phys. stat. sol. (b) **233**, No. 3, 482–489 (2002)

Implementation of the Four-Bit Deutsch–Jozsa Algorithm with Josephson Charge Qubits

N. SCHUCH[1]) and J. SIEWERT*)

Institut für Theoretische Physik, Universität Regensburg, 93040 Regensburg, Germany

(Received May 13, 2002; accepted July 9, 2002)

PACS: 03.67.Lx; 73.23.–b; 85.25.Cp

We show that the Deutsch-Jozsa algorithm for up to four qubits can be realized with a setup of Josephson charge qubits. While existing proposals for the implementation of the algorithm (for up to three qubits) are based on certain classifications of the oracles, this approach becomes increasingly cumbersome for higher qubit numbers. Here we present a method to implement all balanced functions for the four-qubit algorithm by a fixed sequence of operations. The free parameters which define the implemented function are the rotation angles of 15 one-qubit operations.

1. Introduction During the past few years, implementations for quantum bits in many different areas of physics have been proposed. That is, potentially there are many physical systems which are suitable to realize one-qubit and two-qubit operations. Due to the universality of quantum computation this is, in principle, sufficient to realize any arbitrary N-bit quantum operation [1].

At the present technological level, however, the feasibility of quantum computation still crucially depends on the details of a given physical implementation such as the available Hamiltonians (and therefore, in particular, the available two-bit operations), the maximal decoherence time, the scalability, etc. Therefore, the test whether or not a certain hardware setup can really be used as a quantum computer is the realization of a complete algorithm.

Here we investigate the implementation of the Deutsch–Jozsa algorithm in a system of Josephson charge qubits [2]. Superconducting nanocircuits are promising candidates for the realization of a quantum computer. Recently, considerable progress has been achieved in the experimental design and control of the quantum dynamics of such systems [3–7].

The Deutsch problem was the first example for which an algorithm on a quantum computer can outperform the solution on a classical computer [8]. It has been proposed by Deutsch in 1985 and since then has undergone several refinements by Deutsch, Jozsa, and others [9–11]. The implementation of the Deutsch–Jozsa algorithm has been realized for up to three qubits in liquid state NMR [12]; proposals exist to implement the three-bit algorithm also for Josephson charge qubits [13] and for molecular systems [14].

Systematic implementations for three qubits start from certain classifications of the oracles. Since this approach becomes rather cumbersome for four and more qubits, a universal strategy for implementing the oracles becomes desirable, i.e. the treatment of

[1]) Corresponding author e-mail: norbert.schuch@physik.uni-regensburg.de
*) e-mail: jens.siewert@physik.uni-regensburg.de

 0370-1972/02/23310-0482 $ 17.50+.50/0

all oracles on the same footing. Here we present such a universal implementation of *all* four-bit Deutsch oracles, where the selection of the function can be done by changing 15 well-defined one-bit operations. The parameters of the operation sequence can be determined straightforwardly from the function which has to be encoded in the oracle.

2. Josephson Charge Qubits Before we discuss the implementation of the algorithm we consider a specific hardware setup in order to see what kind of operations are available in a real system. A Josephson charge qubit consists of a superconducting island, coupled to a voltage source by a capacitor and to a superconducting reservoir by a tunable Josephson junction (Fig. 1). There are two characteristic energy scales for this system: the charging energy $\mathcal{E}_c$ and the Josephson energy J of the junction. The Josephson junction is designed as a SQUID loop such that J can be tuned by applying a magnetic flux Φ. At sufficiently low temperatures $T \ll J \ll \mathcal{E}_c$ only two charge states of the superconducting island are important for the dynamics of the system. In this regime, the system behaves as a two-level system $\{|0\rangle, |1\rangle\}$ to a good approximation (see Ref. [2]) with the Hamiltonian

$$H_{1q} = -\tfrac{1}{2} E_c(V)\, \sigma_z - \tfrac{1}{2} J(\Phi)\, \sigma_x \,.$$

The energy $E_c(V)$ is proportional to the charging energy $\mathcal{E}_c$ and can be controlled by the gate voltage V.

In order to do quantum computation with Josephson charge qubits we also need a controllable coupling between the qubits. Here we consider nearest-neighbor coupling via a Josephson junction threaded by a magnetic flux [13, 15, 16].

For this coupling, the interaction Hamiltonian is

$$H_{int}^{(i)} = -\tfrac{1}{2} J_{int}^{(i)}(\Phi_{int}^{(i)})\, (\sigma_+^{(i)} \sigma_-^{(i+1)} + \text{h.c.}) \,,$$

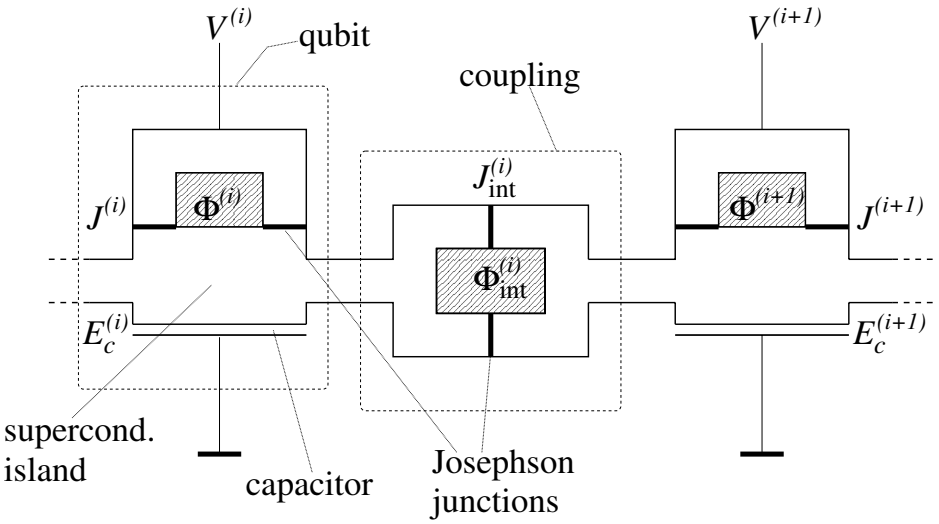


Fig. 1. The hardware setup used. Two Josephson charge qubits are coupled by a symmetric SQUID loop. The setup used for four qubits consists of four qubits arranged in a circle, so we have interactions between the pairs 1–2, 2–3, 3–4 *and* 4–1

where we have assumed that the capacitive part of the coupling can be neglected [13]. The full Hamiltonian is now the sum of H_{1q} and H_{int} over all qubits. This Hamiltonian allows us to perform all necessary operations for quantum computing, the system is *universal.*

2.1 One-qubit operations By switching off the Josephson energy for a time t, we can achieve a rotation around the z axis:

$$R_z(\phi) = \mathrm{e}^{-i\left[-\frac{1}{2}E_c\sigma_z\right]t/\hbar} = \begin{pmatrix} \mathrm{e}^{i\phi/2} & 0 \\ 0 & \mathrm{e}^{-i\phi/2} \end{pmatrix}, \tag{1}$$

where $E_c t = \hbar\phi$. Here we have written the matrix for R_z with respect to the standard charge basis $\{|0\rangle, |1\rangle\}$ (correspondingly we use the product basis for more than one qubit: $\{|000\rangle, |001\rangle, |010\rangle, \ldots\}$). In quantum circuits, we will denote a $R_z(\phi)$-operation by $-[\phi]_z-$.

On the other hand, if $E_c(V)$ is set to zero we obtain a rotation around the x axis

$$R_x(\phi) = \mathrm{e}^{-i\left[-\frac{1}{2}J\sigma_x\right]t/\hbar} = \begin{pmatrix} \cos(\phi/2) & i\sin(\phi/2) \\ i\sin(\phi/2) & \cos(\phi/2) \end{pmatrix},$$

where $Jt = \hbar\phi$. In quantum circuits, a $R_x(\phi)$-operation will be denoted by $-[\phi]_x-$.

The one-bit Hadamard gate can be generated as follows:

$$-\boxed{H}- \equiv -[\pi/2]_z[\pi/2]_x[\pi/2]_z\,.$$

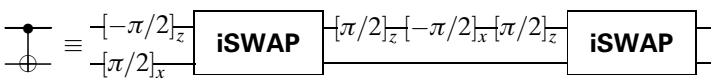
2.2 Two-qubit operations If the local energies $E_c(V)$, $J(\Phi)$ of the two qubits involved are set to zero and only the coupling energy J_{int} is finite the resulting two-qubit Hamiltonian is

$$\tilde{H}_{int} = -\frac{1}{2}\,J_{int}\begin{pmatrix} 0 & & & \\ & 0 & 1 & \\ & 1 & 0 & \\ & & & 0 \end{pmatrix}$$

which, if applied for a time $t = \pi\hbar/J_{int}$, leads to the so-called iSWAP gate

$$\mathsf{iSWAP} = \mathrm{e}^{-i\tilde{H}_{int}t/\hbar} = \begin{pmatrix} 1 & & & \\ & 0 & i & \\ & i & 0 & \\ & & & 1 \end{pmatrix}.$$

The iSWAP gate is universal. This can be easily shown by giving a sequence for generating CNOT using iSWAP and one-bit operations since CNOT is known to be universal for quantum computation [17]. One possible sequence is



. (2)

3. The Deutsch–Jozsa Algorithm First, we give a brief description of the Deutsch–Jozsa algorithm. We focus on the refined version by Collins et al. [11].

Consider the Boolean functions $f : \{0,1\}^N \to \{0,1\}$ on N bits where it is promised that f is either *constant* or *balanced* (which means the number of 0s and 1s as outcomes of f is equal). The Deutsch–Jozsa algorithm then determines – for a given unknown function f – whether the function is constant or balanced. The algorithm works as follows:

1. All qubits are prepared in the initial state $|0\rangle$, therefore the system is in the state $|00\ldots0\rangle$,
2. An N-qubit *Hadamard* transformation $\mathcal{H}$ is performed:

$$\mathcal{H} : |\mathbf{x}\rangle \mapsto \sum_{\mathbf{y}\in\{0,1\}^N} (-1)^{\mathbf{x}\cdot\mathbf{y}} |\mathbf{y}\rangle , \qquad (\mathbf{x} \in \{0,1\}^N) ,$$

where $\mathbf{x}\cdot\mathbf{y} = x_1y_1 \oplus \ldots \oplus x_Ny_N$. (The symbol $\oplus$ denotes the logical XOR operation.) This is the same as applying the one bit Hadamard transformation to each qubit separately.

3. Apply the function f which is encoded in the so-called *oracle* U_f as follows:

$$U_f : |\mathbf{x}\rangle \mapsto (-1)^{f(\mathbf{x})} |\mathbf{x}\rangle , \qquad (\mathbf{x} \in \{0,1\}^N) .$$

4. Perform another Hadamard transformation $\mathcal{H}$.
5. Measure the final state of the register. If the outcome is $|00\ldots0\rangle$, the function f is constant; otherwise it is balanced. This is because the amplitude of the state $|00\ldots0\rangle$ is

$$a_{|00\ldots0\rangle} = \frac{1}{2^N} \sum_{\mathbf{x}\in\{0,1\}^N} (-1)^{f(\mathbf{x})} .$$

The actual computation in this algorithm takes place in step 4, when the second Hadamard transformation is done. The oracle U_f is simply a black box which encodes f. However, since we want to demonstrate the feasibility to realize the entire Deutsch–Jozsa algorithm we need to devise ways how to implement *both* the algorithm (steps 1, 2, 4, 5) *and* the oracle (step 3).

Indeed, it turns out that the crucial step in implementing the Deutsch–Jozsa algorithm is the oracle since the other operations (the preparation of the state (step 2) and the computation (step 4)) are Hadamard transformations, which, in fact, are one-bit operations. Their implementation can be regarded as "simple" since it does not depend on the number of qubits, and therefore requires no further consideration.

The oracle encodes either a constant or a balanced function. For any qubit number N there are only two constant functions. In this case the oracle is the identity operator (up to an irrelevant global phase shift). On the other hand, the number of balanced functions grows quickly with N (see Table 1). Therefore the interesting part in building the oracle is the implementation of the balanced functions.

In the following we will briefly describe the existing implementations for three qubits by considering the approach in Ref. [13]. The discussion will highlight the difficulties

Table 1
The number of balanced functions as a function of the number of qubits

qubits	1	2	3	4	5	6	...
balanced functions	2	6	70	12870	6×10^8	1.8×10^{18}	...

which arise if one tries to extend this approach to higher qubit numbers, thus motivating the quest for a different method.

In order to implement the Deutsch oracle for three qubits it is useful to classify the balanced functions. Then, in some way, one deduces the sequence of operations to encode a given function from the class to which it belongs. A classification can be found, e.g., by considering the type of entanglement the oracle produces. There are completely separable oracles as well as oracles entangling two of the three qubits and finally oracles entangling all three qubits. Moreover, the fully entangled class splits into two subclasses. These classes differ by the operations required to build the oracle: in one case, *two* two-bit operations are needed whereas in the other case three two-bit operations are necessary.

This classification corresponds to identifying all functions f which can be transformed into each other by means of one-bit phase shifts or permutations of the qubit numbers. One might think that this approach can easily be extended to four qubits. However, this turns out to be difficult since the symmetry of the four-bit setup (with the qubits arranged in a ring) with respect to the permutation of qubit numbers is reduced compared to the three-bit case. Taking into account only symmetry-preserving permutations we obtain 178 different classes.

Obviously, this procedure becomes increasingly difficult for higher qubit numbers. Note that, even if all the classes are known, one still needs to find out (i) to which class a given function f belongs and (ii) how the functions in this class can be implemented.

Therefore, what one has to look for is a universal strategy which allows to generate *all* balanced functions without having to classify them in the first place.

4. Universal Strategy to Implement the Deutsch Oracle for Four Qubits In order to implement the Deutsch oracle for four qubits, we first note that the oracle mapping

$$U_f : |\mathbf{x}\rangle \mapsto (-1)^{f(\mathbf{x})} \, |\mathbf{x}\rangle$$

is equivalent to the mapping

$$S_f : |\mathbf{x}\rangle \mapsto \mathrm{e}^{i\theta(\mathbf{x})} \, |\mathbf{x}\rangle \, ,$$

where the phase shift $\theta(\mathbf{x})$ applied to the state $|\mathbf{x}\rangle$ is $\theta(\mathbf{x}) = \pi f(\mathbf{x})$. We now start to rewrite the function $\theta(\mathbf{x})$ in order to obtain a polynomial in $x_1, \ldots, x_N$. To this end we write

$$\theta(\mathbf{x}) = \pi f(\mathbf{x}) = \sum_{\{\mathbf{y} : f(\mathbf{y})=1\}} \pi \delta_{\mathbf{x},\mathbf{y}} \, , \tag{3}$$

where the sum is taken over all $\mathbf{y}$ with $f(\mathbf{y}) = 1$.

Per definition $\delta_{\mathbf{x},\mathbf{y}} = 1$ exactly if $x_i = y_i$ for all i. Now, if $y_i = 1$ for an i, this requires also x_i to be 1, or TRUE (if interpreted as a logical value). If $y_i = 0$, this means "NOT(x_i) is TRUE", or $(1 - x_i) = 1$ (note that this expression also has only the two outcomes 0 and 1). Since these conditions must be fulfilled *for all* i, we need to take the AND of all conditions, or, since they give either 0 or 1, we simply need to multiply them. So each $\delta_{\mathbf{x},\mathbf{y}}$ corresponds to a product over $i = 1, \ldots, N$ with either x_i or $1 - x_i$ in each factor, so finally $\delta_{\mathbf{x},\mathbf{y}}$ can be expanded into a polynomial in x_i, $i = 1, \ldots, N$. Thus, the same also holds for $\theta(\mathbf{x})$ since according to Eq. (3), $\theta(\mathbf{x})$ can be written as a sum of Kronecker deltas.

Now we make use of the identity

$$x_i x_j = \tfrac{1}{2}\left(x_i + x_j - (x_i \oplus x_j)\right), \qquad i \neq j\,. \tag{4}$$

By repeatedly applying this relation, we can remove all products of the variables x_i in $\theta(\mathbf{x})$ and transform it into a (weighted) sum of XORs of the x_i.

To clarify the method, we will give an example (for reasons of brevity for three qubits only). Let us consider the balanced function

$$(f(000), f(001), \ldots, f(110), f(111)) = (1,1,1,0,0,1,0,0)\,.$$

The phase shift $\theta(\mathbf{x})$ corresponding to this function is

$$\theta(\mathbf{x}) = \pi\delta_{000,\mathbf{x}} + \pi\delta_{001,\mathbf{x}} + \pi\delta_{010,\mathbf{x}} + \pi\delta_{101,\mathbf{x}}\,. \tag{5}$$

We now rewrite this in terms of polynomials in $\{x_1, x_2, x_3\}$:

$$\begin{aligned}
\delta_{000,\mathbf{x}} &= (1-x_1)(1-x_2)(1-x_3) = 1 - x_1 - x_2 - x_3 + x_1x_2 + x_1x_3 + x_2x_3 - x_1x_2x_3\\
\delta_{001,\mathbf{x}} &= (1-x_1)(1-x_2)x_3 = x_3 - x_1x_3 - x_2x_3 + x_1x_2x_3\\
\delta_{010,\mathbf{x}} &= (1-x_1)\,x_2(1-x_3) = x_2 - x_1x_2 - x_2x_3 + x_1x_2x_3\\
\delta_{101,\mathbf{x}} &= x_1(1-x_2)\,x_3 = x_1x_3 - x_1x_2x_3\,.
\end{aligned}$$

By inserting these identities into Eq. (5) we obtain

$$\theta(\mathbf{x}) = \pi[1 - x_1 - x_2x_3 + x_1x_3]\,.$$

By virtue of Eq. (4) we can transform this into

$$\theta(\mathbf{x}) = \pi - \frac{\pi}{2}x_1 - \frac{\pi}{2}x_2 + \underbrace{\frac{\pi}{2}}_{\text{conditional phase shift}} \overbrace{(x_2 \oplus x_3)}^{\text{control condition}} - \frac{\pi}{2}(x_1 \oplus x_3)\,. \tag{6}$$

Recall that $\theta(\mathbf{x})$ represents the phase shift which has to be applied to the state $|\mathbf{x}\rangle$ in order to implement the oracle. We can achieve this phase shift by translating Eq. (6) into a sequence of operations: each conditional phase shift is applied if the corresponding control condition is true. In our example, we would apply a phase shift of $-\pi/2$ if $x_1 = 1$, $-\pi/2$ if $x_2 = 1$, a phase shift of $\pi/2$ if $(x_2 \oplus x_3) = 1$, and a shift of $-\pi/2$ if $(x_1 \oplus x_3) = 1$. The constant term π in Eq. (6) is a global phase factor which is irrelevant and can be dropped.

Consequently, what we have to do is to design a setup where phase shifts controlled by all possible XOR-combinations of the x_i can be done. Then the only parameters which have to be adjusted in order to define the balanced function are the values of the phase angles of the one-bit operations. The number of these conditional phase shifts is 15, since there are $2^4 - 1$ different XOR-combinations of $x_1, \ldots, x_4$. (The "empty" combination corresponds to a global phase shift).

In Fig. 2 it is illustrated how this method works. At the beginning, the lines (the qubits) have the values $x_1, \ldots, x_4$, respectively. By repeatedly applying CNOT operations, we can generate any XOR condition in an arbitrary line. To this line (=qubit), a

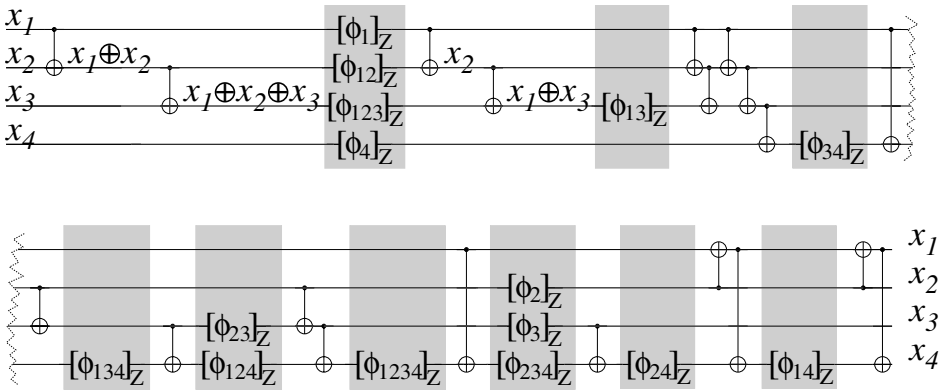


Fig. 2. The complete sequence for the four-bit Deutsch oracle. The labels on the lines give the new value after the preceding CNOT operation. The ϕ's correspond to the different phase shifts, the index lists all x_i's appearing in the control condition. Note that only next-neighbor coupling is used (the CNOT between the qubits 1 and 4 is a next-neighbor interaction, too, since we discuss a circularly coupled setup of qubits)

$\neg[\phi]_z$ operation is applied. This shifts the $|1\rangle$ state by $-\phi$ relative to the $|0\rangle$ state. By choosing ϕ appropriately we can generate the desired conditional phase shift for that control condition. The additional phase shift $\phi/2$ from Eq. (1) is applied in both cases $|0\rangle$ and $|1\rangle$; it is therefore a global phase shift and does not have any effect.

In Fig. 2 we show a complete sequence of operations for four qubits. Note that only nearest-neighbor interactions are involved. To get an operation sequence especially for Josephson charge qubits, one just has to replace all the CNOTs by the sequence in Eq. (2).

It is worth noting that this principle of generating the oracle is an extension of the method used in Ref. [17] to generate the N-bit Toffoli gate. This reference also provides hints how to efficiently build such sequences.

5. Summary We have developed a method for implementing the four-bit Deutsch oracle. The method works with a fixed setup of operations where only 15 operations have to be changed, all of them one-bit phase shifts. These operations are the "knobs" which have to be turned to the right position in order to define the behavior of our "black box". A straightforward procedure to compute the required phase shifts, i.e. the positions of the "knobs", also exists. In particular, this way to implement the Deutsch oracle is easy to handle since no classification of the functions has to be found.

Finally, we would like to point out two important advantages of a fixed operation sequence. First, from the point of view of computer science one would like to formalize the solution of a given problem. The method of the solution should not depend on the particular problem (in this case on the function which has to be implemented). Moreover, an effort to optimize the sequence applies to all possible oracles at the same time, and not only to a particular subclass.

Second, the higher the qubit number is the more it is likely that error correction is necessary. A fixed instruction setup should render it much easier to develop error correcting procedures which meet the needs of *this* setup.

References

[1] D. Deutsch, A. Barenco, and A. Ekert, Proc. R. Soc. London A **449**, 669 (1995).
[2] Y. Makhlin, G. Schön, and A. Shnirman, Rev. Mod. Phys. **73**, 357 (2001).
[3] Y. Nakamura, Yu.A. Pashkin, and J.S. Tsai, Nature **398**, 786 (1999).
[4] D. Vion et al., Science **296**, 886 (2002).
[5] J.R. Friedman et al., Nature **406**, 43 (2000).
[6] C.H. van der Wal et al., Science **290**, 773 (2000).
[7] Y. Yu et al., Science **296**, 889 (2002).
[8] D. Deutsch, Proc. R. Soc. London A **400**, 97 (1985).
[9] D. Deutsch and R. Jozsa, Proc. R. Soc. London A **439**, 553 (1992).
[10] R. Cleve et al., Proc. R. Soc. London A **454**, 339 (1998).
[11] D. Collins, K.W. Kim, and W.C. Holton, Phys. Rev. A **58**, R1633 (1998).
[12] D. Collins et al., Phys. Rev. A **62**, 022304 (2000).
[13] J. Siewert and R. Fazio, Phys. Rev. Lett. **87**, 257905 (2001).
[14] Z. Bihary et al., e-print: **quant-ph/0110041**.
[15] J. Siewert et al., J. Low Temp. Phys. **118**, 795 (2000).
[16] P. Echternach et al., Quantum Information and Computation **1**, 143 (2001).
[17] A. Barenco et al., Phys. Rev. A **52**, 3457 (1995).