XV International Workshop "Computer Algebra in Scientific Computing"
September 9-13, 2013, Berlin, Germany

Simulation of Quantum Error Correction with Mathematica

Vladimir P. Gerdt
Joint Institute for Nuclear Research, Dubna, Russia
E-mail: **gerdt@jinr.ru**

Alexander N. Prokopenya
Warsaw University of Life Sciences - SGGW, Poland
E-mail: alexander_prokopenya@sggw.pl

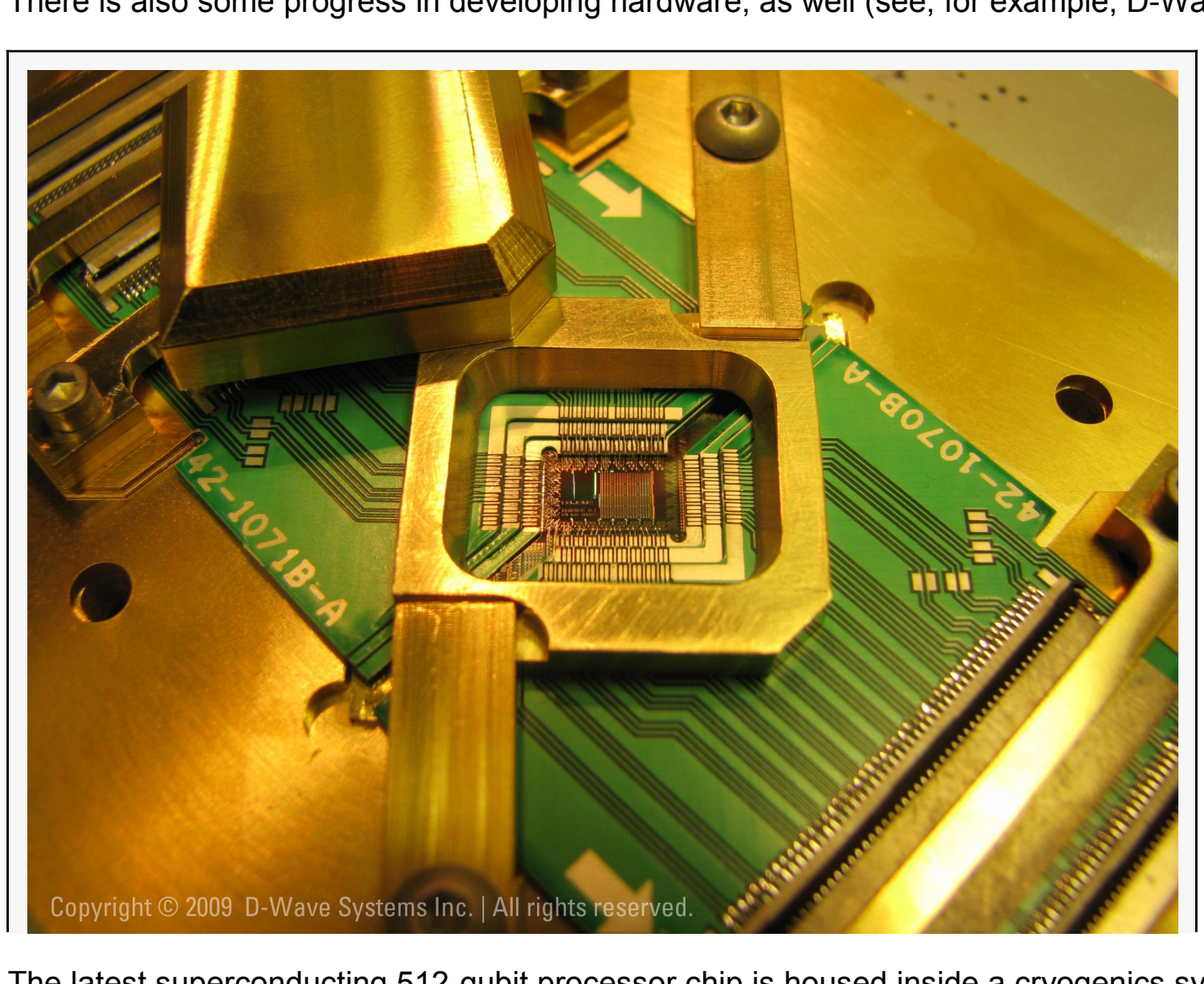
Initialization

Content

Motivation

Nowadays **quantum computation and quantum information is a rapidly developing research area of modern science and technology**. It is sufficient to note that number of publications devoted to different aspects of quantum computation grows exponentially. Main reason for this is a potential ability of a quantum computer to do certain computational task much more efficiently than it can be done by any classical computer (see: Nielsen M. and Chuang I. Quantum Computation and Quantum Information. Cambridge University Press, 2000).
Two the most famous examples of such calculations are Shor's algorithm for efficient factorization of large integers and Grover's algorithm of element search in an unsorted list.

There is also some progress in developing hardware, as well (see, for example, D-Wave's 128-qubit quantum computer (source: http://dwave.wordpress.com/).



The latest superconducting 512-qubit processor chip is housed inside a cryogenics system within a 10 square meter shielded room (from D-Wave – the Quantum Computing Company: http://www.dwavesys.com)

Two topical directions of investigations in quantum computation are
i) Development of the hardware to construct a realistic quantum computer that enables to test some known quantum algorithms;
ii) Searching for the problems which can be solved efficiently with a quantum computer and design the corresponding quantum algorithms.

As a realistic quantum computer is still not available it is expedient to use classical simulators of quantum computation to design new and to test known efficient quantum algorithms. We have developed a simulator of quantum computation, namely, the *Mathematica* package "QuantumCircuit", see
1. Gerdt, V.P., Kragler, R., Prokopenya, A.N.: A Mathematica program for constructing quantum circuits and computing their unitary matrices. Physics of Particles and Nuclei, Letters **6**, No. 7 (2009) 526 – 529.
2. Gerdt V.P., Prokopenya A.N. The circuit model of quantum computation and its simulation with Mathematica. In: "Mathematical Modeling and Computational Science", G. Adam, J. Buša, M. Hnatič (Eds.), LNCS, vol. 7125, Springer-Verlag, Berlin Heidelberg (2012) 43 – 55.
The main purpose of the present talk is
- to **present our *Mathematica* package QuantumCircuit**
- to **demonstrate how quantum error correction can be simulated** with this package
- to **show that such simulation helps to understand ideas of quantum computation better**.

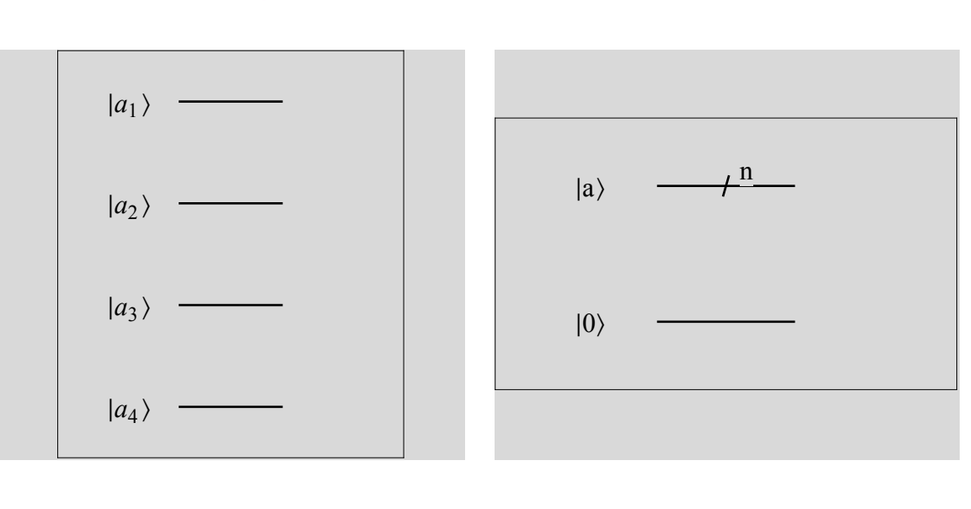
Circuit model for quantum computation

Among several equivalent models of quantum computation the **quantum circuit model is the easiest to implement**, therefore this model is widely used for quantum algorithms. Quantum circuits are also **theoretically interesting as a tool for understanding the power and limitations of quantum computation**.

Quantum computation is composed of **three basic steps**:
(i) **preparation** of the input state of the memory register,
(ii) **implementation of the desired algorithm** (or desired unitary transformation acting on the memory register), and
(iii) **measurement of the output state**.

Elementary unit of quantum information is **a quantum bit** or ***qubit*** that is a **two-level quantum system**. It is assumed that a qubit can be prepared, manipulated and measured in a controlled way. The state of a qubit is denoted as $|a\rangle$ corresponding to standard Dirac notation for quantum mechanical states. Two possible states for a qubit are usually denoted as $|0\rangle$ and $|1\rangle$, which correspond to the states 0 and 1 for a classical bit. But in contrast to classical bits, qubit as a quantum system may exist not only in one of the states $|0\rangle$ or $|1\rangle$ but also in the state $|a\rangle$ being a superposition of these states
$$|a\rangle = \alpha\,|0\rangle + \beta\,|1\rangle,$$
where α and β are complex numbers constrained by the normalization condition $|\alpha|^2 + |\beta|^2 = 1$. Thus, the state of a qubit is represented by the vector $|a\rangle$ in the two-dimensional complex vector space, where the special states $|0\rangle$ and $|1\rangle$ form an orthonormal basis and are known as ***computational basis states***.

A set of qubits forms a quantum memory register, where the input data and any intermediate results of computations are held. It is shown on diagrams as a column of states of the form $|a_j\rangle$ $(j = 1, 2, \ldots, n)$ from which quantum wires start, for example,



Note that the term "wires" is merely used to show evolution of qubits acted on by various quantum gates.

A system of n qubits has 2^n basis states. They are obtained as tensor product of basis states $|0\rangle$ and $|1\rangle$ associated with all n qubits

$$|a_1\rangle \otimes |a_2\rangle \otimes \ldots \otimes |a_n\rangle \equiv |a_1 a_2 \ldots a_n\rangle \equiv |a\rangle_n,$$

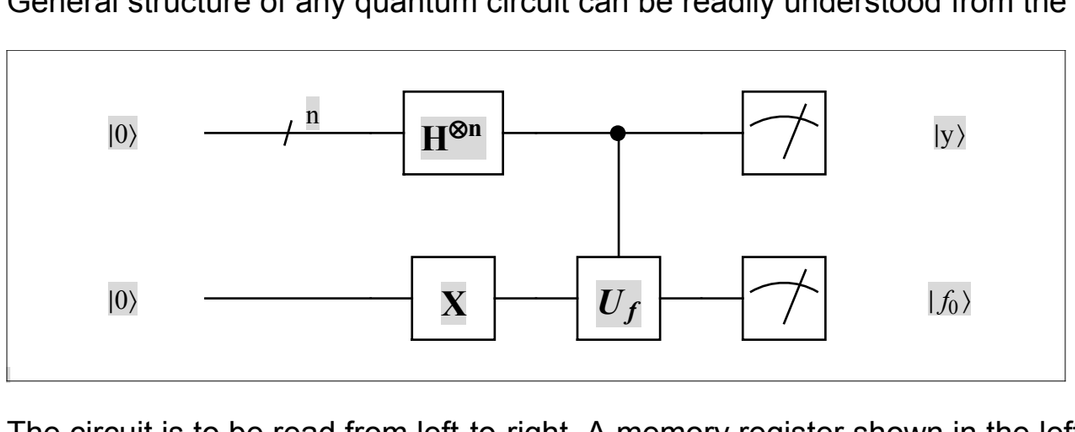
where the symbol $\otimes$ denotes a tensor product of vectors.

For $n = 3$, for example, they are given by

$$|0\rangle_3 \to \{0,0,0\} \to \begin{pmatrix}1\\0\\0\\0\\0\\0\\0\\0\end{pmatrix}, \quad |1\rangle_3 \to \{0,0,1\} \to \begin{pmatrix}0\\1\\0\\0\\0\\0\\0\\0\end{pmatrix}, \quad \ldots, \quad |7\rangle_3 \to \{1,1,1\} \to \begin{pmatrix}0\\0\\0\\0\\0\\0\\0\\1\end{pmatrix}$$

Dimension of the state space grows exponentially and to save memory we try to use sparse vectors to represent qubit states anywhere it is only possible.

General structure of any quantum circuit can be readily understood from the following quantum circuit.



The circuit is to be read from left-to-right. A memory register shown in the left-hand side of the diagram consists of a set of n qubits and one ancillary qubit initially set in the states $|0\rangle_n$ and $|0\rangle$, respectively. Then we apply a sequence of quantum gates to different qubits and measure their final states afterwards, showing the result on the right-hand side of the diagram as the column of qubits $|y\rangle_n$ and $|f_0\rangle$.

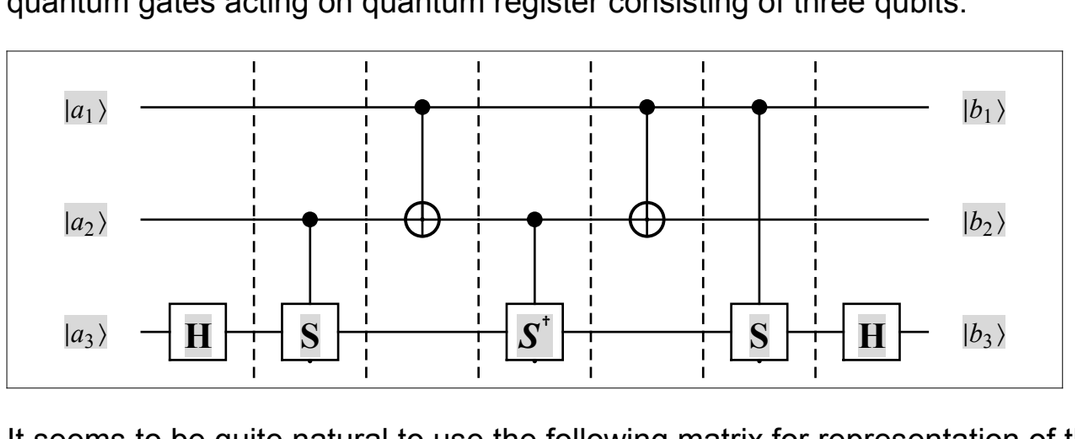
Thus, a quantum circuit can be understood as a device consisting of logical quantum gates that are arranged in the device according to steps in which the gates process qubits in time.

An algorithm of computation is determined by the number of quantum gates and their sequence.

The problem of simulating a quantum computation reduces to constructing the quantum circiut that transformes an initial state of quantum memory register into the final state that can be measured. Such transformation is done by means of the unitary operator which is decomposed into a sequence of single-qubit and multi-qubit gates. Note that our *Mathematica* package "**QuantumCircuit**" enables to calculate a unitary matrix corresponding to the quantum circuit in general case of n-qubit memory register. So we can easily calculate probabilities of its different final states.

Representation of quantum circuits

To demonstrate main idea of representation of an arbitrary quantum circuit in our package let us consider the following example. Picture below shows a sequence of quantum gates acting on quantum register consisting of three qubits.



It seems to be quite natural to use the following matrix for representation of the circuit above.

$$\begin{pmatrix} 1 & 1 & C & 1 & C & C & 1 \\ 1 & C & X & C & X & 1 & 1 \\ H & S & 1 & S^{\dagger} & 1 & S & H \end{pmatrix}$$

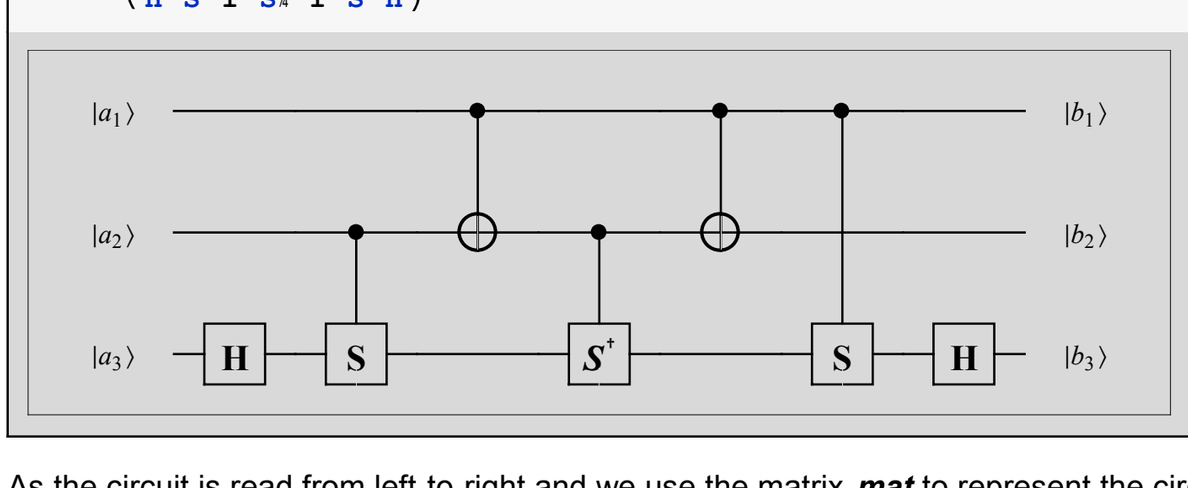
Here the **unit** means **identical transformation** of the qubit, letters "**C**" and "**X**" corresponds to the **control and target qubits** in the **controlled-NOT** gate, **H** is the Hadamard gate, **S** is the phase gate and $S^{\dagger}$ is its adjoint counterpart, and so on.
One can readily see that this matrix contains all the information about a structure of the circuit. It means that defining such a matrix we can encode information about any quntum circuit. This is an idea of our representation of quantum circuits.

Computing the circuit matrix

Remind that a system of n qubits has 2^n basis states. Hence, the unitary matrix U defined by the quantum circuit with n qubits may be represented as a $2^n \times 2^n$ matrix with respect to these basis states.

In[5]:=
```
mat = (1 1 C 1  C C 1
       1 C X C  X 1 1
       H S 1 S† 1 S H); circuit[mat, {a1, a2, a3}, {b1, b2, b3}]
```

Out[5]=



As the circuit is read from left-to-right and we use the matrix ***mat*** to represent the circuit, then the matrix U can be written as the following product

$$U = U_m . U_{m-1} \ldots U_1$$

where U_j ($j = 1, 2, \ldots, m$) is the $2^n \times 2^n$ matrix defined by the quantum gates being in the jth column of the matrix ***mat*** and m is a number of columns.

The unitary matrix U is computed by the function **matrixU[mat]**

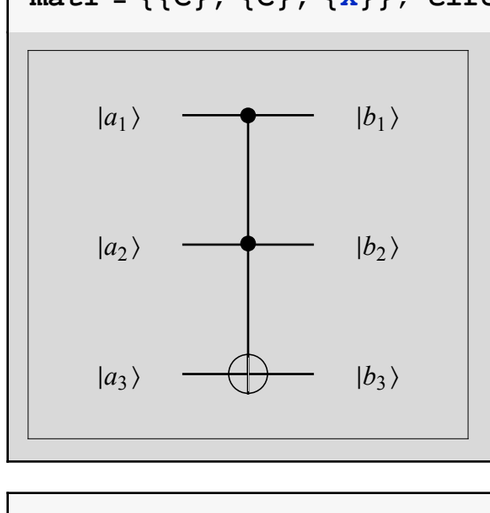
In[6]:=
```
matrixU[mat] // commonMatrixFactor
```

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

One can easily check that the matrix obtained is just the matrix corresponding to the Toffoli gate.

In[7]:=
```
mat1 = {{C}, {C}, {X}}; circuit[mat1, {a1, a2, a3}, {b1, b2, b3}]
```

Out[7]=



In[8]:=
```
{matrixU[mat] // MatrixForm , matrixU[mat1] // MatrixForm }
```

Out[8]=

$$\left\{ \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&1&0 \end{pmatrix}, \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&1&0 \end{pmatrix} \right\}$$

Quantum Error Correction

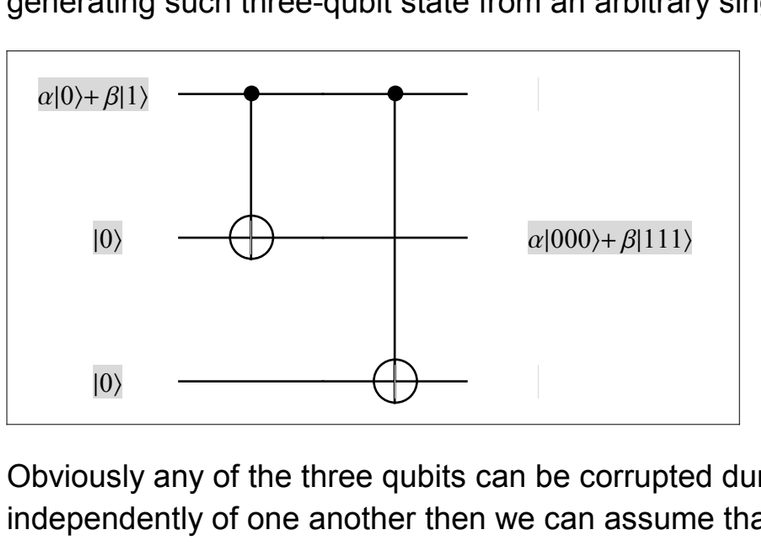
Let us consider an arbitrary qubit state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. Obviuosly, one cannot extirely isolate the qubit from an environment and due to interaction with environment its state changes without control, for example, during transmission of the state. Denoting initial state of environment as $|e\rangle$, one can represent general interaction between the qubit and its environment as the following transformation

$$|e\rangle|\psi\rangle \to (|e_I\rangle I + |e_X\rangle X + |e_Y\rangle Y + |e_Z\rangle Z)\,|\psi\rangle$$

where $|e_I\rangle, |e_X\rangle, |e_Y\rangle, |e_Z\rangle$ are possible final states of the environment, I is the identity operator, and X, Y, Z are the logical Pauli operators. The problem is to construct a quantum circuit that can identify an error using any of the four states $|\psi\rangle$, $X|\psi\rangle$, $Y|\psi\rangle$, $Z|\psi\rangle$ as an input and restore the initial qubit state $|\psi\rangle$.

To clarify basic ideas of constructing quantum circuit correcting error let us first assume for simplicity that the qubit state
$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$
is acted on by the operator Pauli-X, simulating a bit-flip error. Then the qubit state becomes
$X|\psi\rangle = \alpha\,|1\rangle + \beta\,|0\rangle$.
Both states belong to the same 2-dimensional Hilbert space and one cannot separate them. So the problem arises
1) how can the error be diagnosed?
2) how the error can be corrected if the original state is not known?

The first problem can be solved similarly to the classical case. Let us add two qubits to the original one and prepare the state $|\Psi\rangle = \alpha|000\rangle + \beta|111\rangle$. Quantum circuit generating such three-qubit state from an arbitrary single-qubit state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ contains two CNOT-gates and two ancillary qubits each initially in the state $|0\rangle$.



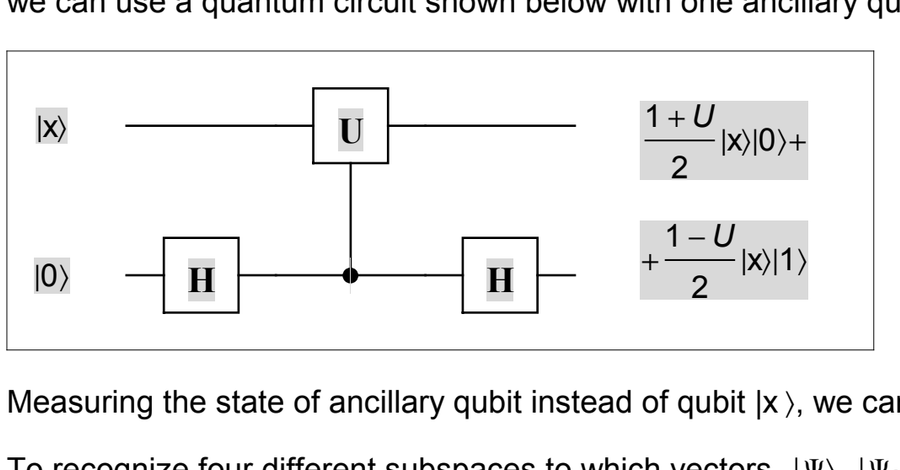
Obviously any of the three qubits can be corrupted during transmission but if probability of corruption of one qubit is sufficiently small and errors occur in different qubits independently of one another then we can assume that probability of simultaneous errors in two or more qubits is negligible and such cases may be excluded from consideration. Thus, we assume that an error occurs only in one of the three qubits. Therefore, the $2^3 = 8$-dimensional space of states of three qubits is divided into four subspaces

$|\Psi\rangle = \alpha|000\rangle + \beta|111\rangle$ – original uncorrupted state

$|\Psi_0\rangle = \alpha|001\rangle + \beta|110\rangle$; $|\Psi_1\rangle = \alpha|010\rangle + \beta|101\rangle$; $|\Psi_2\rangle = \alpha|100\rangle + \beta|011\rangle$ – corrupted states

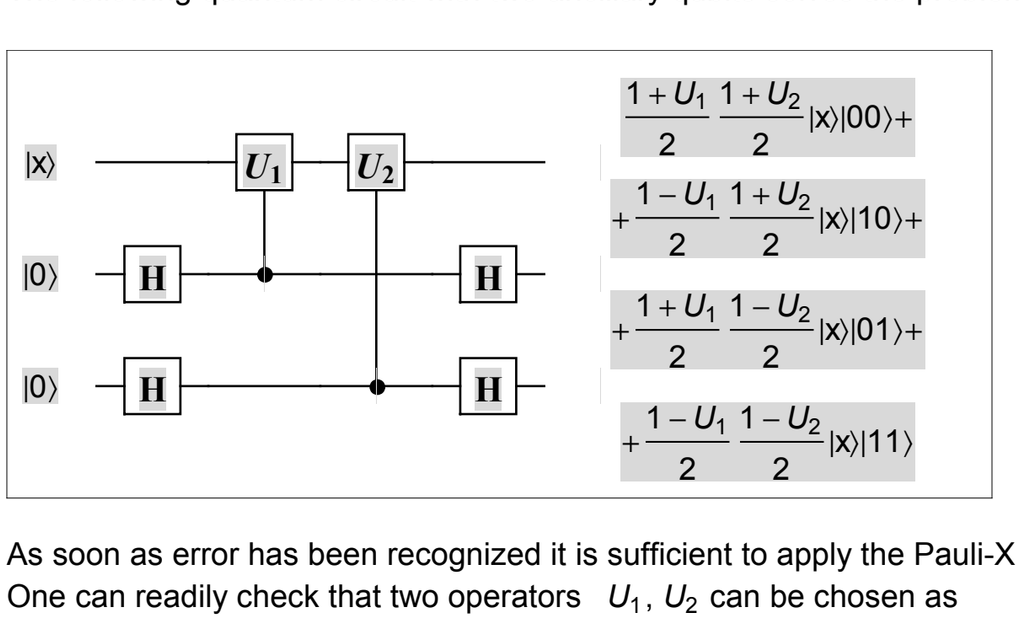
Note that four subspaces are orthogonal to each other and the problem of diagnosing the error reduces to determination to which subspace the final state belongs. But we cannot measure the final state because measurement destroys it.

One can solve the problem of measurement by means of adding ancillary qubits which are entangled with input state $|\Psi\rangle$.
Let us assume that some state $|x\rangle$ is an eigenstate of some Hermitian operator U satisfying the condition $U^2 = I$. Such operator can have the eigenvalues only – 1. Then we can use a quantum circuit shown below with one ancillary qubit set initially in the state $|0\rangle$.



Measuring the state of ancillary qubit instead of qubit $|x\rangle$, we can recognize the state $|x\rangle$ without its measuring.

To recognize four different subspaces to which vectors $|\Psi\rangle, |\Psi_0\rangle, |\Psi_1\rangle, |\Psi_2\rangle$ belong we need to have two commuting Hermitian operators U_1, U_2 satisfying the condition $U_j^2 = I$, j=1,2. Besides, let operator U_1 commute with operator X_0 and anti-commute with operators X_1, X_2 while operator U_2 commute with X_2 and anti-commute with X_0, X_1.
Then vectors $|\Psi\rangle, |\Psi_0\rangle, |\Psi_1\rangle, |\Psi_2\rangle$ are eigenvectors of both operators U_1 and U_2 corresponding to different sets of eigenvalues, namely, (1, 1), (1, -1), (-1, -1), (-1, 1).
The following quantum circuit with two ancillary qubits solves the problem of diagosing an error.

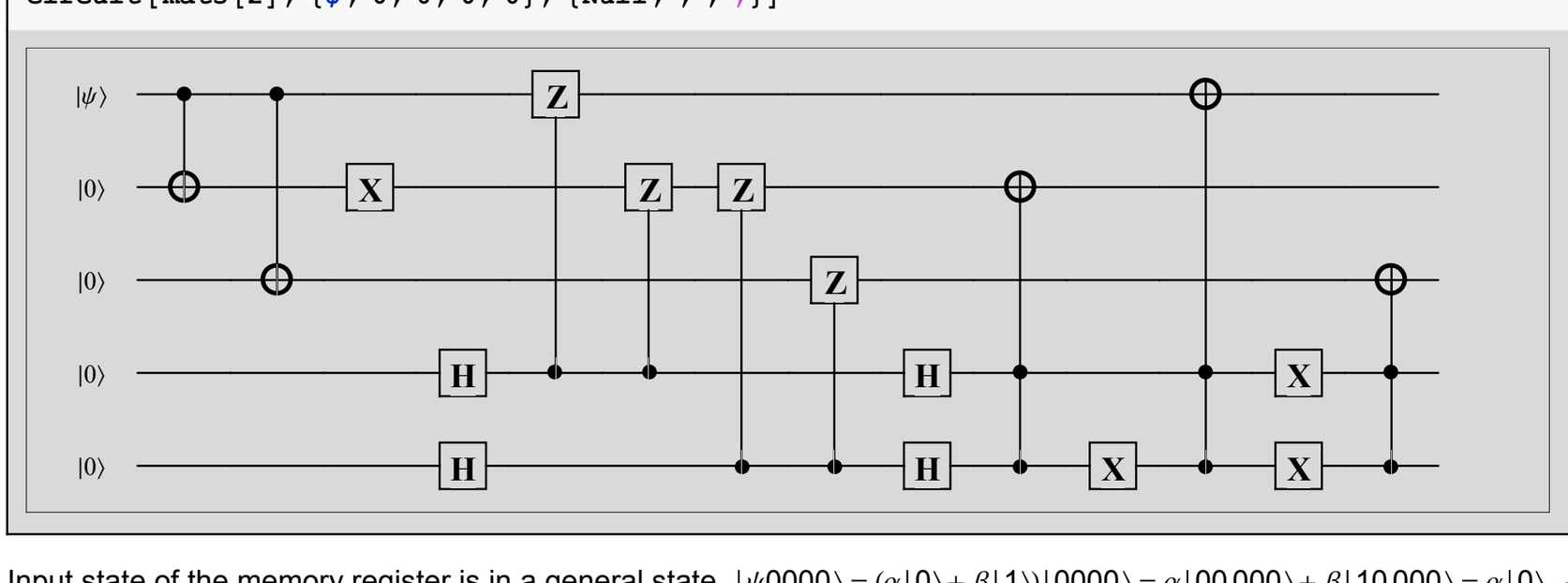


As soon as error has been recognized it is sufficient to apply the Pauli-X operator th the corresponding qubit.
One can readily check that two operators U_1, U_2 can be chosen as
$U_1 = Z_2 Z_1$, $U_2 = Z_1 Z_0$.

Then quantum circuit for error correction can be constructed in the form

In[9]:=
```
mat5[k_] := {{C, C, If[k == 1, X, 1], 1, Z, 1, 1, 1, 1, 1, 1, X, 1, 1},
         {X, 1, If[k == 2, X, 1], 1, 1, Z, Z, 1, 1, X, 1, 1, 1, 1},
         {1, X, If[k == 3, X, 1], 1, 1, 1, 1, Z, 1, 1, 1, 1, 1, X},
         {1, 1, 1, H, C, C, 1, 1, H, C, 1, C, X, C},
         {1, 1, 1, H, 1, 1, C, C, H, C, X, C, X, C}};
circuit[mat5[2], {ψ, 0, 0, 0, 0}, {Null, , , ,}]
```

Out[10]=



Input state of the memory register is in a general state $|\psi 0000\rangle = (\alpha|0\rangle + \beta|1\rangle)|0000\rangle = \alpha|00\,000\rangle + \beta|10\,000\rangle = \alpha|0\rangle_5 + \beta|16\rangle_5$

In[11]:=
```
vecInp = SparseArray[{1 → α, 17 → β}, {32}]; vecInp // Normal
```

Out[11]=
```
{α, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, β, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

Final state is given by

In[12]:=
```
vecFin1 = matrixU[mat5[2]].vecInp // Normal
```

Out[12]=
```
{0, α, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, β, 0, 0}
```

Obviously, the final state is a superposition of the following two basis vectors

In[13]:=
```
IntegerDigits[Position[vecFin1, α]〚1, 1〛 - 1, 2, 5]
```

Out[13]=
```
{0, 0, 0, 0, 1}
```

In[14]:=
```
IntegerDigits[Position[vecFin1, β]〚1, 1〛 - 1, 2, 5]
```

Out[14]=
```
{1, 1, 1, 0, 1}
```

Thus, it is $(\alpha|000\rangle + \beta|111\rangle)|01\rangle$, i.e. initial state $|\Psi\rangle = \alpha|000\rangle + \beta|111\rangle$ has been restored. Note that ancillary qubits state has been changed but it does not matter because in any case after checking each input state ancillary qubits must be set into initial state $|00\rangle$.

Five-qubit error correcting code

In general case an arbitrary two-dimensional vector $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ can be acted on by any of the four operators I, X, Y,Z and final state can be represented as

$$|e\rangle|\psi\rangle \to (|e_I\rangle I + |e_X\rangle X + |e_Y\rangle Y + |e_Z\rangle Z)\,|\psi\rangle,$$

where $|e\rangle$ is an initial state of environment, and $|e_I\rangle, |e_X\rangle, |e_Y\rangle, |e_Z\rangle$ are possible final states of the environment.
In case of n qubits used to encode the state $|\psi\rangle$ each qubit can be acted on by any of the three Pauli operators or the codeword remains uncorrupted. Therefore, to distinguish all possible errors we need $(3n+1)$ two-dimensional subspaces in the Hilbert space of dimension 2^n. The condition $2^n \geq 2(3n+1)$ gives minimum number of qubits needed to encode the input state. Besides, we need four ancillary qubits to recognize $3 \times 5 + 1 = 16$ different errors.

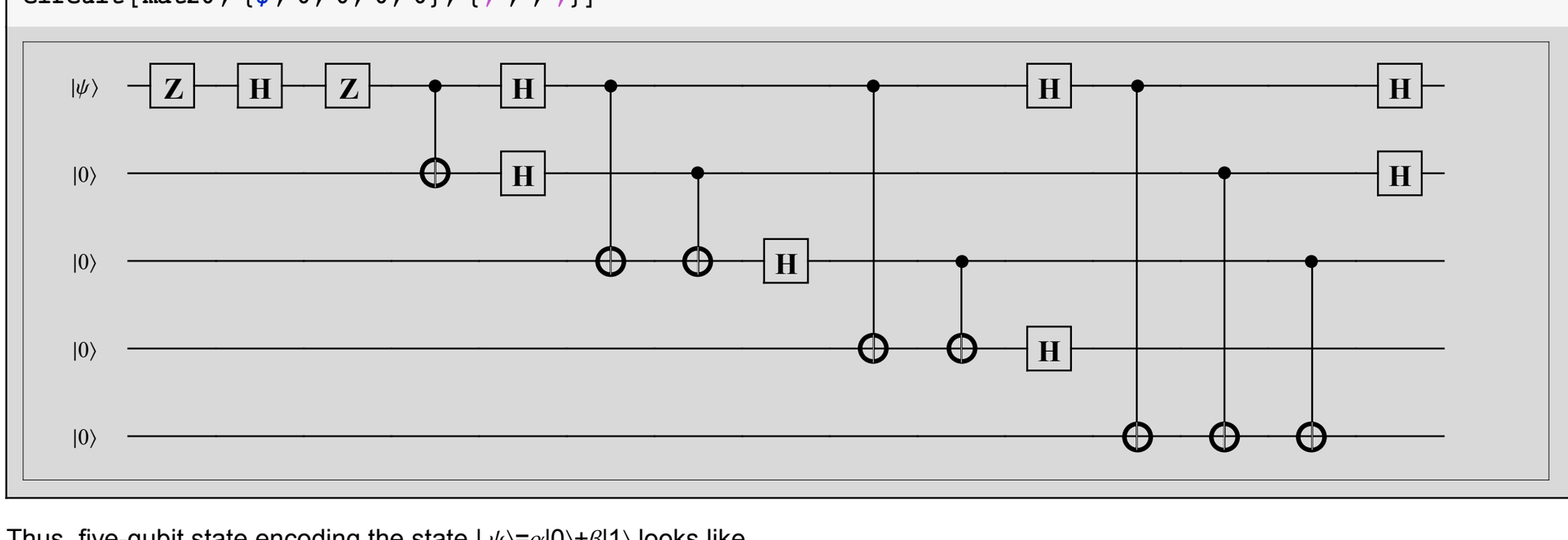
Similarly to the previous case, one can choose four operators U_j used to disgnose an error in the form

$$U_1 = Z_4 X_3 X_2 Z_1,\; U_2 = X_4 X_3 Z_2 Z_0,\; U_3 = X_4 Z_3 Z_1 X_0,\; U_4 = Z_4 Z_2 X_1 X_0.$$

Note that vector encoding the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ must be an eigenvector of each operator U_j, j=1,2,3,4, corresponding to the same eigenvalue +1. It turns out that this vector is a superposition of all 32 basis states. The following quantum circuit can be used to construct such five-qubit codeword.

In[15]:=
```
mat20 = {{Z, H, Z, C, H, C, 1, 1, C, 1, H, C, 1, 1, H},
       {1, 1, 1, X, H, 1, C, 1, 1, 1, 1, 1, C, 1, H},
       {1, 1, 1, 1, 1, X, X, H, 1, C, 1, 1, 1, C, 1},
       {1, 1, 1, 1, 1, 1, 1, 1, X, X, H, 1, 1, 1, 1},
       {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, X, X, X, 1}};
circuit[mat20, {ψ, 0, 0, 0, 0}, {, , , ,}]
```

Out[16]=



Thus, five-qubit state encoding the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ looks like

In[17]:=
```
inputVector[n_] := SparseArray[{1 → α, 2^(n-1) + 1 → β}, {2^n}]
mat20U = matrixU[mat20];
vecCode = mat20U.inputVector[5];
vecCodeN = vecCode // Normal ;
posα = Position[vecCodeN, α];
posβ = Position[vecCodeN, β];
vecIn = Apply[Plus, Table[{vecCodeN〚posα[[k, 1]]〛 StringJoin["|", ToString[posα[[k, 1]] - 1], "⟩5"],
            vecCodeN〚posβ[[k, 1]]〛 StringJoin["|", ToString[posβ[[k, 1]] - 1], "⟩5"]},
          {k, Length[posα]}] // Flatten ] // Collect[#, {α, β}, Simplify] &
```

Out[23]=
```
1/4 (|0⟩5 - |10⟩5 + |12⟩5 - |15⟩5 + |17⟩5 - |18⟩5 - |20⟩5 - |23⟩5 + |24⟩5 - |27⟩5 - |29⟩5 - |30⟩5 + |3⟩5 - |5⟩5 + |6⟩5 - |9⟩5) α +
 1/4 (- |11⟩5 - |13⟩5 + |14⟩5 - |16⟩5 + |19⟩5 - |1⟩5 - |21⟩5 - |22⟩5 + |25⟩5 - |26⟩5 + |28⟩5 - |2⟩5 + |31⟩5 - |4⟩5 + |7⟩5 - |8⟩5) β
```

Assume that a general error arises in the fourth qubit.

In[24]:=
```
matError0 = {{1}, {1}, {1}, {1}, {1}};
matError1 = {{1}, {1}, {1}, {X}, {1}};
matError2 = {{1}, {1}, {1}, {Y}, {1}};
matError3 = {{1}, {1}, {1}, {Z}, {1}};
matErrorU = a matrixU[matError0] + b matrixU[matError1] + c matrixU[matError2] + d matrixU[matError3];
vecError = matErrorU.vecCode;
vecErr = vecError // Normal;
```

Here **a,b,c,d** are arbitrary complex numbers constrained only by the condition of normalization of the vector ***vecErr***.
One can readily see that corrupted five-qubit codeword is much more complicated in comparison with uncorrupted state.

In[31]:=
```
posα = Position[vecErr, α];
posβ = Position[vecErr, β];
Apply[Plus, Table[{vecErr〚posα[[k, 1]]〛 StringJoin["|", ToString[posα[[k, 1]] - 1], "⟩5"],
            vecErr〚posβ[[k, 1]]〛 StringJoin["|", ToString[posβ[[k, 1]] - 1], "⟩5"]},
              {k, Length[posα]}] // Flatten ] // Collect[#, {α, β}, Simplify] &
```

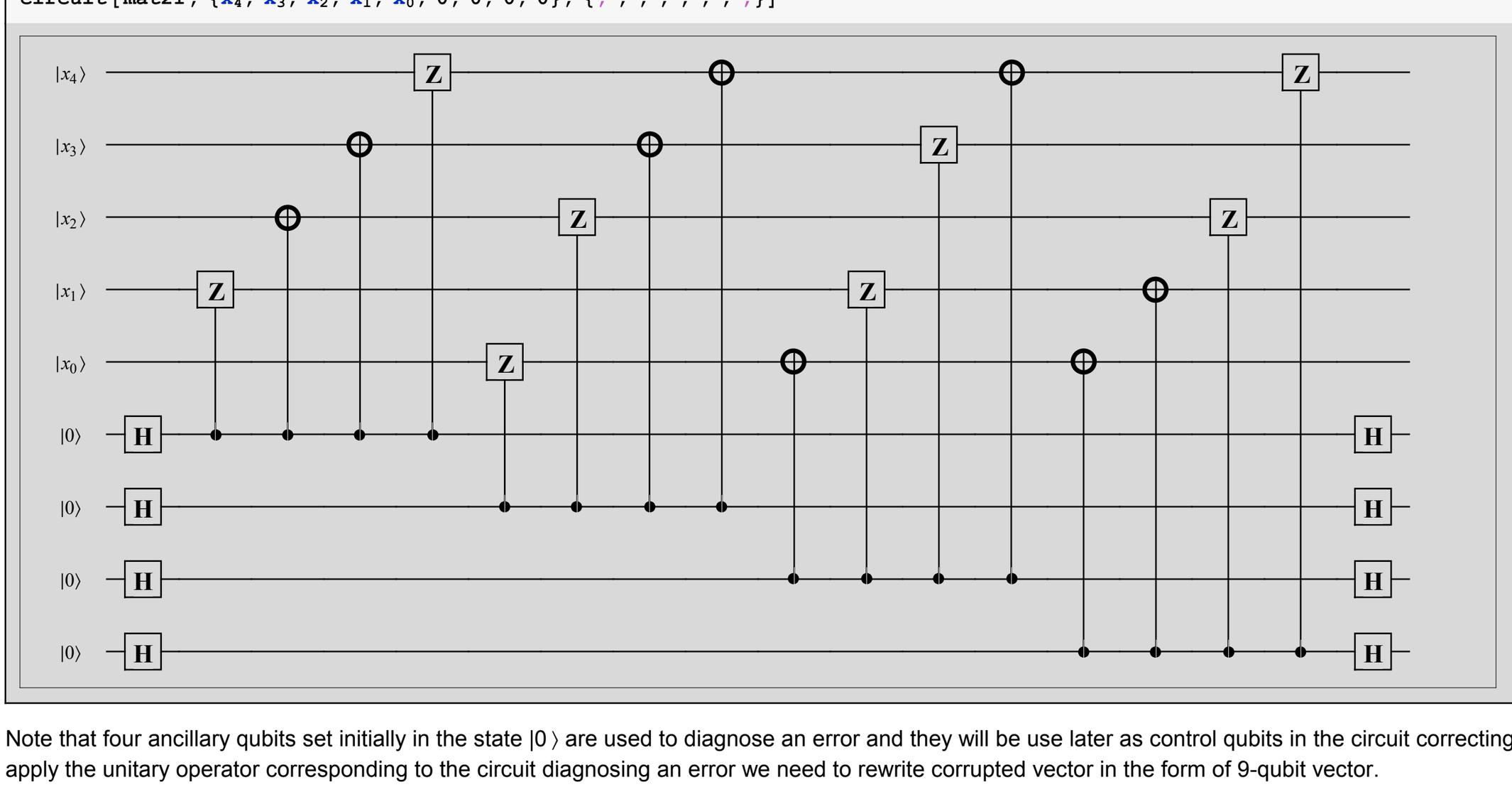
Out[33]=
```
1/2 (|12⟩5 a - |15⟩5 a + |17⟩5 a - |18⟩5 a - |20⟩5 a - |23⟩5 a + |24⟩5 a - |27⟩5 a - |29⟩5 a - |30⟩5 a + |3⟩5 a - |5⟩5 a +
   |6⟩5 a - |9⟩5 a - |11⟩5 b - |13⟩5 b + |14⟩5 b - |16⟩5 b + |19⟩5 b + |1⟩5 b - |21⟩5 b - |22⟩5 b - |25⟩5 b + |26⟩5 b - |28⟩5 b + |2⟩5 b -
   |31⟩5 b + |4⟩5 b - |7⟩5 b - |8⟩5 b - i |11⟩5 c + i |13⟩5 c + i |14⟩5 c + i |16⟩5 c + i |19⟩5 c - i |1⟩5 c + i |21⟩5 c - i |22⟩5 c +
   i |25⟩5 c + i |26⟩5 c + i |28⟩5 c + i |2⟩5 c - i |31⟩5 c - i |4⟩5 c - i |7⟩5 c + i |8⟩5 c + |12⟩5 d + |15⟩5 d + |17⟩5 d + |18⟩5 d -
   |20⟩5 d + |23⟩5 d + |24⟩5 d + |27⟩5 d - |29⟩5 d + |30⟩5 d - |3⟩5 d - |5⟩5 d - |6⟩5 d - |9⟩5 d + |10⟩5 (-a + d) + |0⟩5 (a + d)) α +
 1/2 (|14⟩5 a - |16⟩5 a + |19⟩5 a - |1⟩5 a - |21⟩5 a - |22⟩5 a + |25⟩5 a - |26⟩5 a + |28⟩5 a - |2⟩5 a + |31⟩5 a - |4⟩5 a + |7⟩5 a -
   |8⟩5 a - |0⟩5 b - |10⟩5 b + |12⟩5 b - |15⟩5 b + |17⟩5 b - |18⟩5 b - |20⟩5 b - |23⟩5 b - |24⟩5 b + |27⟩5 b + |29⟩5 b + |30⟩5 b -
   |3⟩5 b + |5⟩5 b - |6⟩5 b - |9⟩5 b + i |0⟩5 c - i |10⟩5 c - i |12⟩5 c - i |15⟩5 c - i |17⟩5 c - i |18⟩5 c + i |20⟩5 c - i |23⟩5 c +
   i |24⟩5 c + i |27⟩5 c - i |29⟩5 c + i |30⟩5 c - i |3⟩5 c - i |5⟩5 c - i |6⟩5 c + i |9⟩5 c - |14⟩5 d - |16⟩5 d - |19⟩5 d - |1⟩5 d -
   |21⟩5 d + |22⟩5 d + |25⟩5 d + |26⟩5 d + |28⟩5 d + |2⟩5 d - |31⟩5 d - |4⟩5 d - |7⟩5 d - |8⟩5 d + |11⟩5 (-a + d) - |13⟩5 (a + d)) β
```

The circuit identifying an error looks like as

In[34]:=
```
mat21 = {{1, 1, 1, 1, Z, 1, 1, 1, X, 1, 1, 1, X, 1, 1, 1, Z, 1},
       {1, 1, 1, X, 1, 1, 1, X, 1, 1, 1, Z, 1, 1, 1, 1, 1, 1},
       {1, 1, X, 1, 1, 1, Z, 1, 1, 1, 1, 1, 1, 1, 1, Z, 1, 1},
       {1, Z, 1, 1, 1, 1, 1, 1, 1, 1, Z, 1, 1, 1, X, 1, 1, 1},
       {1, 1, 1, 1, 1, Z, 1, 1, 1, X, 1, 1, 1, X, 1, 1, 1, 1},
       {H, C, C, C, C, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, H},
       {H, 1, 1, 1, 1, C, C, C, C, 1, 1, 1, 1, 1, 1, 1, 1, H},
       {H, 1, 1, 1, 1, 1, 1, 1, 1, C, C, C, C, 1, 1, 1, 1, H},
       {H, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, C, C, C, C, H}};
```

In[35]:=
```
circuit[mat21, {x4, x3, x2, x1, x0, 0, 0, 0, 0}, {, , , , , , , ,}]
```

Out[35]=



Note that four ancillary qubits set initially in the state $|0\rangle$ are used to diagnose an error and they will be use later as control qubits in the circuit correcting the error. To apply the unitary operator corresponding to the circuit diagnosing an error we need to rewrite corrupted vector in the form of 9-qubit vector.

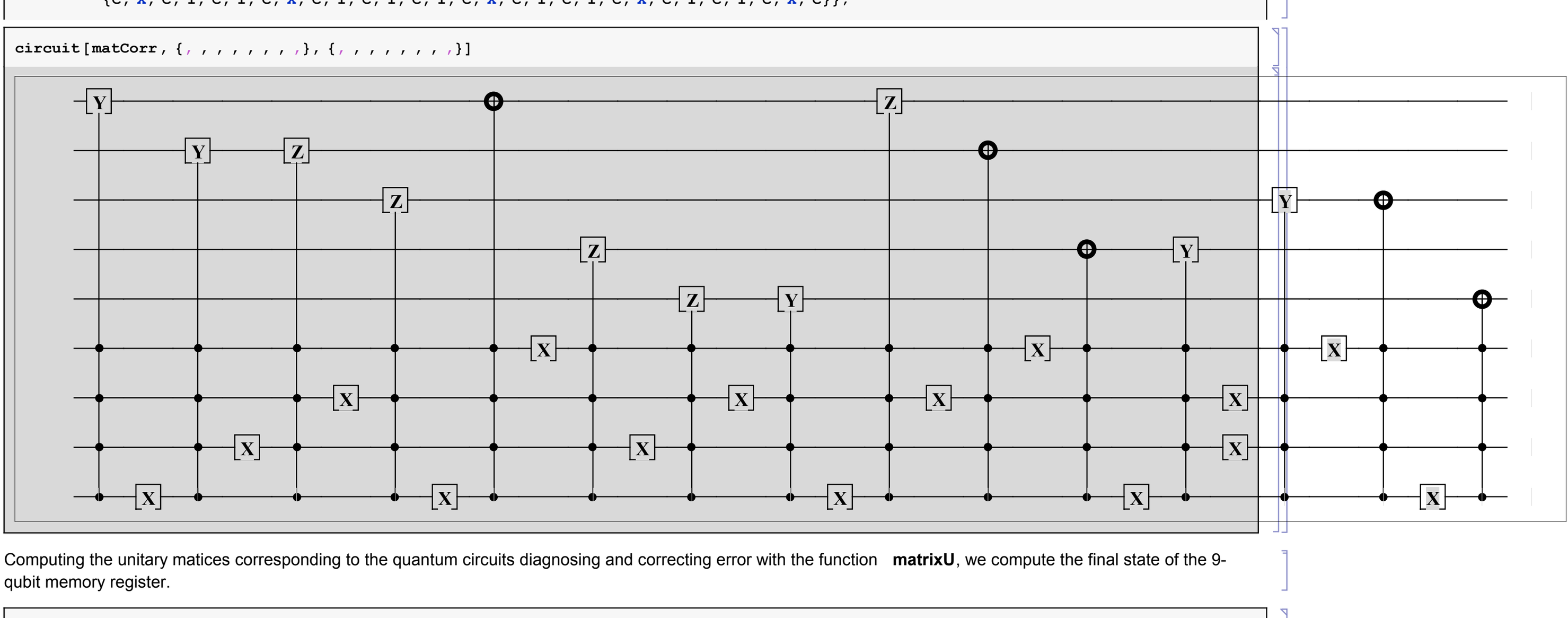
In[36]:=
```
vecError9 = Table[0, {2^9}];
Do[vecError9〚1 + 16 k〛 = vecErr[[k + 1]],
 {k, 0, 2^5 - 1}]
```

The circuit correcting the eroor is defined as follows.

In[38]:=
```
matCorr = {{Y, 1, 1, 1, 1, 1, 1, 1, X, 1, 1, 1, 1, 1, 1, 1, Z, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
         {1, 1, Y, 1, Z, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, X, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
         {1, 1, 1, 1, 1, 1, Z, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, Y, 1, X, 1, 1},
         {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, Z, 1, 1, 1, 1, 1, 1, 1, 1, 1, X, 1, Y, 1, 1, 1, 1, 1, 1},
         {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, Z, 1, Y, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, X},
         {C, 1, C, 1, C, 1, C, 1, C, X, C, 1, C, 1, C, 1, C, 1, C, X, C, 1, C, 1, C, X, C, 1, C},
         {C, 1, C, 1, C, X, C, 1, C, 1, C, 1, C, X, C, 1, C, X, C, 1, C, 1, C, X, C, 1, C, 1, C},
         {C, 1, C, X, C, 1, C, 1, C, 1, C, X, C, 1, C, 1, C, 1, C, 1, C, 1, C, X, C, 1, C, 1, C},
         {C, X, C, 1, C, 1, C, X, C, 1, C, 1, C, 1, C, X, C, 1, C, 1, C, X, C, 1, C, 1, C, X, C}};
```

In[39]:=
```
circuit[matCorr, {, , , , , , , ,}, {, , , , , , , ,}]
```

Out[39]=



Computing the unitary matices corresponding to the quantum circuits diagnosing and correcting error with the function **matrixU**, we compute the final state of the 9-qubit memory register.

In[40]:=
```
vecCorrected = matrixU[matCorr].matrixU[mat21].vecError9 // Normal // Simplify;
vecCorr1 = {};
Do[
 vecCorr1 = Append[vecCorr1, vecCorrected〚k〛 StringJoin["|", ToString[FromDigits[Drop[IntegerDigits[k - 1, 2, 9], -4], 2]], "⟩5"]
    StringJoin["|", ToString[FromDigits[Drop[IntegerDigits[k - 1, 2, 9], 5], 2]], "⟩4"]] , {k, Length[vecCorrected]}]
```

In[43]:=
```
vecCorr2 = Apply[Plus, DeleteCases[vecCorr1, 0]] // Collect[#, {α, β}, Simplify] &;
Map[Collect[#, {α, β}, Simplify] &, vecCorr2 // Factor, {1}]
```

Out[44]=
```
1/4 (|11⟩4 a + |1⟩4 b + |0⟩4 c + |10⟩4 d)
 ((|0⟩5 - |10⟩5 + |12⟩5 - |15⟩5 + |17⟩5 - |18⟩5 - |20⟩5 - |23⟩5 + |24⟩5 - |27⟩5 - |29⟩5 - |30⟩5 + |3⟩5 - |5⟩5 + |6⟩5 - |9⟩5) α +
   (- |11⟩5 - |13⟩5 + |14⟩5 - |16⟩5 + |19⟩5 - |1⟩5 - |21⟩5 - |22⟩5 + |25⟩5 - |26⟩5 + |28⟩5 - |2⟩5 + |31⟩5 - |4⟩5 + |7⟩5 - |8⟩5) β)
```

In[45]:=
```
vecIn
```

Out[45]=
```
1/4 (|0⟩5 - |10⟩5 + |12⟩5 - |15⟩5 + |17⟩5 - |18⟩5 - |20⟩5 - |23⟩5 + |24⟩5 - |27⟩5 - |29⟩5 - |30⟩5 + |3⟩5 - |5⟩5 + |6⟩5 - |9⟩5) α +
 1/4 (- |11⟩5 - |13⟩5 + |14⟩5 - |16⟩5 + |19⟩5 - |1⟩5 - |21⟩5 - |22⟩5 + |25⟩5 - |26⟩5 + |28⟩5 - |2⟩5 + |31⟩5 - |4⟩5 + |7⟩5 - |8⟩5) β
```

Conclusion

In this talk we have presented application of our *Mathematica* package to simulation of quantum circuits correcting quantum errors arising in one qubit during transmission of quantum information.
Note that the package provides a user-friendly graphical interface to specify a quantum circuit, to draw it, and to construct the corresponding unitary matrix for quantum computation defined by the circuit. The matrix is computed by means of the linear algebra tools built into *Mathematica*.
Using the unitary matrix, we can evaluate probability of different results after measurement and to check different quantum algorithms.
Our simulation confirms that quantum error correction is possible if the noise level in the transmission channel is sufficiently small.