Fast Signal Transforms for Quantum Computers

Martin Rötteler[1], Markus Püschel[2], and Thomas Beth[1]

[1] Institut für Algorithmen und Kognitive Systeme
Universität Karlsruhe
Am Fasanengarten 5
D-76128 Karlsruhe, Germany
{roettele, EISS_Office}@ira.uka.de
[2] Dept. of Electrical and Computer Engineering
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213
pueschel@ece.cmu.edu
http://avalon.ira.uka.de/home/pueschel/

Abstract. We present the discrete Fourier transform as a basic primitive in the treatment of controlled quantum systems. Based on the complexity model for quantum circuits the Fourier transform of size 2^n surprisingly can be realised with $O(n^2)$ elementary operations which is an exponential speedup compared to the classical case. This is the reason for its presence in almost all known quantum algorithms among which Shor's algorithm for factoring is the most prominent example.
We show how recent results in the theory of signal processing (for a classical computer) can be applied to obtain fast quantum algorithms for various discrete signal transforms. As an example we derive a quantum circuit implementing the discrete cosine transform $\mathrm{DCT}_{\mathrm{IV}}(8)$ efficiently.

1 Introduction

In classical computer science Turing machines are considered a unifying model of universal computation, but recently it has been realized, that, taking into account physical effects predicted by quantum mechanics, there are problems on which a putative quantum computer could outperform every classical computer (see [24] for a comparison between classical and quantum Turing machines). Quantum algorithms benefit from the superposition principle applied to the internal states of the quantum computer which are considered to be states of a (finite dimensional) Hilbert space. Therefore they lead to a new theory of computation and might be of central importance to physics and computer science. Striking examples of quantum algorithms are Shor's factoring algorithm (see [23]) and the search algorithm of Grover [14].

We give an example for the computational power immanent in the architecture of a quantum computer, namely the discrete Fourier transform which is known to have numerous applications in signal processing. It is a major ingredient for the famous algorithm of Shor [23] for finding factors of a composite number in polynomial time.

In the following we shall introduce the complexity model useful for quantum computing and give a justification for this model from physical reasons. Afterwards we will give quantum circuits for the Fourier transform (of size 2^n) on a quantum computer which will only need $O(n^2)$ basic operations.

Finally, we show how recent results concerning the automatic generation of fast signal transforms (see [17, 9, 19]) for classical computers can be used as a starting point to find a realisation suitable for a quantum computer. As an example a quantum algorithm for a certain cosine transform is given.

2 Quantum Gates

In this paper the state of a quantum computer is represented by a normalised vector Ψ in a Hilbert space $\mathcal{H}_n$ of dimension 2^n, which is given the natural tensor structure $\mathcal{H}_n = \mathbb{C}^2 \otimes \ldots \otimes \mathbb{C}^2$ (n factors). The restriction of the computational space to Hilbert spaces of this particular form is motivated by the idea of a quantum register consisting of n quantum bits. A quantum bit, also called qubit, is a state of the form

$$|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad |\alpha|^2 + |\beta|^2 = 1, \quad \alpha, \beta \in \mathbb{C}.$$

Thus the possible operations this computer can carry out theoretically are the elements of the unitary group $\mathcal{U}(2^n)$. To study the feasibility of performing unitary operations on n-qubit quantum systems given a family of elementary quantum operations we introduce the following two types of gate primitives:

– Local unitary operations on the qubit i which are of the form

$$U^{(i)} := \mathbf{1}_{2^{i-1}} \otimes U \otimes \mathbf{1}_{2^{n-i}},$$

where U is an element of the unitary group $\mathcal{U}(2)$ of 2×2-matrices.

– The controlled NOT gate (also called measurement gate) $\mathrm{CNOT}^{(i,j)}$ between the qubits i (control) and j (target) which is defined by

$$\begin{aligned} \mathrm{CNOT}^{(i,j)}\,|00\rangle &= |00\rangle\,, \mathrm{CNOT}^{(i,j)}\,|01\rangle = |01\rangle\,, \\ \mathrm{CNOT}^{(i,j)}\,|10\rangle &= |11\rangle\,, \mathrm{CNOT}^{(i,j)}\,|11\rangle = |10\rangle \end{aligned}$$

when restricted to the tensor component of the Hilbert space spanned by the qubits i and j.

We assume that these so-called elementary quantum gates can be performed with cost $O(1)$. This assumption leads to a complexity model for networks built from elementary quantum gates. The reason for treating these unitary transforms as building blocks can be illustrated by the example of the linear ion trap (see [4]):

Local rotations correspond to a laser-ion interaction involving just one ion. To perform the CNOT gate one can use the modes of the harmonic oscillator coupling all ions present in the trap. The interaction between two ions could be achieved using the center-of-mass mode, but also the higher modes could be of potential interest. Implementing these elementary gates in the linear trap is an active area of research.

The two types of elementary gates are the quantum computing analogue for the basic logical gates used in classical information processing. An important result obtained in [1] is the fact, that this family of unitary operations is universal, i. e., each unitary transform in $\mathcal{U}(2^n)$ can be factored into a sequence of controlled NOTs and local unitary operations.

Theorem 1. *The set $\mathcal{G}_n$ consisting of all local gates $U^{(i)}$ and controlled NOT gates $\mathrm{CNOT}^{(i,j)}$, where $i \neq j$, is a generating set for the unitary group $\mathcal{U}(2^n)$.*

In general only exponential upper bounds for the minimal length occurring in factorisations have been proved (see [1]) but there are many interesting classes of unitary matrices in $\mathcal{U}(2^n)$ affording only polylogarithmic word length, which means, that the minimal length k is asymptotically $O(p(n))$ where p is a polynomial.

We shall take the opportunity to introduce a graphical notation for quantum circuits which goes back to Feynman [12] and was propagated in [1]. In this notation the tensor product structure of the Hilbert space is reflected by drawing n parallel lines (so-called *quantum wires*) each of which represents one tensor component $\mathbb{C}^2$. So the lines represent the qubits and the least significant bit of the quantum register is the lowest line. A box sitting just on one wire denotes a local transform whereas the controlled NOT gate occupies two wires: one for the control and one for the target (see figure 1).

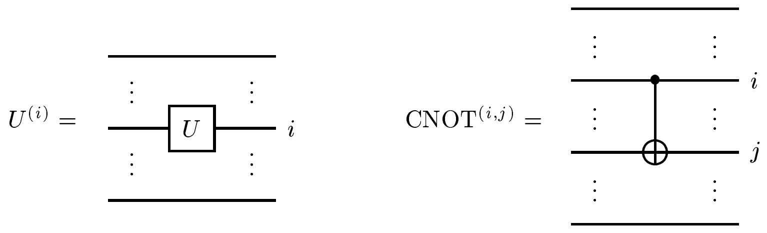

Fig. 1. Elementary quantum gates

This means that the state of the part of the register belonging to j is flipped if and only if the state in i is 1.

In the proof of theorem 1 the result is used that each unitary transform acting on a quantum register can be decomposed into a product of matrices acting only on two states s_1 and s_2 of the whole register. These transforms $T(s_1, s_2)$ can for instance be found in [22] and have the following form:

$$T(s_1,s_2)=\begin{array}{cc}
\begin{array}{ccccccccccc} &&&s_1&&&&s_2&&& \\ &&&\downarrow&&&&\downarrow&&& \end{array} & \\
\left(\begin{array}{ccccccccccc}
1 &&&&&&&&&& \\
& \ddots &&&&&&&&& \\
&& 1 &&&&&&&& \\
&&& * &&&& * &&& \\
&&&& 1 &&&&&& \\
&&&&& \ddots &&&&& \\
&&&&&& 1 &&&& \\
&&& * &&&& * &&& \\
&&&&&&&& 1 && \\
&&&&&&&&& \ddots & \\
&&&&&&&&&& 1
\end{array}\right) &
\begin{array}{c} \\ \\ \\ \longleftarrow s_1 \\ \\ \\ \\ \longleftarrow s_2 \\ \\ \\ \\ \end{array}
\end{array}$$

The submatrix formed by the $*$-entries denotes a unitary operation in $\mathcal{U}_2$. Considering these matrices the difference between local operations in the sense of quantum mechanics and classical local operations should become clear: A quantum local gate acts on *tensor components* and influences all states of the Hilbert space at cost of one basic operation in our complexity model. On the other hand the transformations $T(s_1, s_2)$ act on *subspaces* of the Hilbert space and consequently manipulate only some components of the state vector but may cost many elementary operations. Nevertheless, the rotations $T(s_1, s_2)$ on two states can be used to write each element $U \in \mathcal{U}(2^n)$ in the form

$$U = D \cdot \prod_{s_1, s_2 \in \{0,1\}^n} T(s_1, s_2),$$

where the diagonal matrix $D = \mathrm{diag}(\phi_1, \ldots, \phi_{2^n})$ takes care of occurring phase factors. Note, however, that since the whole computational space is exponentially large, an exponential number of $T(s_1, s_2)$ appears in this factorisation.

Moreover to realize a $T(s_1, s_2)$ gate it is necessary to use multiple controlled gates (which are the generalisation of CNOT gates to more than one conditioned input) and these in turn have to be broken down to gates from the set $\mathcal{G}_n$ which yields a quadratic overhead (see [1]).

One comment about the difference between the architecture of a classical computer and the computational model of the quantum computer: Considered as a transition matrix of a finite state machine (which each classical computer can be thought of if one allows a sufficiently large configuration space) a quantum computer is a Single Instruction-Multiple Data (SIMD) machine—with the big difference to the classical model of computation that the data which is manipulated stems from an exponentially large space.

3 Fourier Transforms

In this section we recall the definition and basic properties of the discrete Fourier transform (DFT) and give examples of its use in quantum computing.

In most standard texts on signal processing (see, e.g., [11]) the DFT_N of a periodic signal given by a function $f : \mathbf{Z}_N \to \mathbb{C}$ (where $\mathbf{Z}_N$ is the cyclic group of order N) is defined as the function F given by

$$F(\nu) := \sum_{t \in \mathbf{Z}_N} e^{2\pi i \nu t/N} f(t).$$

If we equivalently adapt the point of view that the signal f and the Fourier transform are vectors in $\mathbb{C}^N$ we see that performing the DFT_N is a matrix vector multiplication of f with the unitary matrix

$$\mathrm{DFT}_N := \frac{1}{\sqrt{N}} \cdot \left[\omega^{i \cdot j}\right]_{i,j=0,\ldots,N-1},$$

where $\omega = e^{2\pi i/N}$ denotes a primitive N-th root of unity.

From an algebraic point of view the DFT_N gives an isomorphism Φ of the group algebra $\mathbb{C}\mathbf{Z}_N$

$$\Phi : \mathbb{C}\mathbf{Z}_N \longrightarrow \mathbb{C}^N$$

onto the direct sum of the irreducible matrix representations of $\mathbf{Z}_N$ (where multiplication is performed pointwise). This means that DFT_N decomposes the regular representation of $\mathbf{Z}_N$ into its irreducible constituents. It is known that this property allows the derivation of a fast convolution algorithm (see [2]) in a canonical way and can be generalised to more general group circulants.

The view towards the DFT as a decomposition matrix for the regular representation leads to the generalisation of Fourier transforms to arbitrary finite groups (cf. [2]).

Very important for applications in classical signal processing is the fact that DFT_N can be computed in $O(N \log N)$ arithmetic operations (counting additions and multiplications). This possibility to perform a Fast Fourier Transform justifies the heavy use of the DFT_N in signal processing. In the next section we will show that the $O(N \log N)$ bound which is sharp in the arithmetic complexity (see [5], chapters 4 and 5) can be improved on a quantum computer to $O(\log^2 N)$ operations.

3.1 Getting Quantum

From now on we restrict ourselves to DFT_N where $N = 2^n$ is a power of 2 since these transforms naturally fit to the tensor structure imposed by the qubits.

The implementation of the Fourier transform on a quantum computer ([7, 23]) starts from the well-known Cooley-Tukey decomposition (see [6, 2]): after performing a certain row permutation Π_n (usually called *bit-reversal*) the DFT_{2^n} has the following block structure:

$$\begin{aligned}\Pi_n \, \mathrm{DFT}_{2^n} &= \left(\begin{array}{c|c} \mathrm{DFT}_{2^{n-1}} & \mathrm{DFT}_{2^{n-1}} \\ \hline \mathrm{DFT}_{2^{n-1}} \, W_n & -\mathrm{DFT}_{2^{n-1}} \, W_n \end{array}\right) \\ &= (\mathbf{1}_2 \otimes \mathrm{DFT}_{2^{n-1}}) \cdot T_n \cdot (\mathrm{DFT}_2 \otimes \mathbf{1}_{2^{n-1}})\end{aligned}$$

Here we denote by

$$T_n := \mathbf{1}_{2^{n-1}} \oplus W_n, \quad W_n := \begin{pmatrix} 1 & & & & \\ & \omega_{2^n} & & & \\ & & \omega_{2^n}^2 & & \\ & & & \ddots & \\ & & & & \omega_{2^n}^{2^{n-1}} \end{pmatrix}$$

the matrix of *Twiddle factors* (see [2]). Taking into account the fact that T_n has the tensor decomposition

$$T_n = \mathbf{1}_{2^{n-1}} \oplus \begin{pmatrix} 1 & \\ & \omega_{2^n}^{n-1} \end{pmatrix} \otimes \ldots \otimes \begin{pmatrix} 1 & \\ & \omega_{2^n} \end{pmatrix}$$

we see that T_n can be implemented by gates which have one control wire. These can be factored into the elementary gates $\mathcal{G}_n$ with constant overhead. The bit-reversal Π_n is nothing but a permutation of the quantum wires and can be implemented with cost $O(n)$ (see [20]).

Because tensor products are free in our computation model we arrive at an upper bound for the computation of the Fourier transform on a quantum computer of $O(n^2)$.

3.2 An Application: Shor's algorithm

In this section we briefly review Shor's factorisation algorithm and show how the Fourier transform comes into play. It is known that factoring a number N is easy under the assumption that it is easy to determine the (multiplicative) order of an arbitrary element in $(\mathbf{Z}_N)^\times$. For a proof of this we refer the reader to Shor's paper [23].

Having done this reduction the following observation is the crucial step for the quantum algorithm: Let y be randomly chosen and $\gcd(y, N) = 1$. To determine the multiplicative order r of y mod N consider the function

$$f_y(x) := y^x \bmod N.$$

Clearly $f_y(x + r) = f_y(x)$, i.e., f_y is a periodic function with period length r. The quantum algorithm is as follows:

Algorithm 2 *Let $N^2 \leq Q \leq 2N^2$ be given. This number Q will be the length of the Fourier transform to be performed in the sequel.*

1. *Randomly choose y with $(y, N) = 1$.*
2. *Prepare the state $|0\rangle \otimes |0\rangle$ on the two registers, each of which has length $\lceil \log_2 N \rceil$. To carry out the computation on a real physical system this is done by "cooling it down" to a well-defined ground-state. In case of the ion trap this can be taken literally whereas in an NMR computer the preparation of the $|0\rangle$ state is one of the major problems.*

3. *Application of the Fourier transform DFT_Q to the left part of the register results in a superposition of all possible inputs*

$$\frac{1}{\sqrt{Q}} \sum_{x=0}^{Q-1} |x\rangle \otimes |0\rangle \,.$$

4. *Calculation of the function $f_y(x) = y^x \bmod N$ yields for this superposition (normalisation omitted):*

$$\sum_{x=0}^{Q-1} |x\rangle \otimes |y^x \bmod N\rangle \,.$$

5. *Measuring the right part of the register gives a certain value z_0. The remaining state is the superposition of all x satisfying $f_y(x) = z_0$:*

$$\sum_{y^x = z_0} |x\rangle \, |z_0\rangle = \sum_{k=0}^{Q-1} |x_0 + kr\rangle \, |z_0\rangle \,, \quad \textit{where } y^{x_0} = z_0 .$$

6. *Performing a* DFT *on the left part of the register leads to*

$$\sum e^{2\pi i x_0 l / r} \left| l \frac{Q}{r} \right\rangle |z_0\rangle \,.$$

7. *Finally a measurement of the left part of the register gives a value $l_0 \frac{Q}{r}$.*

Iterated application of this algorithm produces a set $\{l_i \frac{Q}{r}\}$ from which after a classical post-processing involving Diophantine approximation (see [23]) the period r can be extracted.

A thorough analysis of this algorithm must take into account the overhead for the calculation of the function $f_y : x \mapsto y^x \bmod N$. However, having realised this function as a quantum network once (which can be obtained from a classical network for this function in polynomial time) the superposition principle applies since all inputs can be processed by *one* application of f_y.

The rôle played by the Fourier transform in this algorithm is twofold: In step 3 it is used to generate a superposition of all inputs from the ground state $|0\rangle$. But this could have also been done by *any* unitary transform having the all-one vector in the first column. In step 6 the use of the DFT is much more fundamental because it extracts the information about the period r which was hidden in the graph of the function f_y. This property of feature-extraction is also discussed in the following section.

3.3 The Hidden Subgroup Problem

The approach to the factorisation problem sketched in the preceding section can be boiled down to the basic principle of algorithmic pattern recognition, namely separating preimages under a map. This principle becomes apparent in the hidden subgroup problem which we will briefly describe in the following (see also [3] for an introduction).

Consider a function $f : G \to R$ from a finite group G to an arbitrary domain R. Suppose that we are given the promise that there is a subgroup $H \subseteq G$ such that

- f is constant on the cosets G/H: if $xH = yH$ then $f(x) = f(y)$.
- f separates the cosets of G/H: if $xH \neq yH$ then $f(x) \neq f(y)$.

Of course this can equivalently be formulated as: The map f induces a well-defined injective map $\overline{f} : G/H \to R$ of sets. The task of finding generators for H (given generators of G) is known as the *hidden subgroup problem*. It is of interest since it allows a unified formulation of Shor's algorithms for discrete logarithm and factoring and the somewhat artificial problems which were proposed as first applications for quantum computers (see [18] for more discussion). In all these examples the Fourier transform for abelian groups is used but still the question remains open whether the non-abelian Fourier transforms (see [2, 20]) can be used to find hidden subgroups of non-abelian groups.

Recently there has been much research interest in the graph isomorphism problem (see [15]), which can also be formulated as a hidden subgroup problem.

4 Quantum Signal Transforms

As the Cooley-Tukey decomposition for the DFT shows, many interesting unitary resp. orthogonal transforms in signal processing bear an inherent redundancy which can be exploited to get fast algorithms. Recent research results (see [17, 9, 19]) show that it is possible to generate fast signal transforms completely automatic in many cases using a certain kind of *symmetry* of the corresponding matrices. Based on [9, 19] these methods has been implemented in the GAP share package AREP (see [13, 10]). In this section we will sketch briefly these methods and show why they might be useful for quantum computing, too. Finally, we give a quantum implementation of a certain discrete cosine transform derived from a decomposition generated using AREP.

4.1 Decomposition via Symmetry

We suppose a discrete signal transform to be given by a matrix. A fast algorithm for the transform is usually given by a decomposition of this matrix into a product of sparse matrices (sparse = many entries equal zero). As an example consider the Cooley-Tukey decomposition in section 3.1.

The automatic generation of such a decomposition is based on representation theory of finite groups (see, e. g., [8]) and essentially consists of two steps: First a certain kind of symmetry is determined which captures part of the redundancy contained in the matrix representing the signal transform, as a second step the symmetry is used to decompose the matrix. Formally, a symmetry of a matrix M is given by a pair (ϕ, ψ) of matrix representations of the same group G satisfying

$\phi(g) \cdot M = M \cdot \psi(g)$ for all $g \in G$. Now let A_ϕ, A_ψ be decomposition matrices for ϕ, ψ resp. which means that ϕ^{A_ϕ} and ψ^{A_ψ} both are a direct sum of irreducible representations of G. Hence, the matrix $D = A_\phi^{-1} \cdot M \cdot A_\psi$ is block-diagonally sparse (follows from Schur's lemma) and we get $M = A_\phi \cdot D \cdot A_\psi^{-1}$. Of course, this decomposition is useful only if A_ϕ, A_ψ themselves can be obtained as products of sparse matrices. This is in particular possible for monomial representations of solvable groups (monomial = all matrices have exactly one non-zero entry in each row and column) which has been shown in [17, 19].

The algorithm to decompose a given matrix (discrete signal transform) M into a product of sparse matrices now reads as follows:

1. Determine a useful (monomial) symmetry (ϕ, ψ) of M.
2. Decompose ϕ and ψ with (factored) matrices A_ϕ, A_ψ resp.
3. Compute $D = A_\phi^{-1} \cdot M \cdot A_\psi$ by matrix multiplication.

Application to well-known signal transforms like the Fourier transform, Haar transform, Walsh-Hadamard transform, different cosine and sine transforms and others have yielded useful decompositions in all cases (see [19], chap. 4).

For the reader familiar with representation theory we present one of the main theorems involved in the algorithm above. The theorem is taken from [19], p. 60, and allows the stepwise decomposition of a monomial representation (we use standard notation, see e. g. [5]).

Theorem 3. *Let $N \trianglelefteq G$ be a normal subgroup of prime index p with (cyclic) transversal $T = (t^0, t^1, \ldots, t^{(p-1)})$ and ϕ a representation of degree d of N which has an extension $\overline{\phi}$ to G. Suppose that A is matrix decomposing ϕ into irreducibles, i. e. $\phi^A = \rho = \rho_1 \oplus \ldots \oplus \rho_k$ and that $\overline{\rho}$ is an extension of ρ to G. Then*

$$B = (\mathbf{1}_p \otimes A) \cdot D \cdot (DFT_p \otimes \mathbf{1}_d), \quad \text{where} \quad D = \bigoplus_{i=0}^{p-1} \overline{\rho}(t)^i, \tag{1}$$

is a decomposition matrix for $\phi \uparrow_T G$, more precisely

$$(\phi \uparrow_T G)^B = \bigoplus_{i=0}^{p-1} \lambda_i \cdot \overline{\rho},$$

where $\lambda_i : \ t \mapsto \omega_p^i$, $i = 0, \ldots, p-1$, are the p 1-dimensional representations of G arising from the factor group G/N.

Note, that in the case of $G = \mathbf{Z}_N$ and ϕ the regular representation of G, formula 1 coincides exactly with the Cooley-Tukey decomposition. Furthermore, if the size of G is a 2-power (and hence $p = 2$), formula 1 fits very well to the special architecture of a quantum computer. Figure 2 visualises the formula in terms of quantum wires.

In [20] theorem 3 is applied to obtain fast generalised quantum Fourier transforms for four series of 2-groups, including dihedral and quaternion groups.

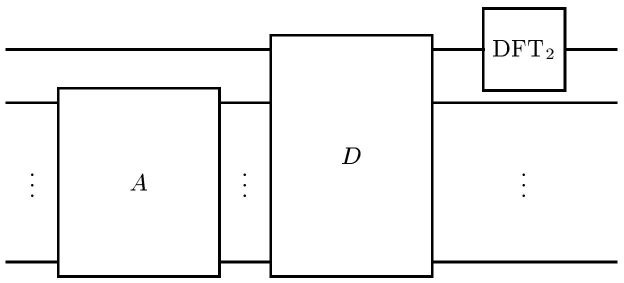

Fig. 2. Coarse quantum circuit corresponding to theorem 3

4.2 Circuit for the Discrete Cosine Transform $\mathrm{DCT}_{\mathrm{IV}}(8)$

The discrete cosine transform is a real orthogonal transform and was invented for approximating the eigenvectors of the autocorrelation matrix of an autoregressive signal with one time-step history (see [16]). The DCT comes in four different flavours ($\mathrm{DCT}_{\mathrm{I}}$, $\mathrm{DCT}_{\mathrm{II}}$, $\mathrm{DCT}_{\mathrm{III}}$ and $\mathrm{DCT}_{\mathrm{IV}}$) which vary slightly in their definitions (see [21], p. 10) and their use in computational applications. Notwithstanding this, they all have been shown to be asymptotically equivalent to the Karhunen-Loève transform for signals coming from a first-order Markov process. The $\mathrm{DCT}_{\mathrm{II}}$ is also used in the image compression standard JPEG (see [21]).

Now we are going to present a quantum circuit implementing the discrete cosine transform (type IV) of size 8 on a quantum computer. The $\mathrm{DCT}_{\mathrm{IV}}$ of size n is defined by

$$\mathrm{DCT}_{\mathrm{IV}}(n) := \left[\sqrt{2/n}\cdot\cos\left(\frac{(i+1/2)(j+1/2)\pi}{n}\right)\,\middle|\, i,j\in\{0,\ldots,n-1\}\right]$$

Many efficient implementations for the computation of a matrix vector product with a DCT are known (see, e.g., the monograph [21]). The decomposition into a product of sparse matrices which we will use as a starting point for the quantum realisation of the $\mathrm{DCT}_{\mathrm{IV}}(8)$ has been generated entirely automatic (even in the presented LaTeX-form) in [19] using AREP [10]:

$$\begin{aligned}\mathrm{DCT}_{\mathrm{IV}} = {} & [(3,4,7,6,8,5),(\omega_4,\omega_{16}^5,\omega_8^3,-\omega_{16}^7,1,-\omega_{16},\omega_8,-\omega_{16}^3)] \\ & \cdot(\mathrm{DFT}_2\otimes\mathbf{1}_4)\cdot\ \mathrm{diag}(1,1,1,1,1,\omega_8,\omega_4,\omega_8^3)\cdot\ (\mathbf{1}_2\otimes\mathrm{DFT}_2\otimes\mathbf{1}_2) \\ & \cdot\mathrm{diag}(1,1,1,\omega_4,1,1,1,\omega_4)\cdot\ (\mathbf{1}_4\otimes\mathrm{DFT}_2) \\ & \cdot\mathrm{diag}(-\omega_{64},-\omega_{64},\omega_{64}^9,-\omega_{64}^9,\omega_{64}^{23},-\omega_{64}^{23},\omega_{64}^{31},\omega_{64}^{31}) \\ & \cdot(\mathbf{1}_4\otimes\mathrm{DFT}_2)\cdot\ \mathrm{diag}(1,1,1,-\omega_4,1,1,1,-\omega_4)\cdot\ (\mathbf{1}_2\otimes\mathrm{DFT}_2\otimes\mathbf{1}_2) \\ & \cdot\mathrm{diag}(1,1,1,1,1,-\omega_8^3,-\omega_4,-\omega_8)\cdot\ (\mathrm{DFT}_2\otimes\mathbf{1}_4) \\ & \cdot[(2,6,3,4,7,5,8),(\omega_4,\omega_{16}^5,-\omega_{16}^7,\omega_8,\omega_8^3,-\omega_{16}^3,-\omega_{16},1)]\end{aligned}$$

A note on notation: $\omega_n = e^{2\pi i/n}$, $\mathrm{diag}(L)$ is the diagonal matrix with diagonal L, and, for a permutation σ and a list L, we denote by $[\sigma, L]$ the monomial matrix obtained by multiplication of the permutation matrix corresponding to σ with $\mathrm{diag}(L)$ from the right.

We observe that this matrix factorisation is highly structured: The first factor is monomial and can be implemented on the quantum computer in two steps: First factor the permutation with methods from classical boolean circuit theory in a product of (multiple) controlled NOTs, then factor the diagonal matrix which has tensor structure up to the phase factor $\tau_1 = \mathrm{diag}(1, -1, 1, 1, 1, 1, 1, 1)$: After a multiplication with τ_1 the diagonal matrix takes the form

$$\mathrm{diag}(\omega_8^2, 1) \otimes \mathrm{diag}(1, \omega_8) \otimes \mathrm{diag}(1, -\omega_{16}).$$

Next, in lines 2 and 3, we have a DFT_8 factored according to section 3.1. The diagonal matrix in line 4 can be realised as follows. After an application of the phase correcting matrix $\tau_2 = \mathrm{diag}(-1, 1, 1, 1, 1, 1, 1, -1)$ this matrix again has a complete tensor decomposition

$$\mathrm{diag}(\omega_{64}, \omega_{64}^{23}) \otimes \mathrm{diag}(1, \omega_{64}^{8}) \otimes \mathrm{diag}(1, \omega_{64}^{32}).$$

This matrix is followed by the inverse of a DFT_8 in lines 5 and 6 and finally the monomial matrix in line 7 can be factored the same way as the first factor above.

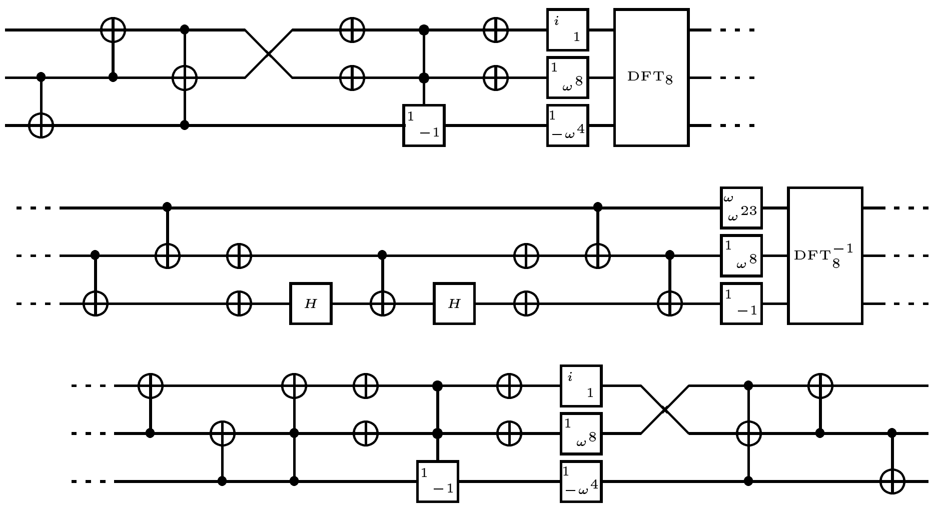

Fig. 3. Quantum circuit implementing the discrete cosine transform $\mathrm{DCT}_{\mathrm{IV}}(8)$

Altogether we obtain the quantum circuit depicted in figure 3, where $\omega = \omega_{64}$. The operation H is the Hadamard transform

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

and besides the controlled NOT gates we have also used Toffoli gates (see [1]) which are doubly controlled NOT gates.

5 Conclusions and Outlook

We presented an overview of the architecture of a quantum computer including a quantum complexity model. The discrete Fourier transform has been introduced as an important subroutine in several quantum algorithms. It is possible to realise the DFT_{2^n} with $O(n^2)$ elementary quantum gates which is an exponential speed-up compared to the classical situation.

The recently developed approach for the automatic generation of fast signal transforms has been presented as a good starting point for the realisation of signal transforms on a quantum computer. As an example the discrete cosine transform $\mathrm{DCT}_{\mathrm{IV}}$ of size 8 was factored into a small circuit of quantum gates.

A natural future area of research is to study other classes of signal transforms from the leading point of sparseness in the quantum complexity model. Candidates are classical signal transforms as well as generalised Fourier transforms for many classes of supersolvable groups.

Acknowledgments

The authors are grateful to Markus Grassl for fruitful discussions. The first author is supported by Deutsche Forschungsgemeinschaft, Graduiertenkolleg Beherrschbarkeit Komplexer Systeme under Contract No. DFG-GRK 209/3-98.

References

1. A. Barenco, Ch. Bennett, R. Cleve, D. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, and H. Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52(5):3457–3467, November 1995. LANL quant–ph/9503016.
2. Th. Beth. *Methoden der Schnellen Fouriertransformation.* Teubner, 1984.
3. G. Brassard and P. Høyer. An Exact Polynomial–Time Algorithm for Simon's Problem. In *Proceedings of Fifth Israeli Symposium on Theory of Computing and Systems*, pages 12–33. ISTCS, IEEE Computer Society Press, 1997. LANL preprint quant–ph/9704027.
4. J. I. Cirac and P. Zoller. Quantum computation with cold trapped ions. *Physical Review Letters*, 74:4091ff, 1995.
5. M. Clausen and U. Baum. *Fast Fourier Transforms.* BI-Verlag, 1993.

6. J. W. Cooley and J. W. Tukey. An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation*, 19:297–301, 1965.
7. D. Coppersmith. An Approximate Fourier Transform Useful for Quantum Factoring. Technical Report RC 19642, IBM Research Division, Yorktown Heights NY, December 1994.
8. W.C. Curtis and I. Reiner. *Methods of Representation Theory*, volume 1. Interscience, 1981.
9. S. Egner. *Zur Algorithmischen Zerlegungstheorie Linearer Transformationen mit Symmetrie.* PhD thesis, Univ. Karlsruhe, Informatik, 1997.
10. S. Egner and M. Püschel. *AREP – A Package for Constructive Representation Theory and Fast Signal Transforms, GAP Share Package*, 1998.
11. D. F. Elliott and K. R. Rao. *Fast Transforms — Algorithms, Analyses, Applications.* Academic Press, 1982.
12. R. P. Feynman. Simulating Physics with Computers. *International Journal of Theoretical Physics*, 21(6/7):467–488, 1982.
13. The GAP Team, Lehrstuhl D für Mathematik, RWTH Aachen, Germany and School of Mathematical and Computational Sciences, U. St. Andrews, Scotland. *GAP – Groups, Algorithms, and Programming, Version 4*, 1997.
14. L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)*, pages 212–219, New York, 1996. ACM.
15. J. Köbler, U. Schöning, and J. Toran. *The Graph Isomorphism Problem.* Birkhäuser, 1993.
16. H. S. Malvar. *Signal Processing with Lapped Transforms.* Artech House, Boston, 1992.
17. T. Minkwitz. *Algorithmensynthese für lineare Systeme mit Symmetrie.* PhD thesis, Universität Karlsruhe, 1993.
18. M. Mosca and A. Ekert. The Hidden Subgroup Problem and Eigenvalue Estimation on a Quantum Computer. In *Proceedings 1st NASA International Conference on Quantum Computing & Quantum Communications*, LNCS 1509. Springer, 1998.
19. M. Püschel. *Konstruktive Darstellungstheorie und Algorithmengenerierung.* PhD thesis, Univ. Karlsruhe, Informatik, 1998.
20. M. Püschel, M. Rötteler, and Th. Beth. Fast Quantum Fourier Transforms for a Class of Non-abelian Groups. LANL quant–ph/9807064.
21. K. R. Rao and R. Yip. *Discrete Cosine Transform: Algorithms, Advantages, and Applications.* Academic Press, 1990.
22. M. Reck, A. Zeilinger, H. Bernstein, and P. Bertani. Experimental Realization of Any Discrete Unitary Operator. *Physical Review Letters*, 73(1):58–61, 4. July 1994.
23. P. Shor. Algorithms for Quantum Computation: Discrete Logarithm and Factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 124–134. Institute of Electrical and Electronic Engineers Computer Society Press, November 1994.
24. U. Vazirani and E. Bernstein. Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411–1473, 1997.