\rangle \end{pmatrix}$$

$|1\rangle_L$ defined similarly.

This code consists of the codewords that are simultaneous eigenvalues of

$$\begin{array}{ll} \sigma_z \otimes \sigma_z \otimes \sigma_z \otimes I \otimes \sigma_z \otimes I \otimes I & \sigma_x \otimes \sigma_x \otimes \sigma_x \otimes I \otimes \sigma_x \otimes I \otimes I \\ I \otimes \sigma_z \otimes \sigma_z \otimes \sigma_z \otimes I \otimes \sigma_z \otimes I & I \otimes \sigma_x \otimes \sigma_x \otimes \sigma_x \otimes I \otimes \sigma_x \otimes I \\ I \otimes I \otimes \sigma_z \otimes \sigma_z \otimes \sigma_z \otimes I \otimes \sigma_z & I \otimes I \otimes \sigma_x \otimes \sigma_x \otimes \sigma_x \otimes I \otimes \sigma_x \end{array}$$



Back to the quantum Hamming code

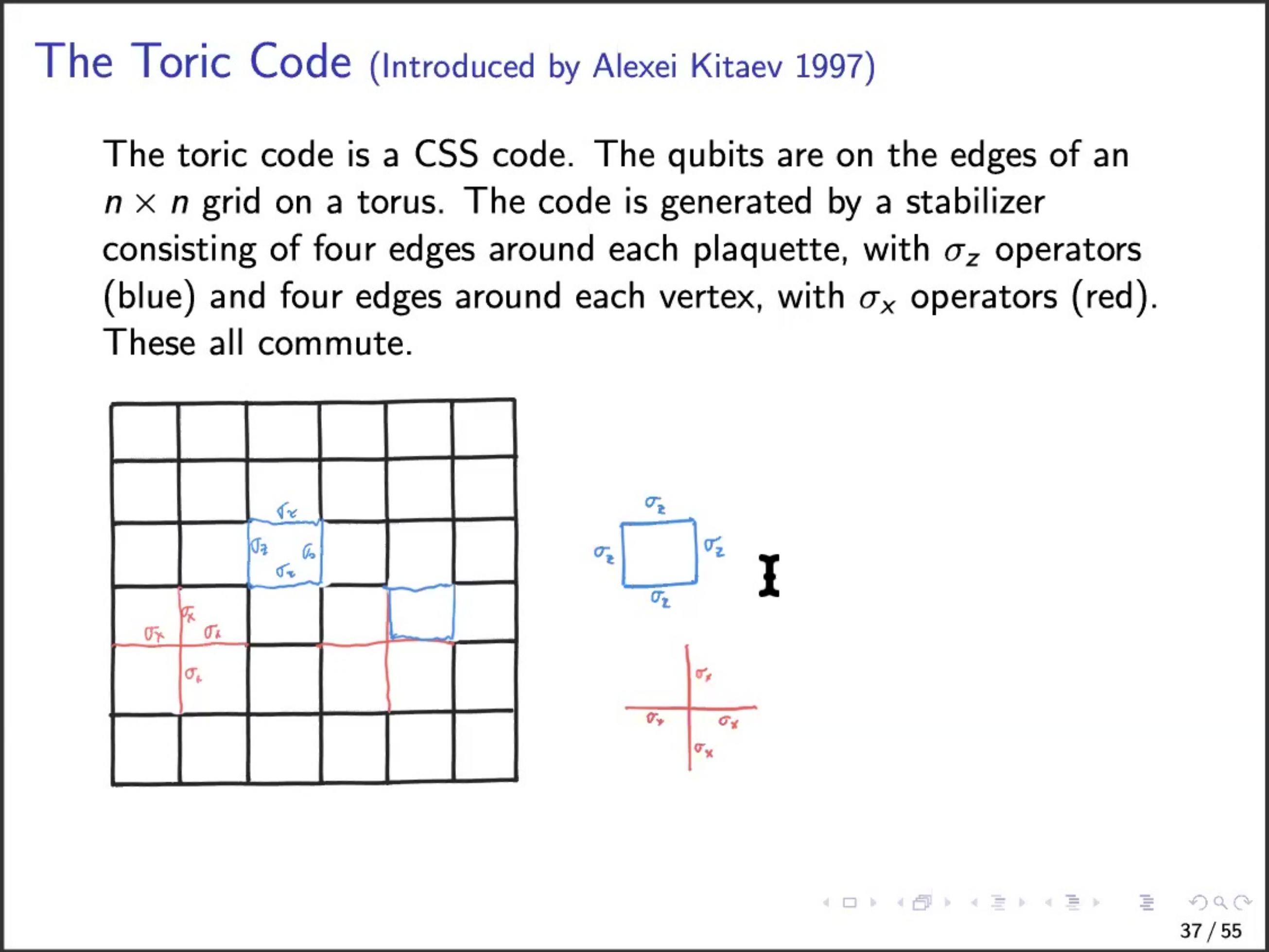
$$|0\rangle_L = \frac{1}{\sqrt{8}} \begin{pmatrix} & |0000000\rangle & + & |1110100\rangle \\ + & |0111010\rangle & + & |0011101\rangle \\ + & |1001110\rangle & + & |0100111\rangle \\ + & |1010011\rangle & + & |1101001\rangle \end{pmatrix}$$

$|1\rangle_L$ defined similarly.

This code consists of the codewords that are simultaneous eigenvalues of

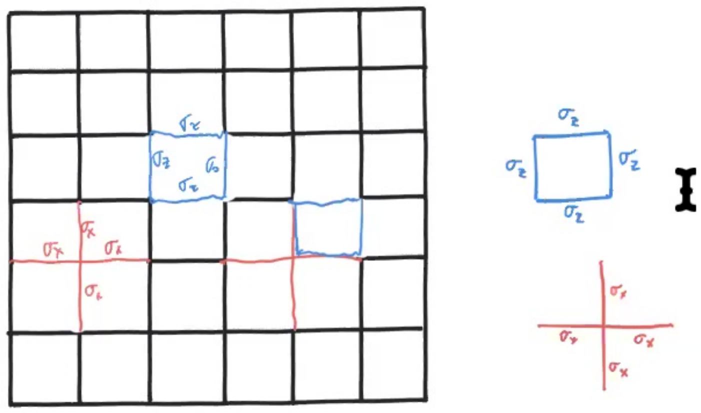
$$\begin{array}{ll} \sigma_z \otimes \sigma_z \otimes \sigma_z \otimes I \otimes \sigma_z \otimes I \otimes I & \sigma_x \otimes \sigma_x \otimes \sigma_x \otimes I \otimes \sigma_x \otimes I \otimes I \\ I \otimes \sigma_z \otimes \sigma_z \otimes \sigma_z \otimes I \otimes \sigma_z \otimes I & I \otimes \sigma_x \otimes \sigma_x \otimes \sigma_x \otimes I \otimes \sigma_x \otimes I \\ I \otimes I \otimes \sigma_z \otimes \sigma_z \otimes \sigma_z \otimes I \otimes \sigma_z & I \otimes I \otimes \sigma_x \otimes \sigma_x \otimes \sigma_x \otimes I \otimes \sigma_x \end{array}$$

The logical σ_x operator is $\sigma_x^{\otimes 7}$. and the logical σ_z operator is $\sigma_z^{\otimes 7}$. They commute with all the elements of the stabilizer, but they don't commute with each other.



The Toric Code (Introduced by Alexei Kitaev 1997)

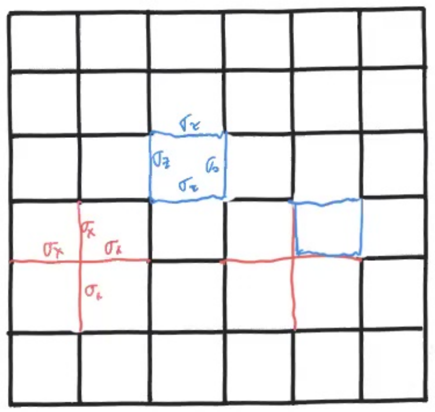
The toric code is a CSS code. The qubits are on the edges of an $n \times n$ grid on a torus. The code is generated by a stabilizer consisting of four edges around each plaquette, with σ_z operators (blue) and four edges around each vertex, with σ_x operators (red). These all commute.

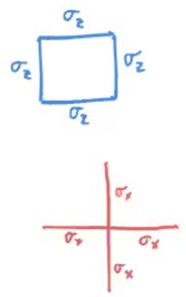


37 / 55

The Toric Code (Introduced by Alexei Kitaev 1997)

The toric code is a CSS code. The qubits are on the edges of an $n \times n$ grid on a torus. The code is generated by a stabilizer consisting of four edges around each plaquette, with σ_z operators (blue) and four edges around each vertex, with σ_x operators (red). These all commute.





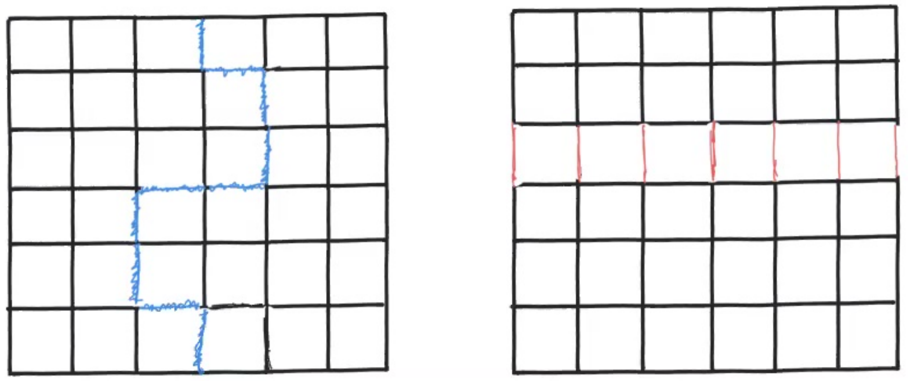
The stabilizer contains $n^2 - 1$ linearly independent plaquette operators

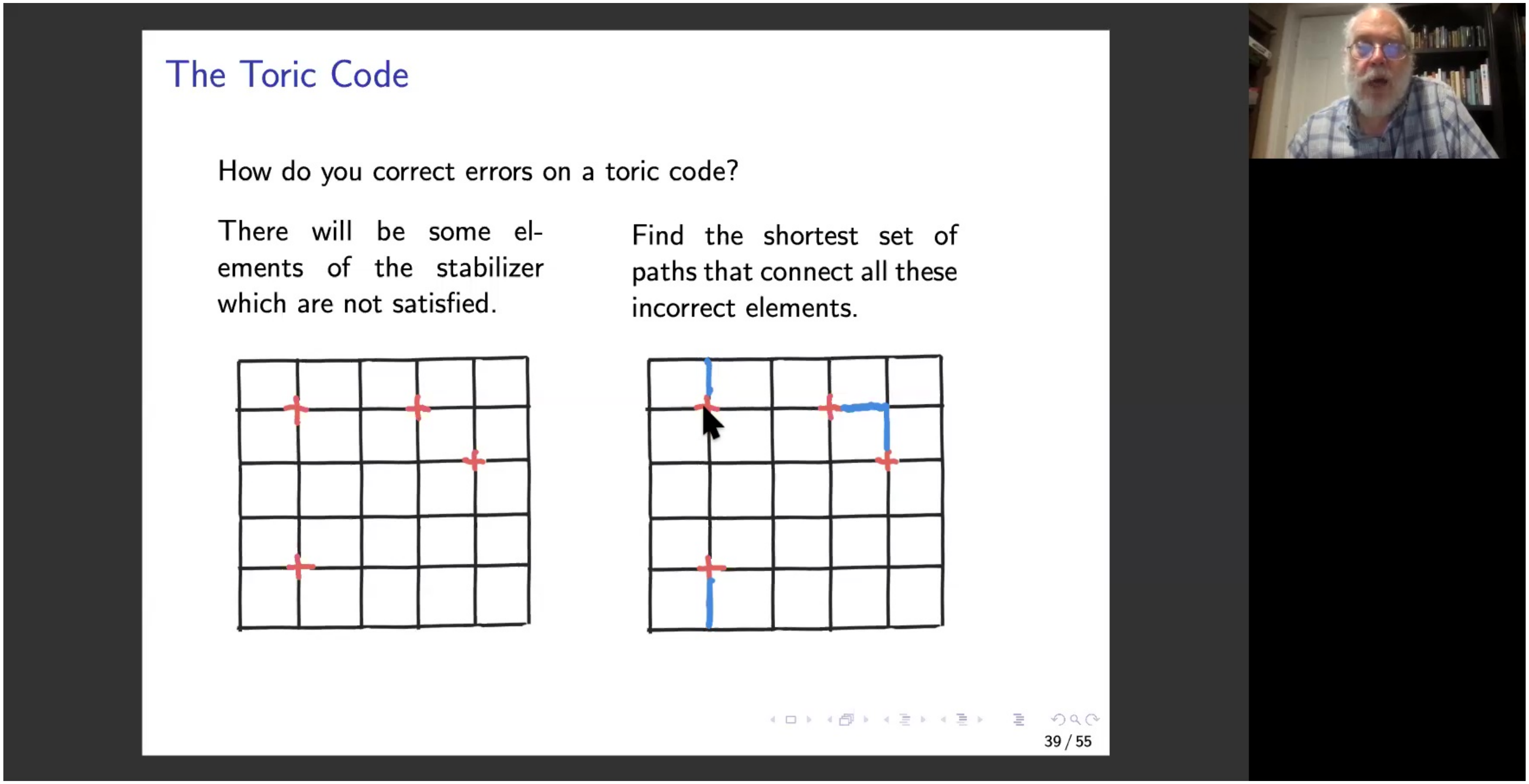
and $n^2 - 1$ linearly independent vertex operators.

This means that it encodes $2n^2 - 2(n^2 - 1) = 2$ qubits.

The Toric Code

The toric code encodes two qubits. The logical σ_z operators are chains of σ_z's on edges that wrap around the torus, and the logical σ_x operators chains of σ_x's on edges that wrap around the torus.

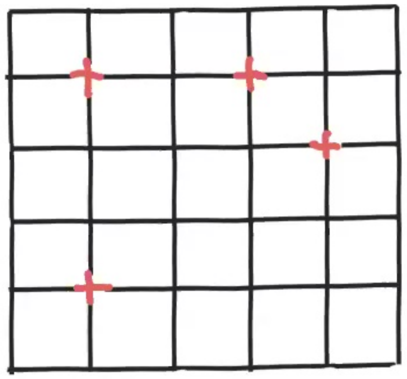




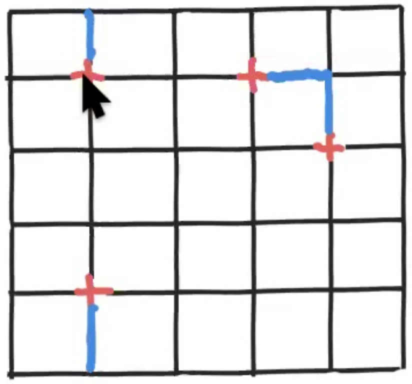
The Toric Code

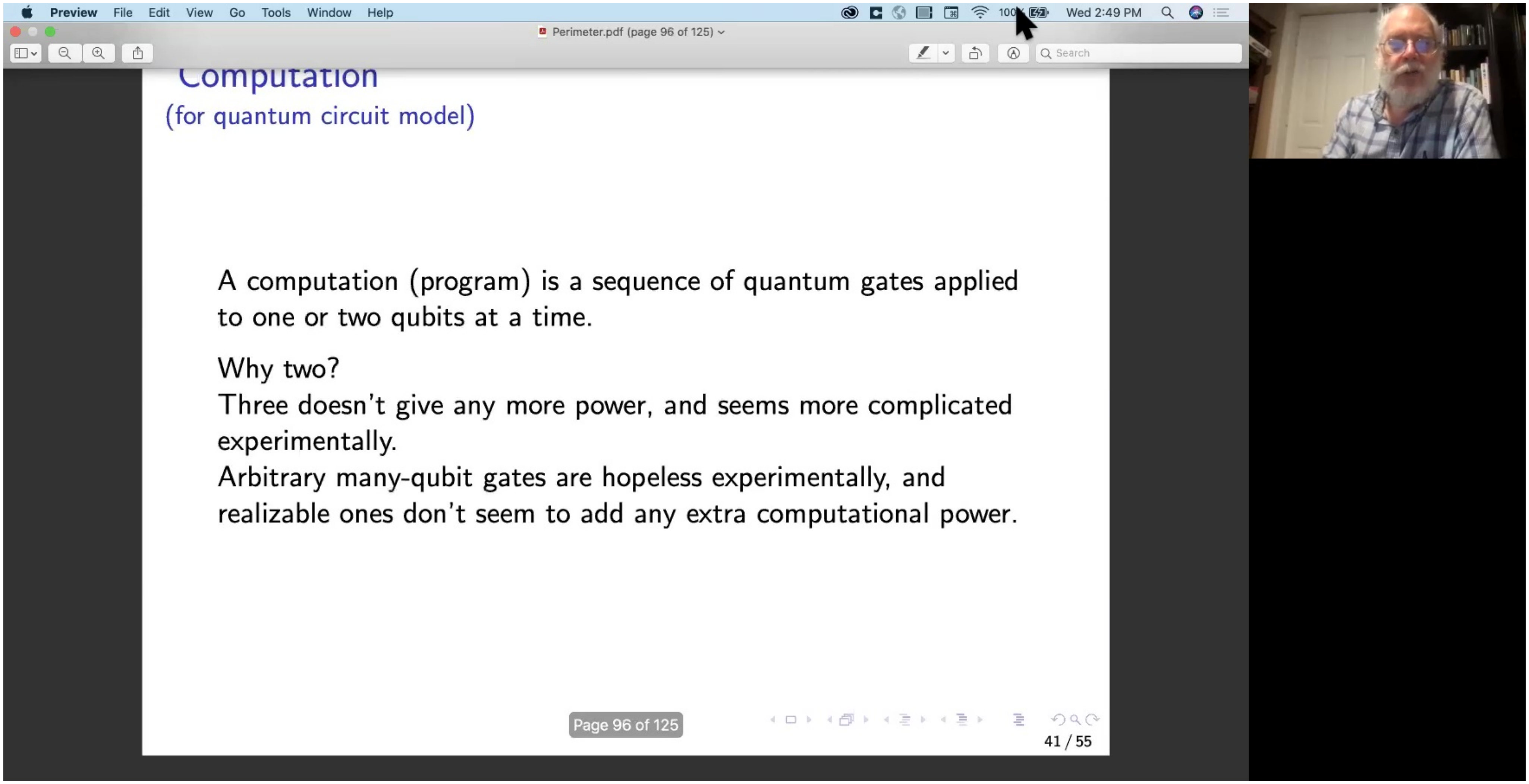
How do you correct errors on a toric code?

There will be some elements of the stabilizer which are not satisfied.



Find the shortest set of paths that connect all these incorrect elements.



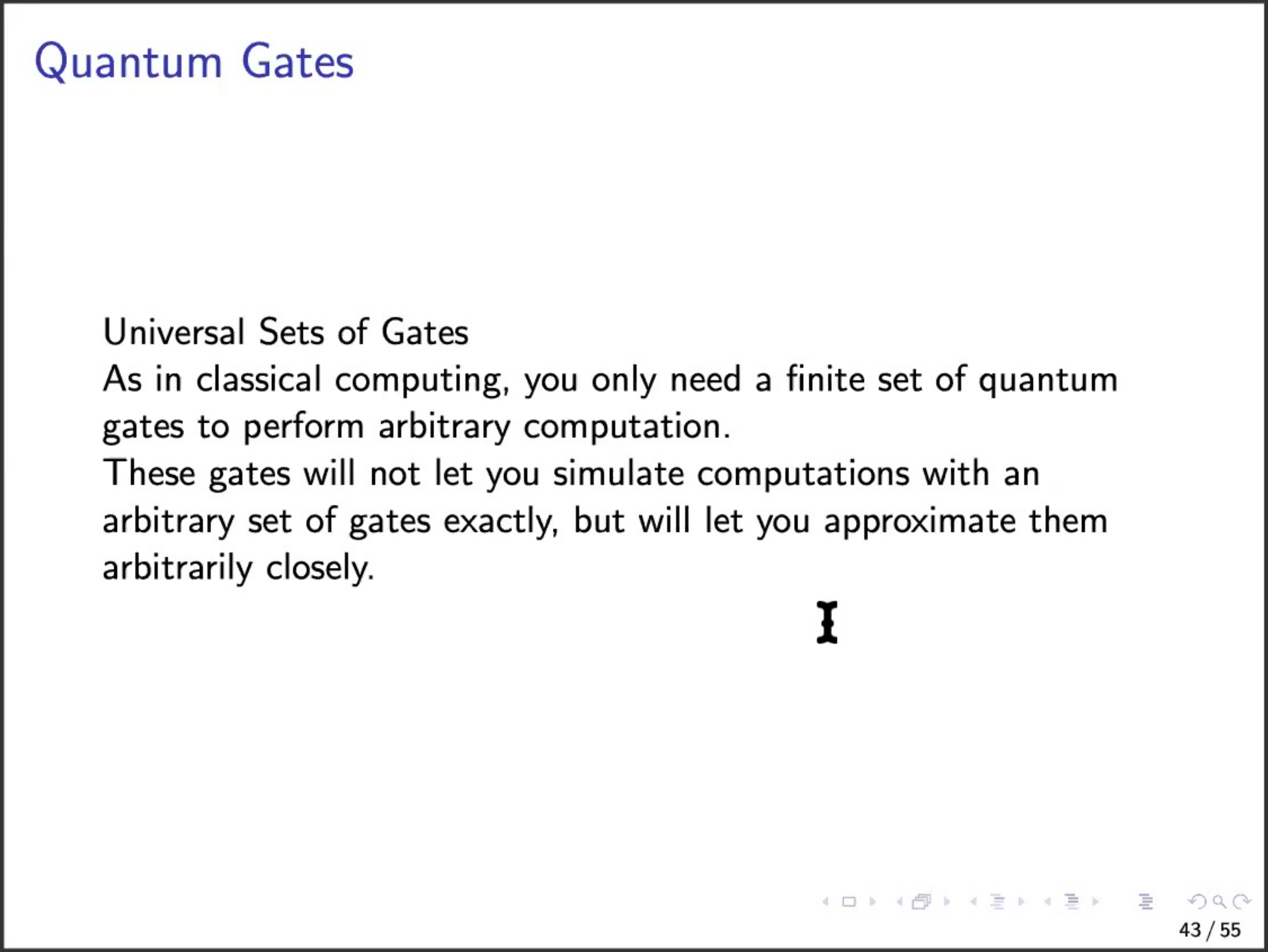

Preview File Edit View Go Tools Window Help
Wed 2:49 PM
Perimeter.pdf (page 96 of 125)
Search

Computation
(for quantum circuit model)

A computation (program) is a sequence of quantum gates applied to one or two qubits at a time.

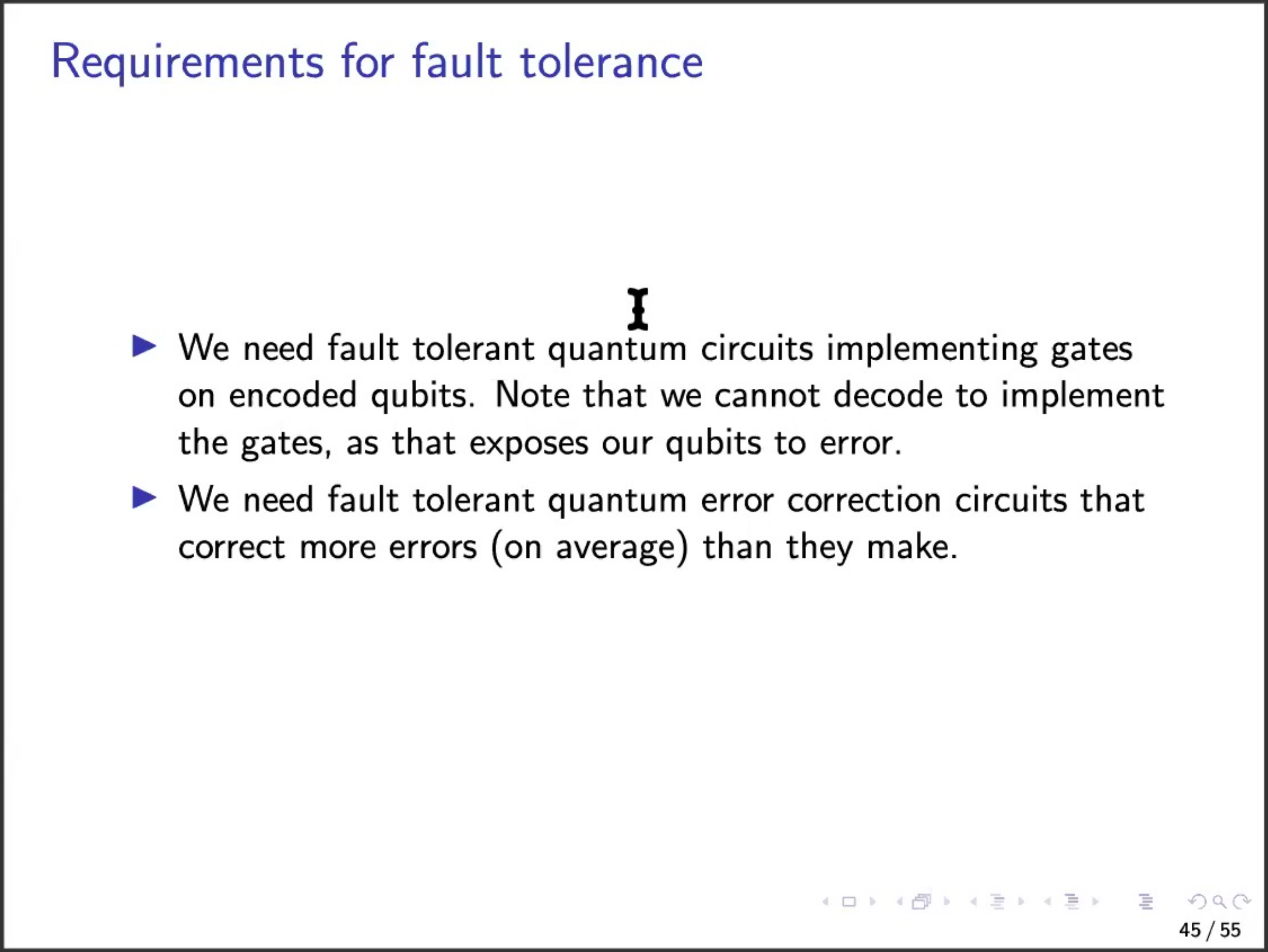
Why two?
Three doesn't give any more power, and seems more complicated experimentally.
Arbitrary many-qubit gates are hopeless experimentally, and realizable ones don't seem to add any extra computational power.

41 / 55



Quantum Gates

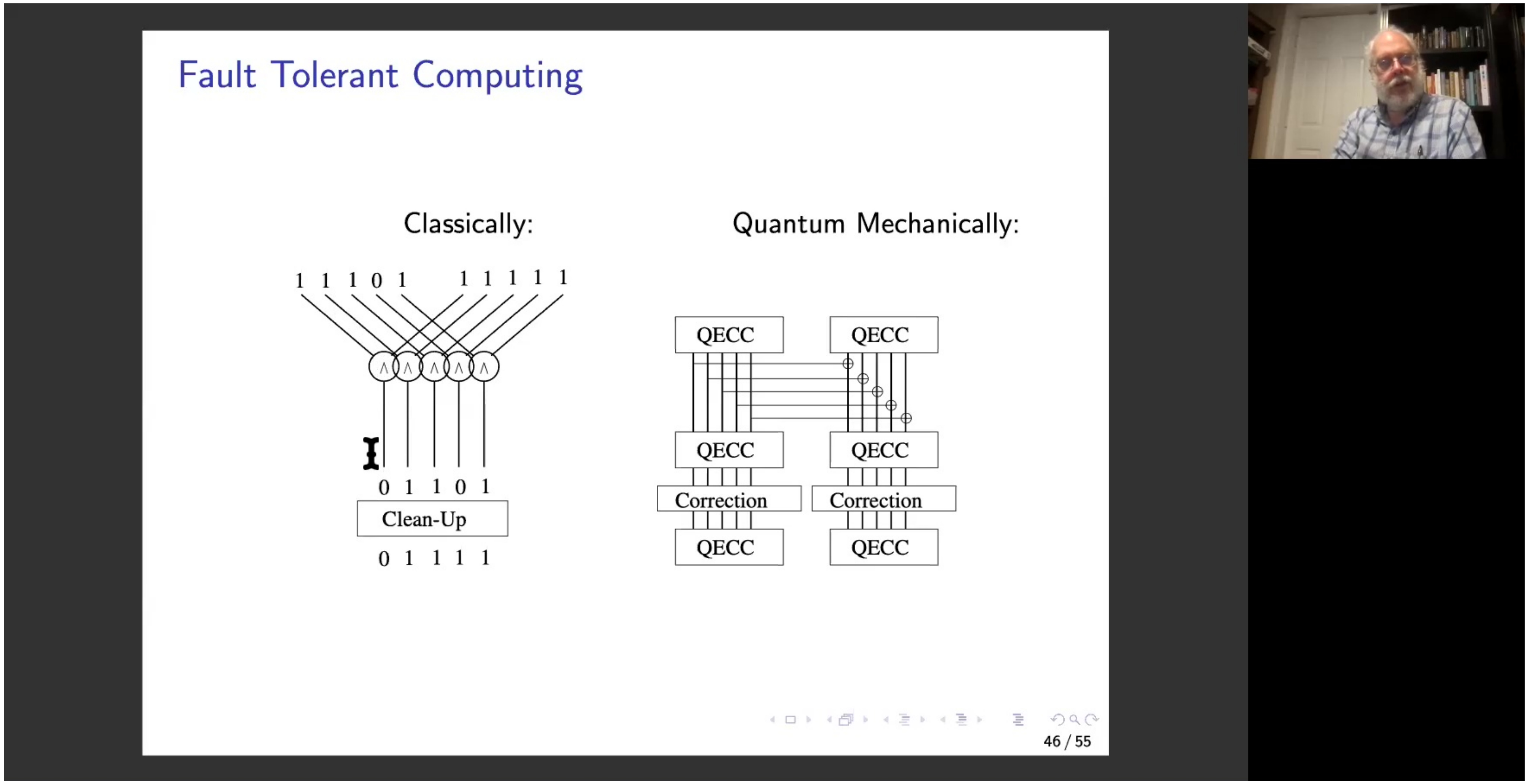
Universal Sets of Gates
As in classical computing, you only need a finite set of quantum gates to perform arbitrary computation.
These gates will not let you simulate computations with an arbitrary set of gates exactly, but will let you approximate them arbitrarily closely.



Requirements for fault tolerance

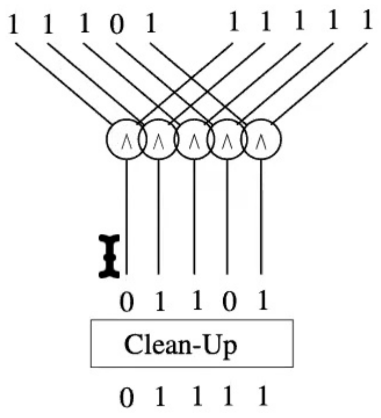
- We need fault tolerant quantum circuits implementing gates on encoded qubits. Note that we cannot decode to implement the gates, as that exposes our qubits to error.
- We need fault tolerant quantum error correction circuits that correct more errors (on average) than they make.

45 / 55

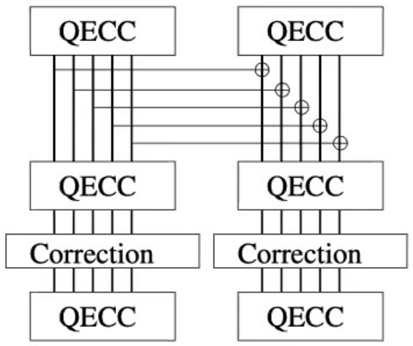


Fault Tolerant Computing

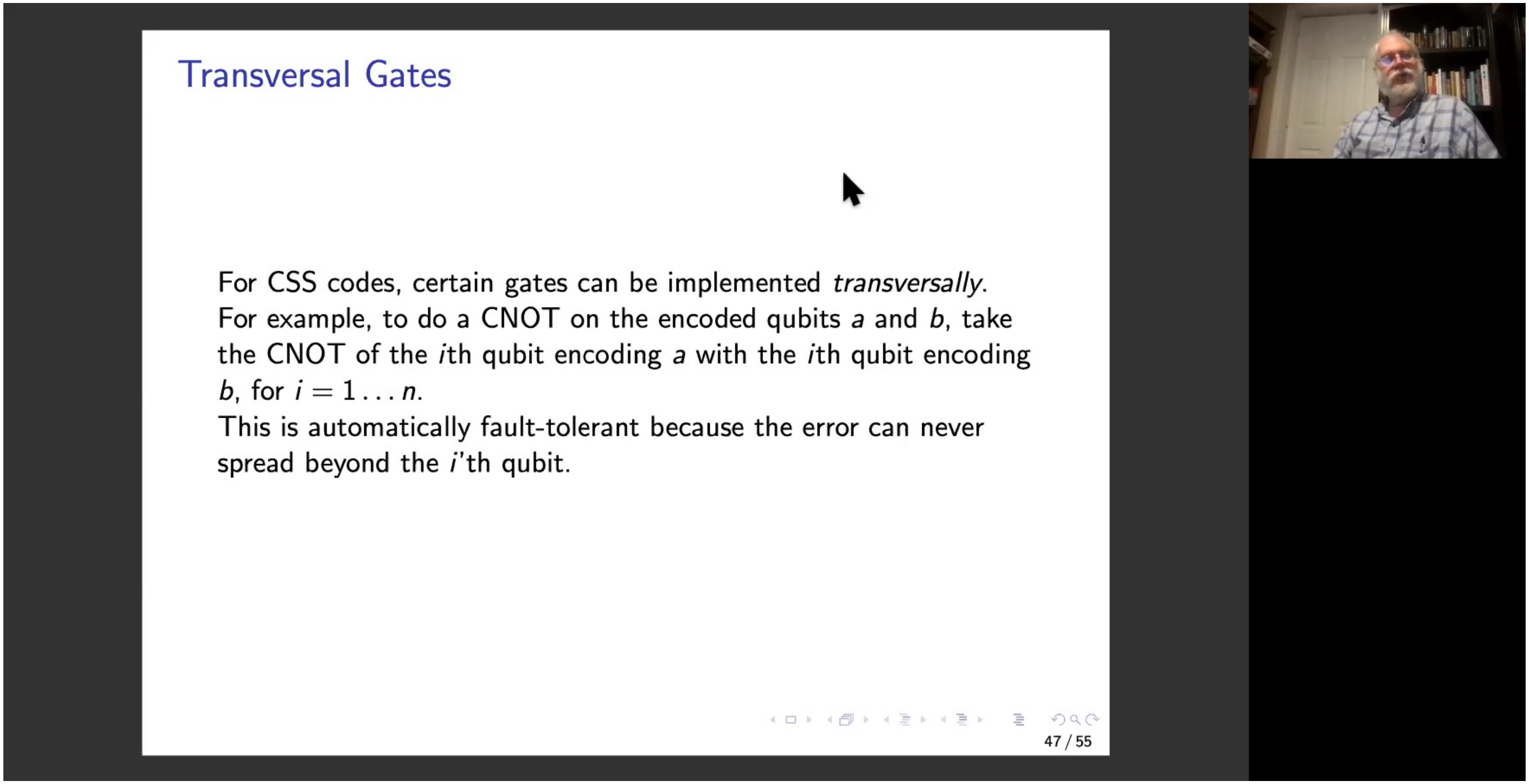
Classically:



Quantum Mechanically:

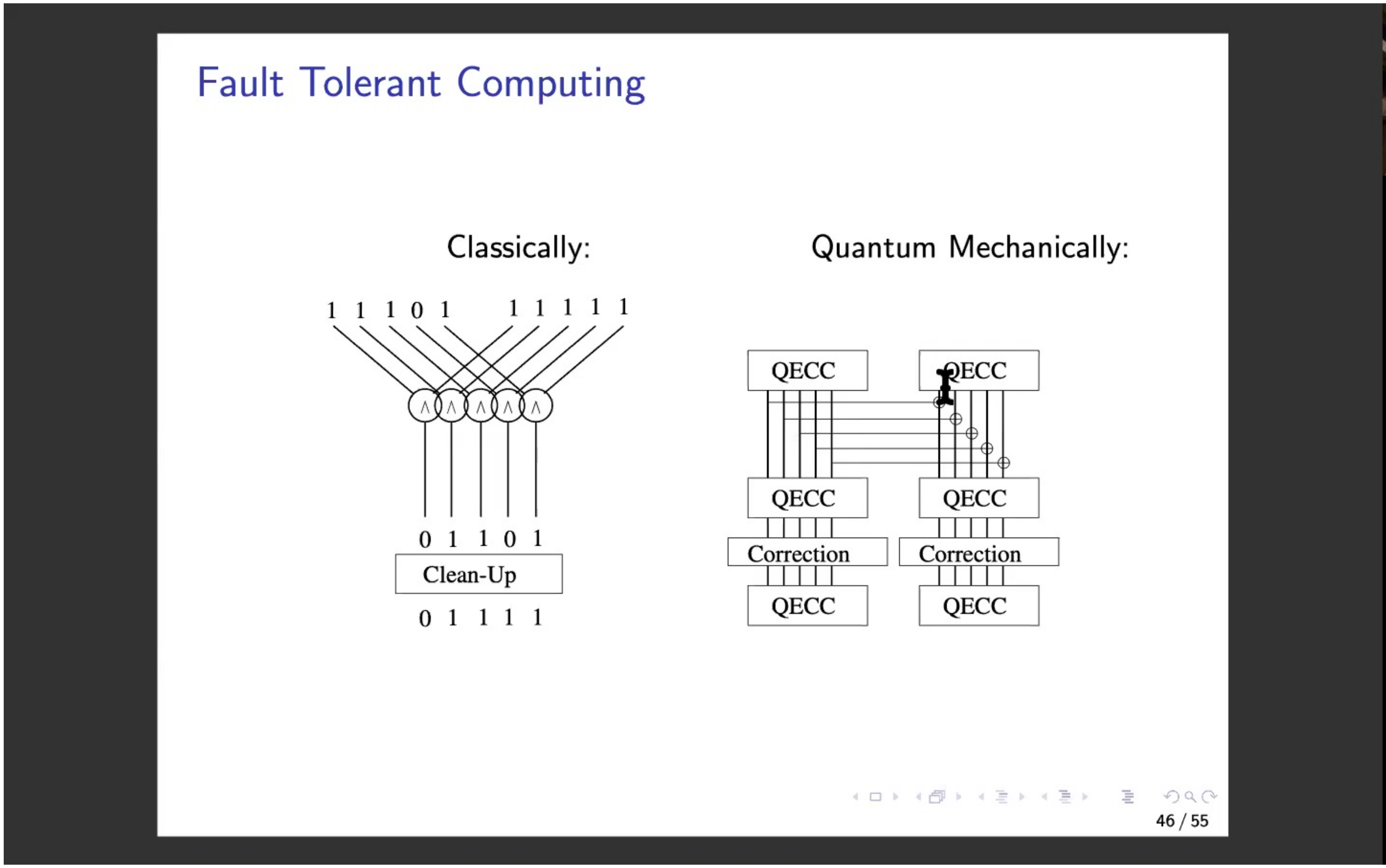


46 / 55



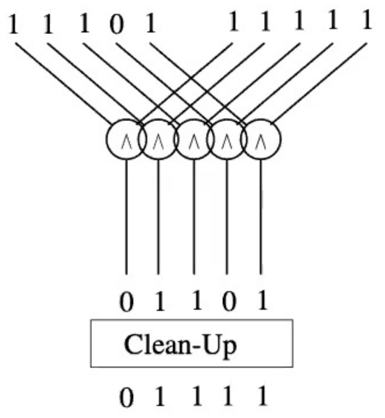
Transversal Gates

For CSS codes, certain gates can be implemented *transversally*.
For example, to do a CNOT on the encoded qubits a and b, take the CNOT of the ith qubit encoding a with the ith qubit encoding b, for $i = 1 \ldots n$.
This is automatically fault-tolerant because the error can never spread beyond the i'th qubit.

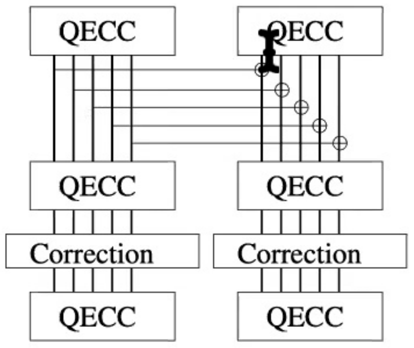


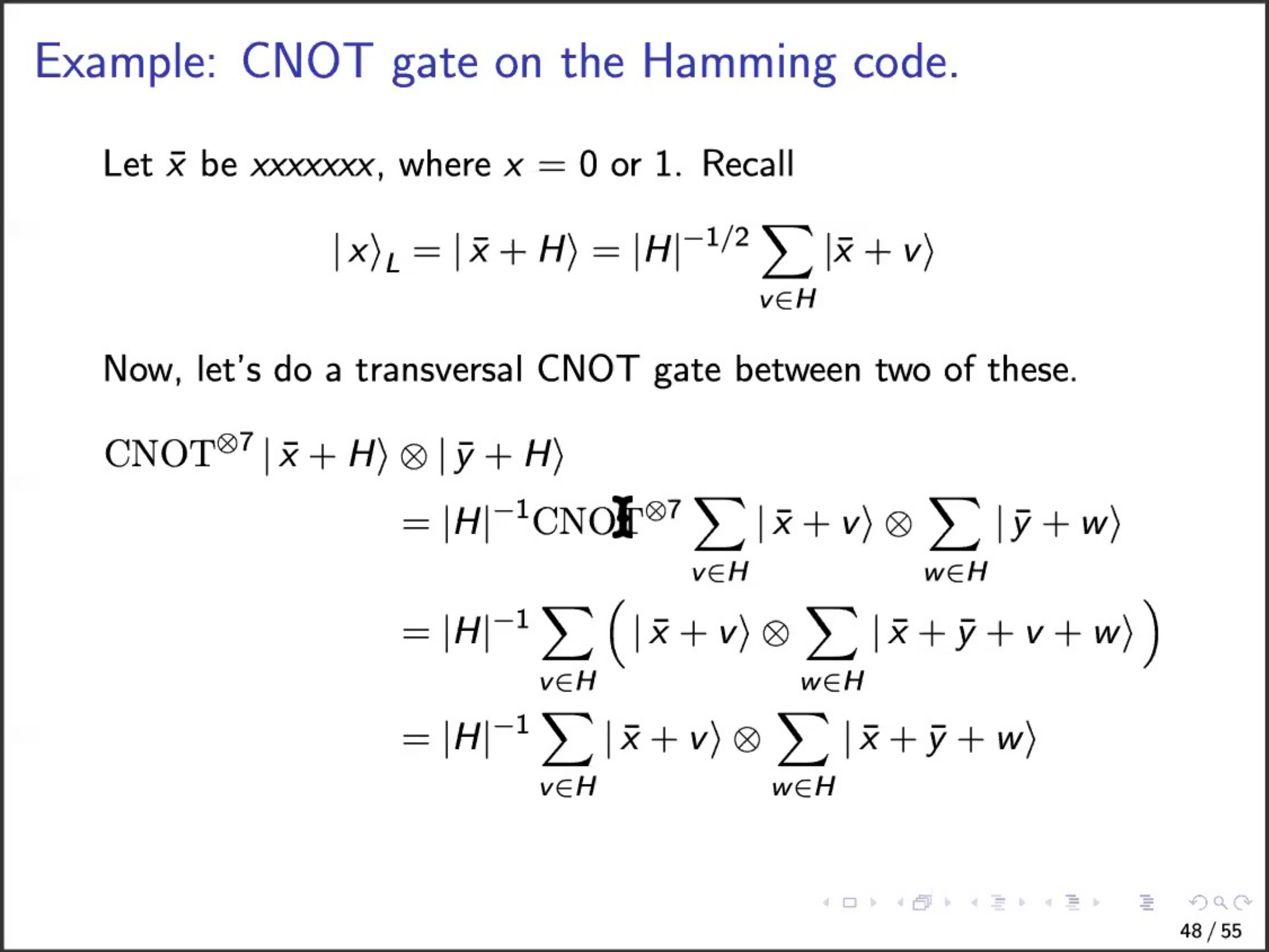
Fault Tolerant Computing

Classically:



Quantum Mechanically:





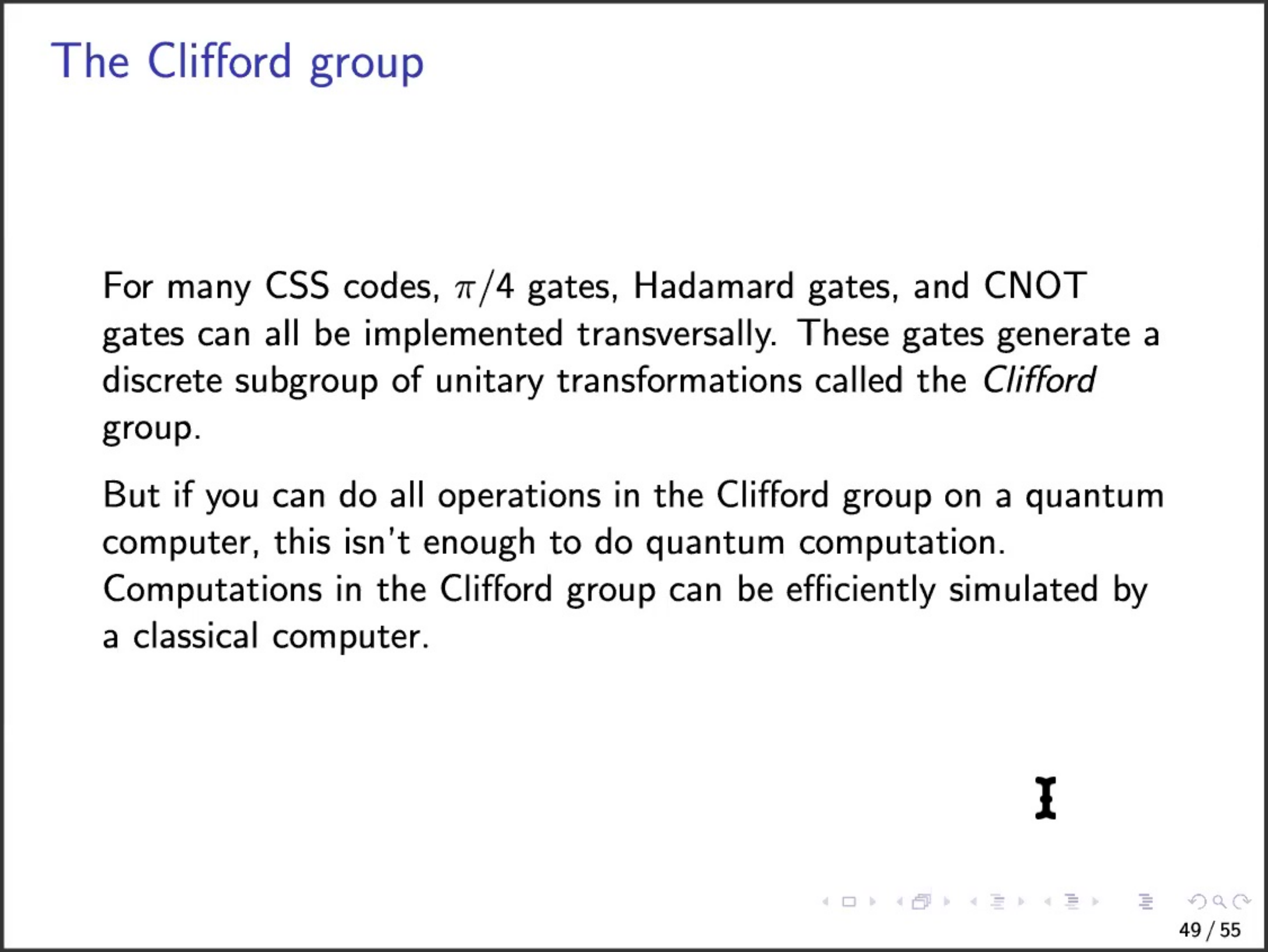
Example: CNOT gate on the Hamming code.

Let $\bar{x}$ be *xxxxxxx*, where $x = 0$ or 1. Recall

$$|x\rangle_L = |\,\bar{x} + H\rangle = |H|^{-1/2} \sum_{v \in H} |\bar{x} + v\rangle$$

Now, let's do a transversal CNOT gate between two of these.

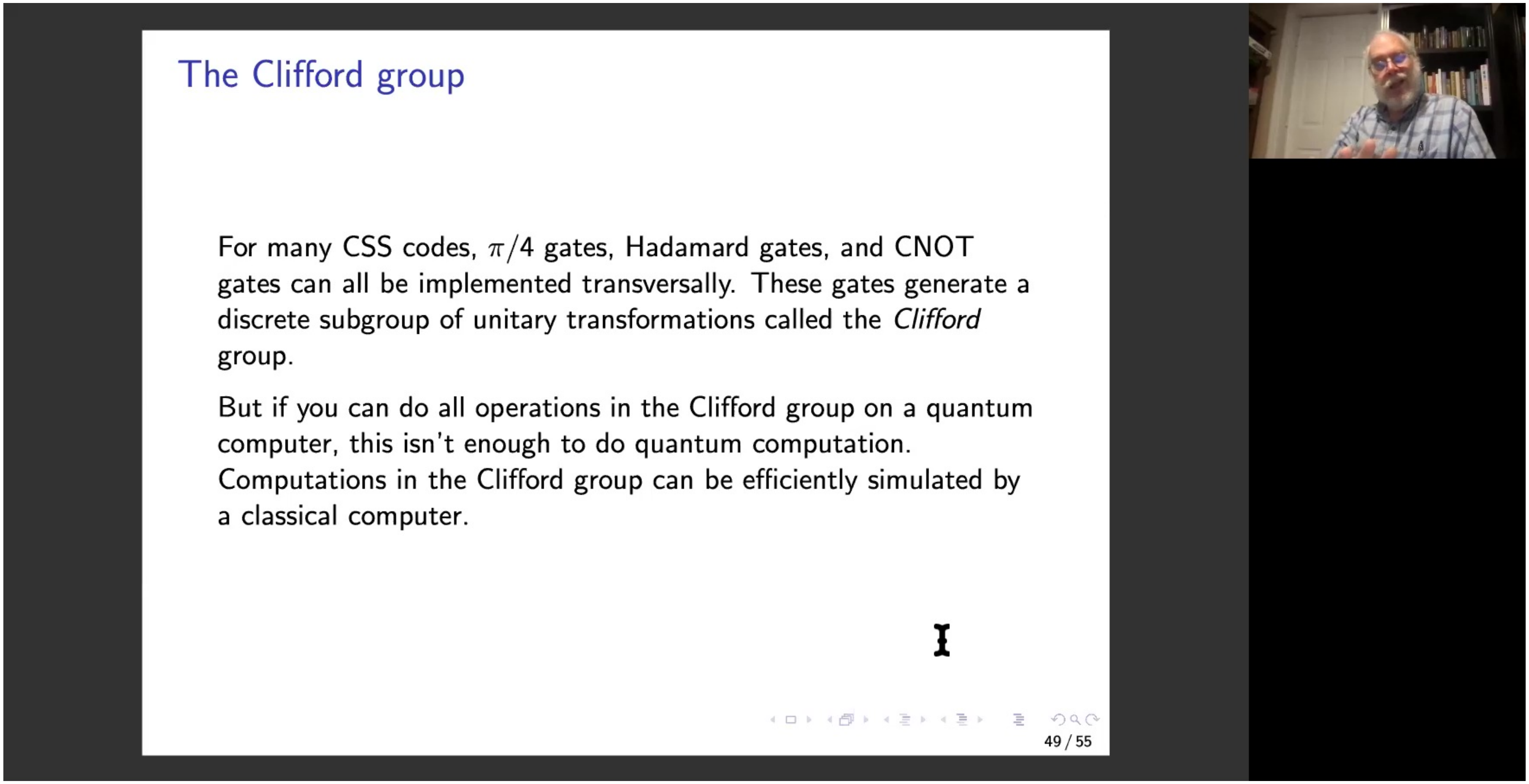
$$\begin{aligned}
&\mathrm{CNOT}^{\otimes 7}\,|\,\bar{x} + H\rangle \otimes |\,\bar{y} + H\rangle \\
&\qquad = |H|^{-1}\mathrm{CNOT}^{\otimes 7} \sum_{v \in H} |\,\bar{x} + v\rangle \otimes \sum_{w \in H} |\,\bar{y} + w\rangle \\
&\qquad = |H|^{-1} \sum_{v \in H} \Big(|\,\bar{x} + v\rangle \otimes \sum_{w \in H} |\,\bar{x} + \bar{y} + v + w\rangle \Big) \\
&\qquad = |H|^{-1} \sum_{v \in H} |\,\bar{x} + v\rangle \otimes \sum_{w \in H} |\,\bar{x} + \bar{y} + w\rangle
\end{aligned}$$



The Clifford group

For many CSS codes, $\pi/4$ gates, Hadamard gates, and CNOT gates can all be implemented transversally. These gates generate a discrete subgroup of unitary transformations called the *Clifford* group.

But if you can do all operations in the Clifford group on a quantum computer, this isn't enough to do quantum computation. Computations in the Clifford group can be efficiently simulated by a classical computer.

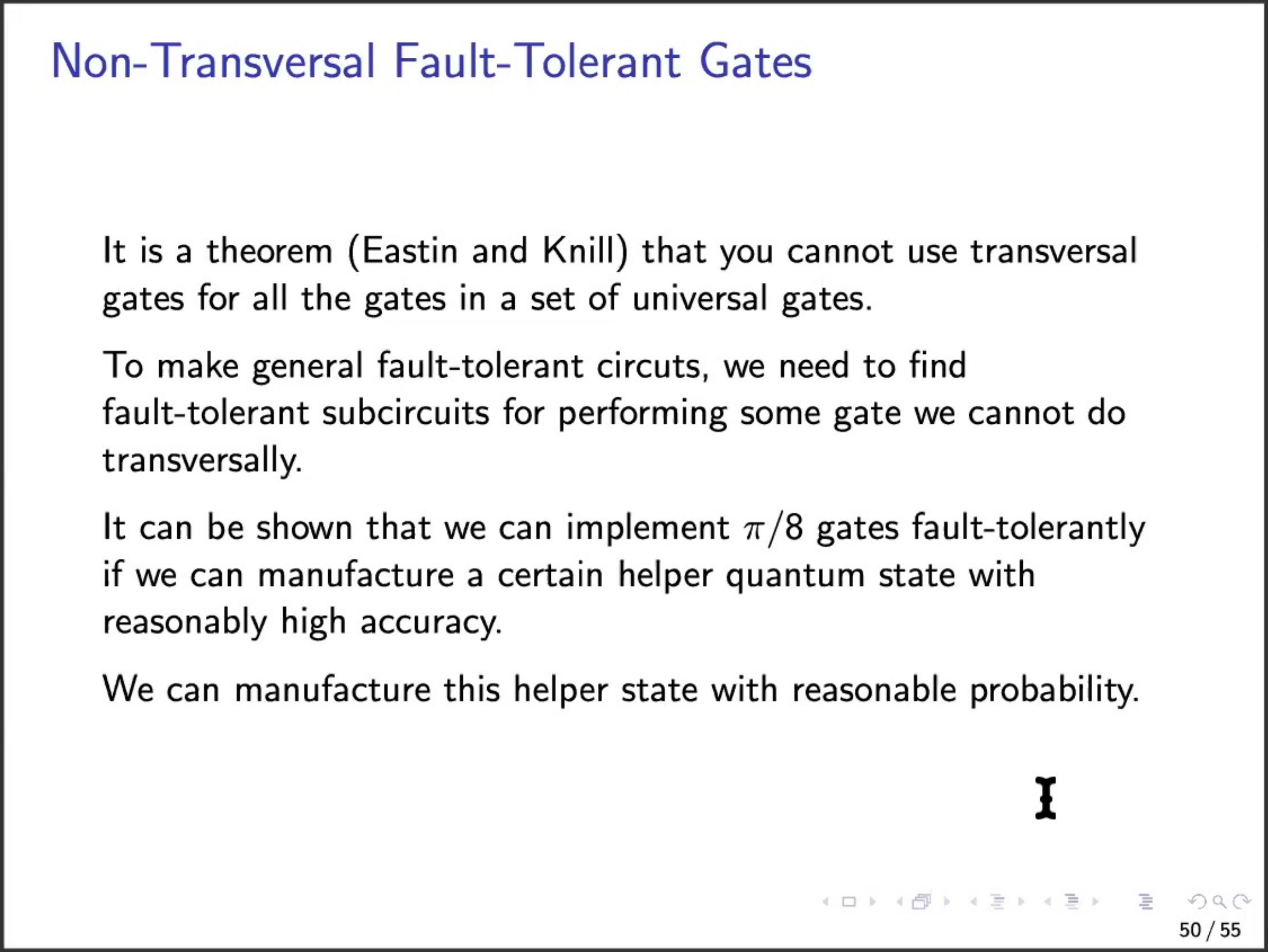


The Clifford group

For many CSS codes, $\pi/4$ gates, Hadamard gates, and CNOT gates can all be implemented transversally. These gates generate a discrete subgroup of unitary transformations called the *Clifford* group.

But if you can do all operations in the Clifford group on a quantum computer, this isn't enough to do quantum computation. Computations in the Clifford group can be efficiently simulated by a classical computer.

49 / 55



Non-Transversal Fault-Tolerant Gates

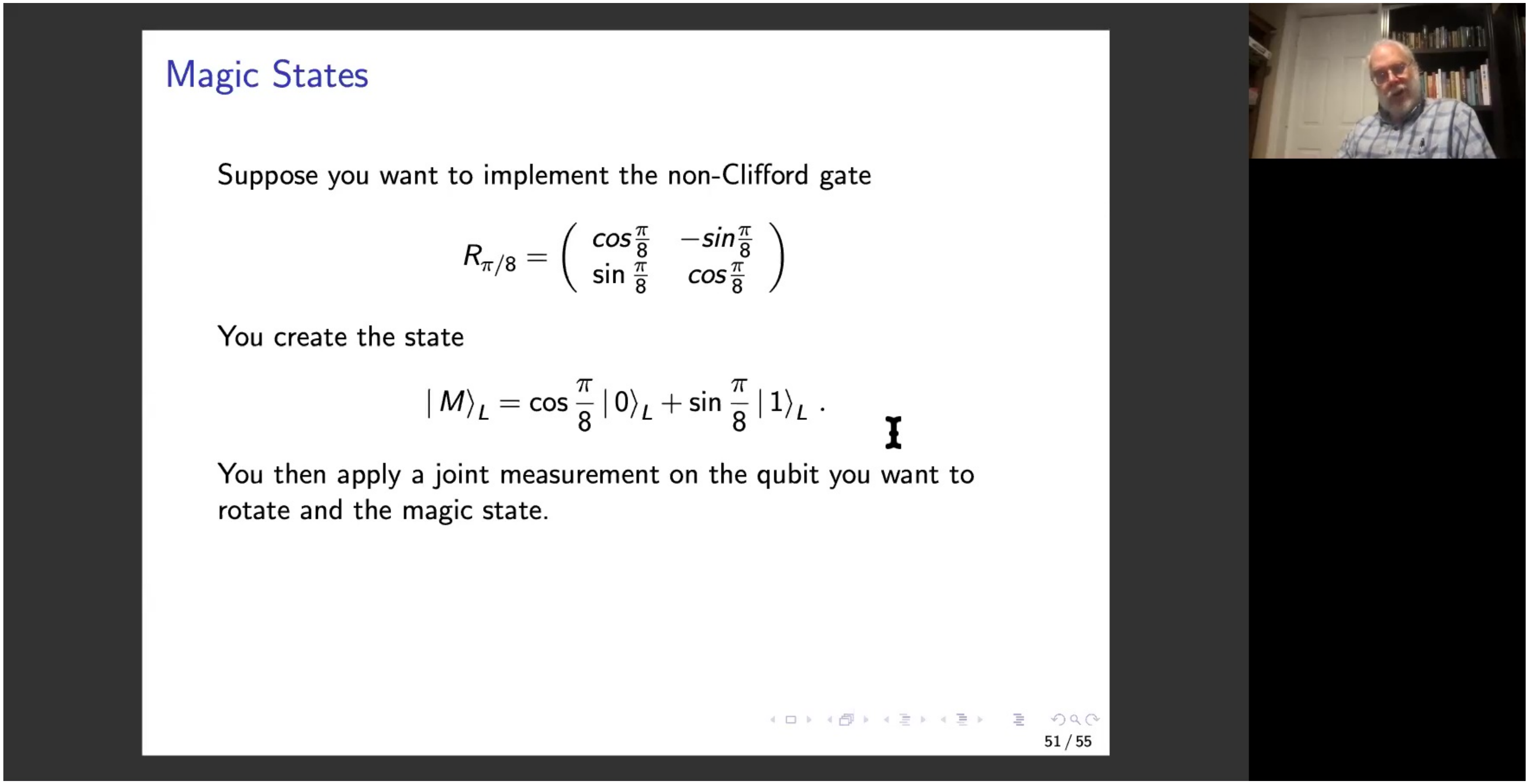
It is a theorem (Eastin and Knill) that you cannot use transversal gates for all the gates in a set of universal gates.

To make general fault-tolerant circuts, we need to find fault-tolerant subcircuits for performing some gate we cannot do transversally.

It can be shown that we can implement $\pi/8$ gates fault-tolerantly if we can manufacture a certain helper quantum state with reasonably high accuracy.

We can manufacture this helper state with reasonable probability.

50 / 55



Magic States

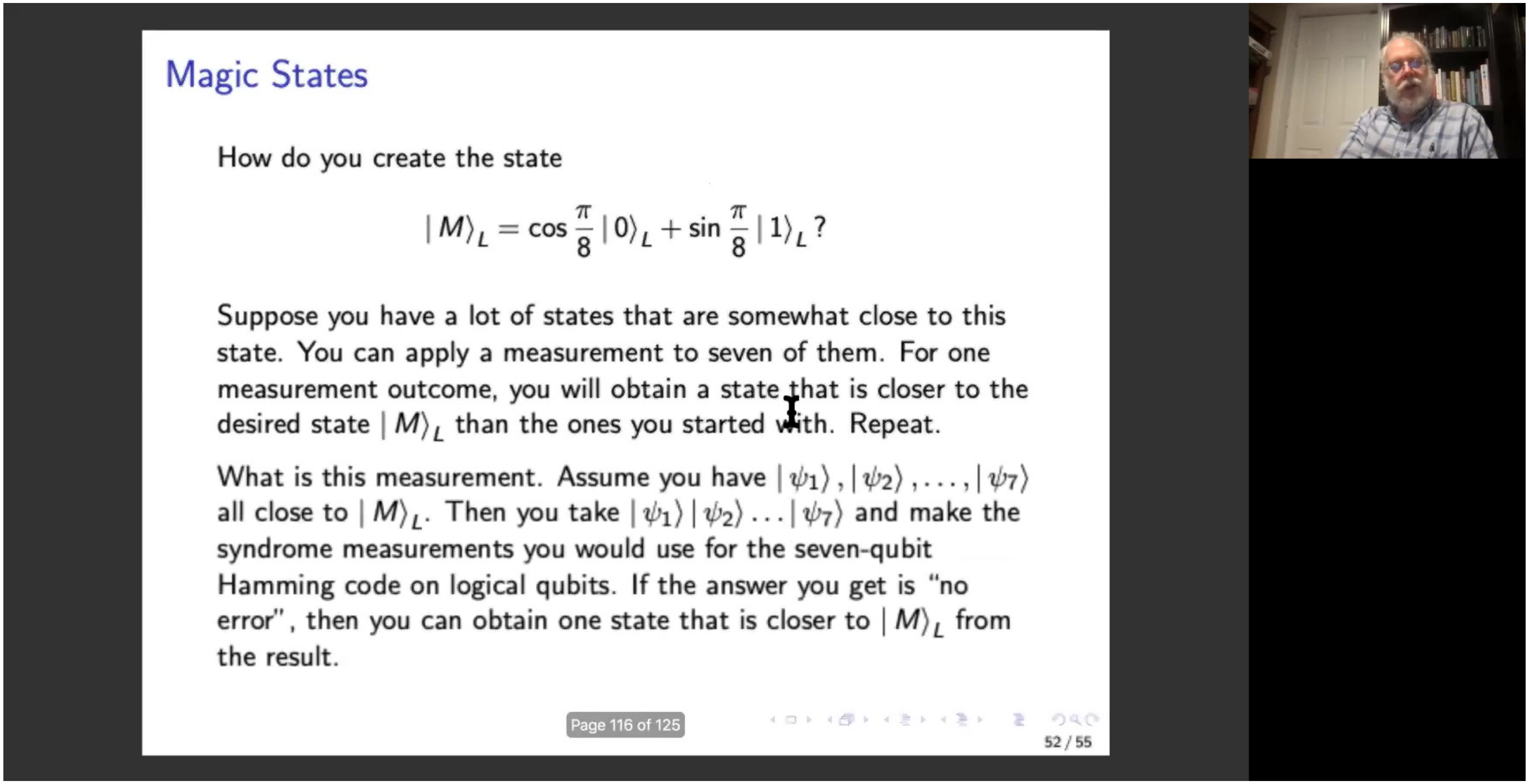
Suppose you want to implement the non-Clifford gate

$$R_{\pi/8} = \begin{pmatrix} cos\frac{\pi}{8} & -sin\frac{\pi}{8} \\ \sin\frac{\pi}{8} & cos\frac{\pi}{8} \end{pmatrix}$$

You create the state

$$|\,M\rangle_L = \cos\frac{\pi}{8}\,|\,0\rangle_L + \sin\frac{\pi}{8}\,|\,1\rangle_L\ .$$

You then apply a joint measurement on the qubit you want to rotate and the magic state.



Magic States

How do you create the state

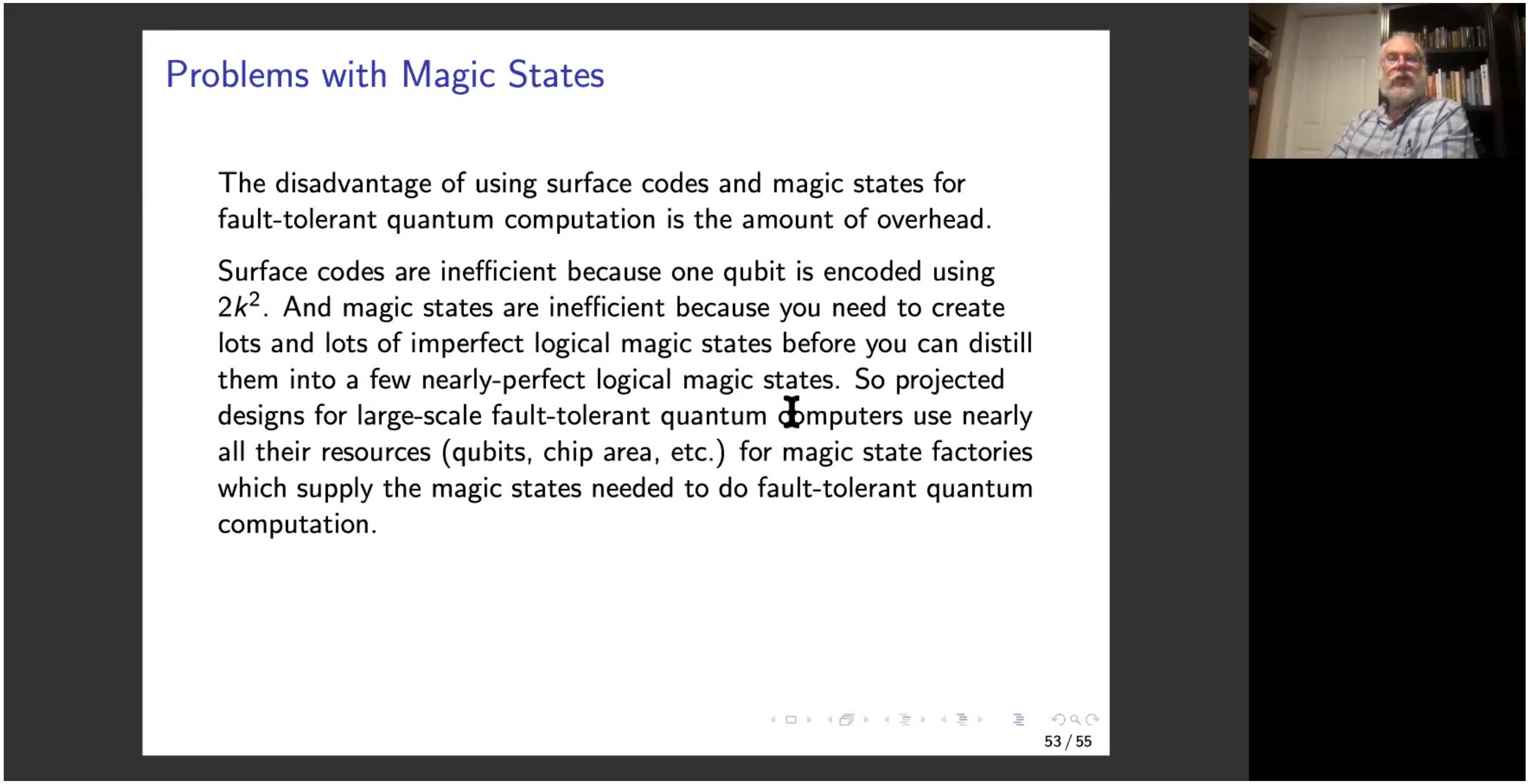
$$|M\rangle_L = \cos\frac{\pi}{8}|0\rangle_L + \sin\frac{\pi}{8}|1\rangle_L ?$$

Suppose you have a lot of states that are somewhat close to this state. You can apply a measurement to seven of them. For one measurement outcome, you will obtain a state that is closer to the desired state $|M\rangle_L$ than the ones you started with. Repeat.

What is this measurement. Assume you have $|\psi_1\rangle, |\psi_2\rangle, \ldots, |\psi_7\rangle$ all close to $|M\rangle_L$. Then you take $|\psi_1\rangle|\psi_2\rangle\ldots|\psi_7\rangle$ and make the syndrome measurements you would use for the seven-qubit Hamming code on logical qubits. If the answer you get is "no error", then you can obtain one state that is closer to $|M\rangle_L$ from the result.

Page 116 of 125

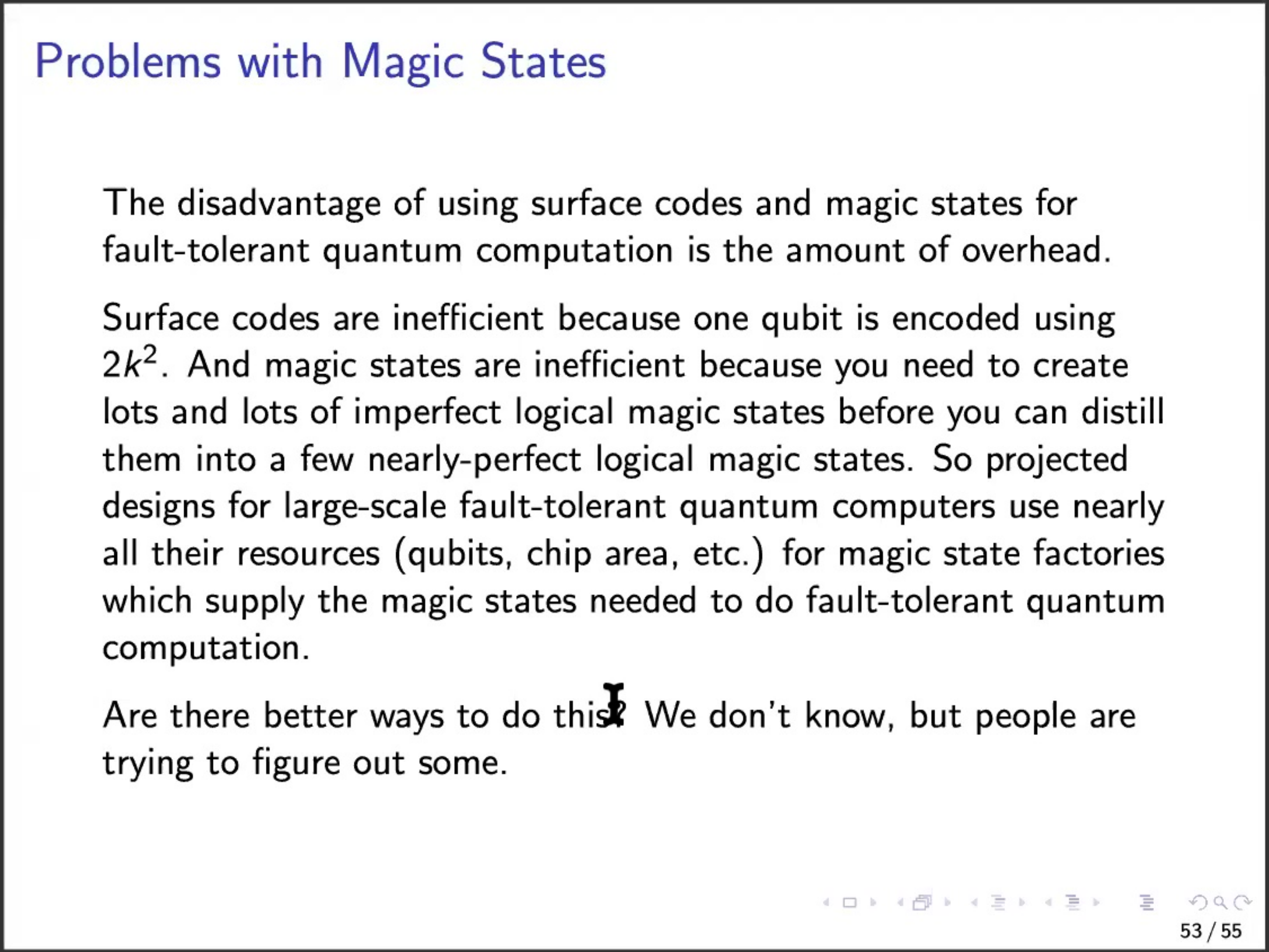
52 / 55



Problems with Magic States

The disadvantage of using surface codes and magic states for fault-tolerant quantum computation is the amount of overhead.

Surface codes are inefficient because one qubit is encoded using $2k^2$. And magic states are inefficient because you need to create lots and lots of imperfect logical magic states before you can distill them into a few nearly-perfect logical magic states. So projected designs for large-scale fault-tolerant quantum computers use nearly all their resources (qubits, chip area, etc.) for magic state factories which supply the magic states needed to do fault-tolerant quantum computation.

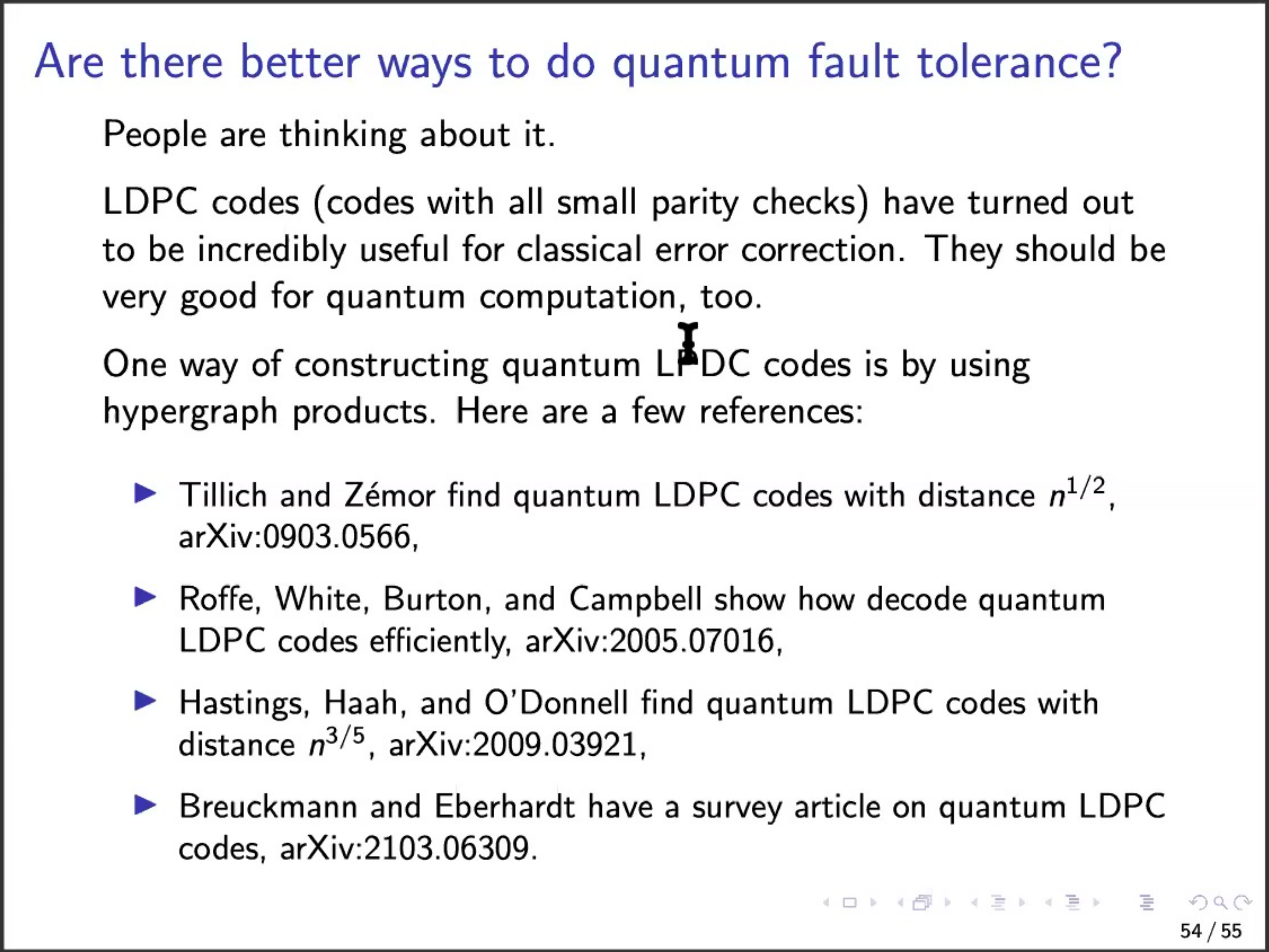


Problems with Magic States

The disadvantage of using surface codes and magic states for fault-tolerant quantum computation is the amount of overhead.

Surface codes are inefficient because one qubit is encoded using $2k^2$. And magic states are inefficient because you need to create lots and lots of imperfect logical magic states before you can distill them into a few nearly-perfect logical magic states. So projected designs for large-scale fault-tolerant quantum computers use nearly all their resources (qubits, chip area, etc.) for magic state factories which supply the magic states needed to do fault-tolerant quantum computation.

Are there better ways to do this? We don't know, but people are trying to figure out some.



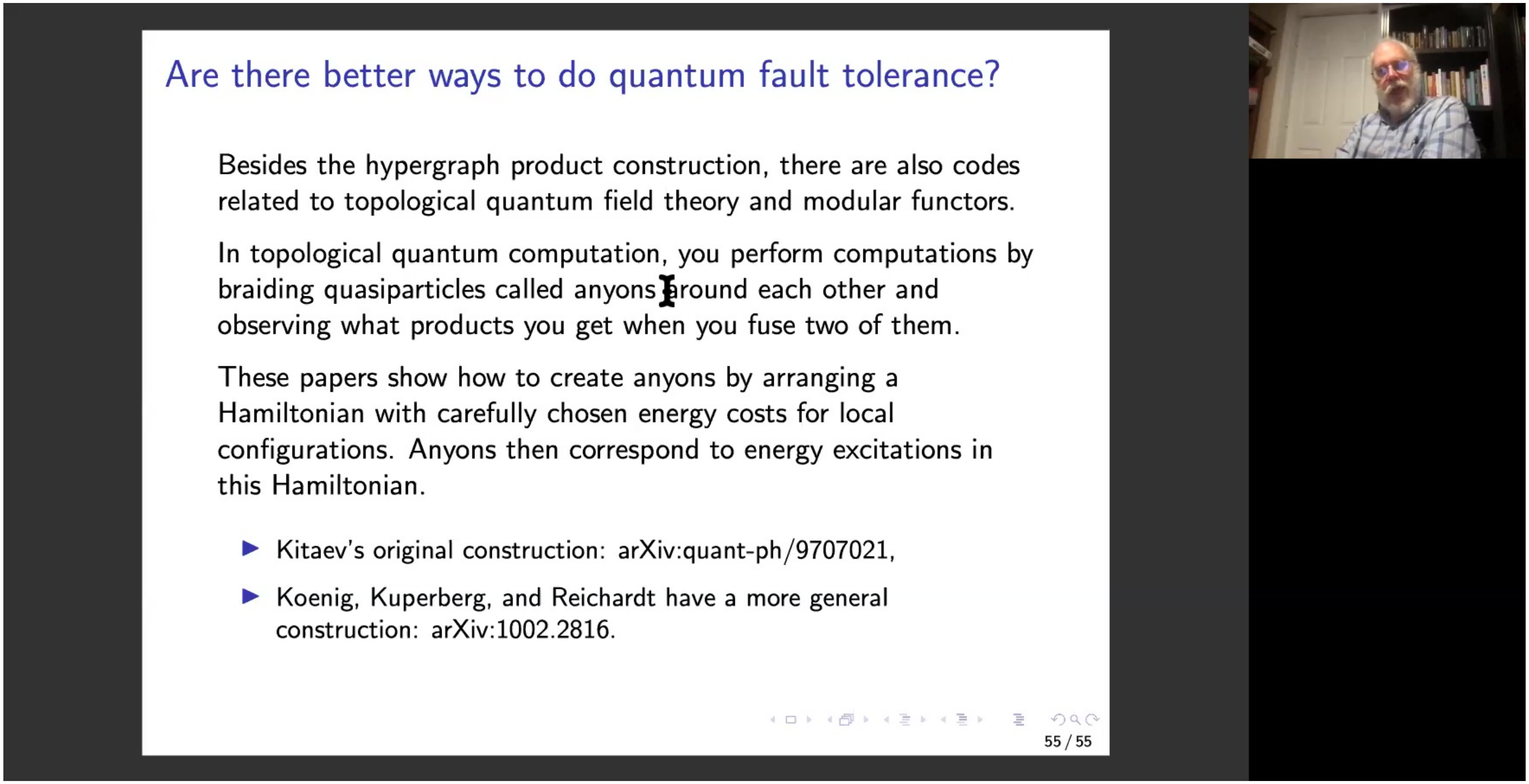
Are there better ways to do quantum fault tolerance?

People are thinking about it.

LDPC codes (codes with all small parity checks) have turned out to be incredibly useful for classical error correction. They should be very good for quantum computation, too.

One way of constructing quantum LDPC codes is by using hypergraph products. Here are a few references:

- Tillich and Zémor find quantum LDPC codes with distance $n^{1/2}$, arXiv:0903.0566,
- Roffe, White, Burton, and Campbell show how decode quantum LDPC codes efficiently, arXiv:2005.07016,
- Hastings, Haah, and O'Donnell find quantum LDPC codes with distance $n^{3/5}$, arXiv:2009.03921,
- Breuckmann and Eberhardt have a survey article on quantum LDPC codes, arXiv:2103.06309.



Are there better ways to do quantum fault tolerance?

Besides the hypergraph product construction, there are also codes related to topological quantum field theory and modular functors.

In topological quantum computation, you perform computations by braiding quasiparticles called anyons around each other and observing what products you get when you fuse two of them.

These papers show how to create anyons by arranging a Hamiltonian with carefully chosen energy costs for local configurations. Anyons then correspond to energy excitations in this Hamiltonian.

- Kitaev's original construction: arXiv:quant-ph/9707021,
- Koenig, Kuperberg, and Reichardt have a more general construction: arXiv:1002.2816.