The HLRB Cluster as Quantum CISC Compiler

Matrix Methods and Applications for Advanced Quantum Control by Gradient-Flow Algorithms on Parallel Clusters

T. Schulte-Herbrüggen, A. Spörl, K. Waldherr, T. Gradl, S.J. Glaser, and T. Huckle

Abstract The project encompasses matrix method developments, tailored parallelisation as well as cutting-edge applications exploiting the power of the HLRB-II cluster: fast matrix exponential algorithms using Chebyshev series are devised in view of calculating quantum dynamics of large systems. They outperform the standard Padé-approximation by a speed-up of approximately 30% in CPU time while obtaining even better accuracy. The routines are incorporated into a fully parallelised package of gradient-flow algorithms for optimal quantum control.
As an application, here we present a quantum CISC compiler: it breaks large target unitary gates into modules of effective m-qubit (i.e. two-level system) interactions. We extend the standard restricted set of modules with $m = 1,2$ (RISC) to a scalable toolbox of multi-qubit optimal controls with $m \leq 10$ forming modules of complex instruction sets (CISC). Typically, the instruction code ('experimental controls') by our quantum CISC compiler is some three to ten times faster than by RISC compilation thus dramatically saving the essential quantum coherences from unnecessary relaxative decay with time. This advantage of our method over standard universal gates is demonstrated for the indirect SWAP gate, the quantum Fourier transform as well as for multiply-controlled NOT gates.

T. Schulte-Herbrüggen, A. Spörl, and S.J. Glaser
Dept. of Chemistry, TU Munich, D-85747 Garching, Germany, e-mail: tosh@ch.tum.de

T. Huckle and K. Waldherr
Dept. of Computer Science, TU Munich, D-85747 Garching, Germany, e-mail: huckle@in.tum.de

T. Gradl
Chair of System Simulation, Dept. Computer Science, Erlangen University, D-91058 Erlangen, Germany. e-mail: tobias.gradl@informatik.uni-erlangen.de

Introduction

Quantum control plays a key role for steering quantum hardware systems in both quantum- and nano-technology [1]. However, for exploiting the power of quantum systems, one has to steer them by classical controls with the shapes of these controls critically determining the performance of the quantum system. Thus providing computational infrastructure for devising optimal shapes by using high-performance computer clusters is tantamount to exploiting the power of present and future quantum resources. The task is computationally particularly demanding since the classical resources needed grow exponentially with the size of the quantum system.

More concretely, quantum dynamics of closed systems is unitary with the system Hamiltonian H being the generator of the one-parameter group of unitary time evolution $U = e^{-itH}$. Numerical calculation of quantum dynamics thus hinges on algorithms for fast matrix exponentials. Here, we devise Chebyshev-series methods that exploit symmetries of pertinent Hamiltonians. They can be readily incorporated into a fully parallelised programme package of optimal quantum control designed to exploit the capacities of the HLRB-II cluster architecture by distributing matrix operations to different nodes with little communication and storage cost. In systems of 10 spin qubits, the time gain by parallelisation is beyond a factor of 500 on a 128-CPU cluster as compared to standard techniques on a single node of the same cluster [2].

Also in terms of computation, quantum systems provide a unique potential for coherent parallelisation that may exponentially speed-up algorithms as in Shor's prime factorisation. Again, compiling a target quantum unitary module into the machine code for steering a concrete quantum hardware device lends itself to be tackled by means of optimal quantum control. To this end, there are two different approaches: (i) one may use a decomposition into the restricted instruction set of so-called universal one- and two-qubit gates that in turn have prefabricated translations into the machine code or (ii) one may prefer to generate the entire target module directly by a complex instruction set of available controls. Here we advocate direct compilation up to the limit of system size a classical computer cluster can reasonably handle. For large systems we propose another way, namely (iii) to make recursive use of medium-sized building blocks generated by optimal control in the sense of a quantum CISC compiler. Implications for upper limits to time complexities are also derived.

Scope and Organisation of the Paper

This account comprises two sections, the first of which is dedicated to developing numerical matrix exponential and matrix multiplication algorithms on parallel clusters. The second section then presents novel applications, to wit a quantum CISC compiler. In doing so, the account shows how fast matrix methods allowing to maintain full parallelisation on high-performance clusters provide the basis for cutting-edge applications, e.g., in optimal quantum control. In turn, these control methods

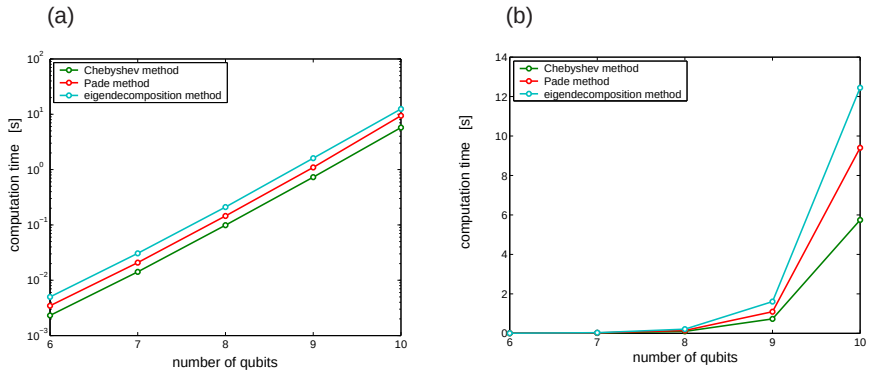

Figure 1 Comparison of the CPU times required for calculating one matrix exponential as a function of the system size, where n qubits translate into a complex matrix of dimension $2^n \times 2^n$; (a) logarithmic scale, (b) linear scale.

can be put to good use for finding optimised experimental steerings of quantum devices in realistic settings as they occur in a broad array of applications comprising quantum electronics, nanotechnology, spectroscopy, and quantum computation.

1 Development of Computational Methods

Faster algorithms for matrix exponentials on high-dimensional systems have been developed in view of application to large quantum systems. We extended our parallelised C++ code of the GRAPE package described in [2] by adding more flexibility allowing to efficiently exploit available parallel nodes independent of internal parameters. Thus computations could be performed on the HLRB-II supercomputer cluster at *Leibniz Rechenzentrum* of the Bavarian Academy of Sciences Munich. It provides an SGI Altix 4700 platform equipped with 9728 Intel Itanium2 Montecito Dual Core processors with a clock rate of 1.6 GHz, which give a total LINPACK performance of 63.3 TFlops/s. Following our previous work on time-optimal control [3], we used the GRAPE algorithm [4] in order to realise unitary target gates in shortest times still allowing for full fidelity.

1.1 Fast Matrix Exponentials

A task paramount to calculating time evolutions of quantum systems hinges on fast numerics: it is the computation of the matrix exponentials of quantum mechanical system Hamiltonians H. With H being complex Hermitian, the propagator e^{-iH} is unitary. The problem of computing e^{-iH} can be reduced to two problems of the half

size with real numbers by exploiting the persymmetry properties of H. For details see [2].

The matrix exponential e^{-iH} is defined by the infinite series

$$e^{-iH} := \sum_{k=0}^{\infty} (-iH)^k/k! \quad . \tag{1}$$

Although over the years a plethora of different methods has been devised for calculating matrix exponentials, none of them is fully satisfactory. This is why they are sometimes referred to as 'dubious' [5, 6]. Standard algorithms include the Padé approximation as well as the eigendecomposition, both of which we use as references. Here, we advocate to calculate the matrix exponential by a Chebyshev series expansion. To this end, we need the Chebyshev polynomials of the first kind given by the following three-term recurrence formula

$$T_0(x) = 1 \tag{2}$$
$$T_1(x) = x \tag{3}$$
$$T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x) \quad . \tag{4}$$

With respect to the weight function $\omega(x) = (1-x^2)^{-1/2}$, these polynomials are orthogonal. In this sense, a function $f(x)$ with arguments $|x| \leq 1$ can be represented by an infinite Chebyshev series according to

$$f(x) = \sum_{k=0}^{\infty} a_k T_k(x) \quad \text{with coefficients} \quad a_k = \frac{2}{\pi}\int_{-1}^{1} f(x)\,T_k(x)\frac{dx}{\sqrt{1-x^2}} \quad . \tag{5}$$

Details can be found in [7, 8]. In our case $f(x) = e^x$ and the coefficients then take the special form $a_k = 2i^k J_k(-i)$ with the Bessel functions J_k. For $|x| \leq 1$ this leads to $e^x = J_0(i) + 2\sum_{k=1}^{\infty} i^k J_k(-i)T_k(x)$. Therefore the matrix exponential of $-iH$ satisfying the normalisation condition $\|H\| \leq 1$ is given by

$$e^{-iH} = J_0(i)\mathbb{1} + 2\sum_{k=1}^{\infty} i^k J_k(-i)T_k(-iH) \quad . \tag{6}$$

Table 1 Matrix Multiplications Needed to Approximate Polynomials of Different Degrees

degree of polynomial m	4	6	8	10	12	14	16	18	20
no. of matrix multiplications									
for Horner scheme	3	5	7	9	11	13	15	17	19
for optim. reordering	2	3	4	5	5	6	6	7	7

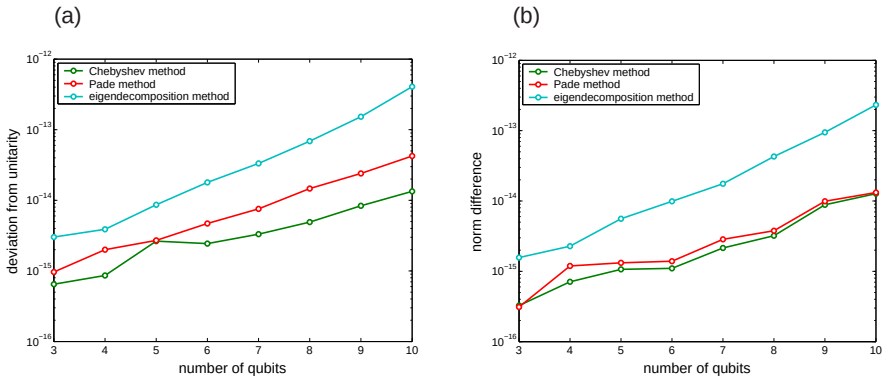

Figure 2 Compariing of the accuracy of different methods subject to the system size: (a) deviation from unitarity measured by $\|\Psi(-iH)^{\dagger}\Psi(-iH)-\mathbb{1}\|$; (b) errors $\|e^{-iH}-\Psi(-iH)\|$ of the different approximation methods Ψ. These results could be obtained, when dealing with Hamiltonians of special properties, which allow to get the exact target exponential.

For dealing with Hamiltonians H of arbitrary finite norm, the 'scaling and squaring' technique is applied. As shown before, the Chebyshev method only requires the computation of a single matrix polynomial per matrix exponential. Since the cost of matrix multiplications supersedes the cost of matrix additions by far, we are interested in reducing the number of required matrix multiplications. When using a conservative method like the Horner scheme, $m-1$ matrix multiplications are required for evaluating a matrix polynomial of degree m. Moreover, as shown in Tab. 1, one can reduce the number of matrix multiplications by a suitable ordering. For instance, the matrix polynomial $\sum_{k=0}^{6}\alpha_k A^k$ can be rewritten as $\sum_{k=0}^{3}\alpha_k A^k + A^3\left(\sum_{k=1}^{3}\alpha_{k+3}A^k\right)$ where only three (instead of five) matrix multiplications are required. In general, only $\mathcal{O}(\sqrt{m})$ nonscalar multiplications are necessary to evaluate a polynomial of degree m. In this sense the partial sum of the Chebyshev series expansion (6) can be evaluated very efficiently [9].

As shown in Fig. 1, the new Chebyshev algorithm for taking matrix exponentials outperforms the standard Padé approximation by a speed-up of 30% in CPU time. Note this acceleration is even more pronounced than the time difference between eigendecomposition and Padé approximation. Moreover, numerical checks illustrate (see Fig. 2) that in cases, where the target matrix exponential is exactly known, the Chebyshev series allows for higher accuracy both in terms of deviation from the target exponential, as well as in terms of deviation from unitarity

1.2 Adapting Parallelisation to the HLRB-II Cluster

In preparative work [2], we implemented two algorithms for multiplying a series of matrices as repeatedly needed in the GRAPE programme package for optimal quan-

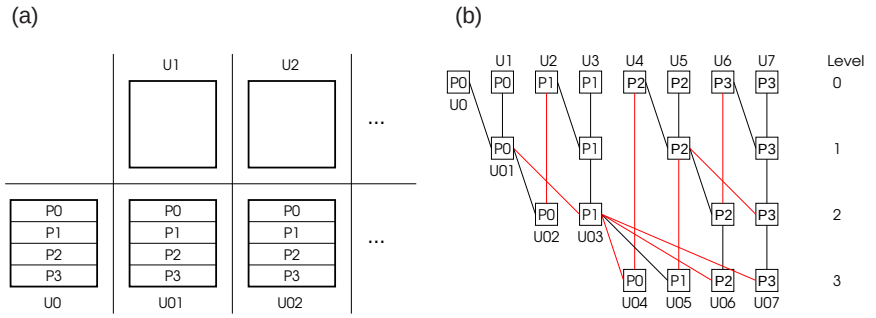

Figure 3 (a) Slice-wise matrix multiplication provides a simple way of parallelisation. U_{0k} denotes the $(k+1)$-fold product $U_k U_{k-1} \cdots U_0$ in the GRAPE-algorithm. The resulting complexity is $\mathcal{O}(M \cdot N^3/p)$. Communication between the processors P is needed solely for broadcasting the matrices U_k prior to propagation. (b) Scheme for tree-like propagation. In this example, propagation is carried out in three steps. Red lines indicate communication between different processors.

tum control. The algorithms differ in run-time and, most importantly, in memory demand. Moreover, the memory redistribution was optimised and tailored to the specific needs of the GRAPE algorithm.

1.2.1 Slice-Wise Propagation

The matrix matrix multiplication AB, with $A, B \in \mathrm{Mat}_{N\times N}(\mathbb{C})$ can most easily be split into jobs distributed to different CPUs by taking say the rows a_ℓ of A separately as

$$AB = (a_1; a_2; \ldots a_N)B = (a_1 B; a_2 B; \ldots; a_N B) \quad . \tag{7}$$

This scheme is readily extendible to k out of the total of M matrices multiplied in the GRAPE-algorithm (see Fig. 3(a)). However, each processor then refers to $k-1$ matrices, which means that they have to be broadcasted. Also, the workspace required by each processor is of the order of $\mathcal{O}(M \cdot N^2)$. The time complexity in this straightforward scheme can easily be evaluated, because the total number of operations is evenly distributed among the available processors. So the order of operations is $\mathcal{O}(M \cdot N^3/p)$, where p is the number of processors.

1.2.2 Tree-Like Propagation

A different approach for computing the propagation is the parallel prefix algorithm [10] depicted in Fig. 3(b). In an extension to previous work [2], it is now applicable to arbitrary combinations of number of processors p and number of matrices M. In contrast to slice-wise propagation, parallel prefix requires communication during the propagation (red lines in Fig. 3(b)): they sum up to $\sum_{l=2}^{\log_2 M} [Broadcast(N^3, p =$

Table 2 Contributions of Parallelised Matrix Operations to Overall Speed-up.

Subroutine	Fraction of CPU Time		Weighted
	with 1 CPU	with 128 CPUs	Speed-up
maxStepSize	0.9	0.713	521
getGradient	0.091	0.287	52.6
expm	0.075	0.049	43.0
propagation	0.01	0.194	6.0
gradient	0.006	0.044	3.5
optimiseCG	1	1	576

$2^{l-1}) + (l-1) \cdot Send(N^3)]$, provided the times for *Broadcast* and *Send* are not influenced by other ongoing communication. Recalling the computation time of $\mathcal{O}(M \cdot N^3/p)$ for the slice-wise propagation, parallel prefix should never be faster (neglecting effects like memory prefetching). On the other hand, parallel prefix does not require all the matrices U_k in all processes, which eliminates the broadcast time prior to the propagation step. It is this advantage that is large enough to outweigh the slower propagation time. Even more important is the reduced memory demand. In our current implementation the maximum number of matrices stored at a single process is $\mathcal{O}(\log_2 M)$ [P_0 produces one result in every level], which is already much less than the $\mathcal{O}(M)$ of the slice-wise propagation.

As summarised in Tab. 2, for a 10-qubit QFT, the parallelisation techniques in combination with other algorithmic improvements result in a speed-up by more than a factor of 576 on 128 processors of the HLRB-II (PHASE 1) as compared to using one node of the same computer. To further improve the efficiency of the implementation, a hybrid method comprising slice-wise and tree-like propagation is being developed. The data transfer will be optimised by combining blockwise computation and hidden communication in a cache-oblivious way.

2 Application: Developing a Quantum CISC Compiler

Richard Feynman's seminal conjecture of using experimentally controllable quantum systems to perform computational tasks [11, 12] roots in reducing the complexity of the problem when moving from a classical setting to a quantum setting. The most prominent pioneering example being Shor's quantum algorithm of prime factorisation [13, 14] which is of polynomial complexity (BQP) on quantum devices instead of showing non-polynomial complexity on classical ones [15]. It is an example of a class of quantum algorithms [16, 17] that solve *hidden subgroup problems* in an efficient way [18], where in the Abelian case, the speed-up hinges on the quantum Fourier transform (QFT). Whereas the network complexity of the fast Fourier transform (FFT) for n classical bits is of order $\mathcal{O}(n2^n)$ [19, 20], the QFT for n qubits

shows a complexity of order $\mathcal{O}(n^2)$. Moreover, Feynman's second observation that quantum systems may be used to efficiently predict the behaviour of other quantum systems has inaugurated a research branch of Hamiltonian simulation [21–26].

Define by $U(\tau) := e^{-it_M H_M} \cdots e^{-it_k H_k} \ldots e^{-it_1 H_1}$ the propagator brought about by a sequence of evolutions of the quantum system under M piece-wise constant Hamiltonians H_k. Then the compilation task is to approximate some U_{target} by $U(\tau)$ (i) with high fidelity and (ii) with least dissipative loss, e.g., with minimal $\tau := \sum_k t_k$.

For implementing a quantum algorithm in an experimental setup, local operations and universal two-qubit quantum gates are required as a minimal set ensuring every unitary module can be realised [27]. More recently, it turned out that generic qubit and qudit pair interaction Hamiltonians suffice to complement local actions to universal controls [28, 29]. Common sets of quantum computational instructions comprise (i) local operations such as the Hadamard gate, the phase gate and (ii) the entangling operations CNOT, controlled-phase gates, $\sqrt{\text{SWAP}}$, i SWAP as well as (iii) the SWAP operation. The number of elementary gates required for implementing a quantum module then gives the network or gate complexity. However, gate complexity often translates into too coarse an estimate for the actual time required to implement a quantum module (see e.g. [30–32]), in particular, if the time scales of a specific experimental setting have to be matched. Instead, effort has been taken to give upper bounds on the actual time complexity [33], e.g., by way of numerical optimal control [3].

Interestingly, in terms of quantum control theory, the *existence of universal gates* is equivalent to the statement that the quantum system is *fully controllable* as has first been pointed out by Ramakrishna and Rabitz [34]. This is, e.g. the case in systems of n spin-$\frac{1}{2}$ qubits that form Ising-type weak-coupling topologies described by arbitrary connected graphs [35–37]. Therefore the usual approach to quantum compilation in terms of local plus universal two-qubit operations [38–42] lends itself to be complemented by optimal-control based direct compilation into machine code: it may be seen as a technology-dependent optimiser in the sense of Ref. [41], however, tailored to deal with more complex instruction sets than the usual local plus two-qubit building blocks. Not only is it adapted to the specific experimental setting, it also allows to fight decoherence by either being near timeoptimal or by exploiting decoherence-protected subspaces [43]. Devising quantum compilation methods for optimised realisations of given quantum algorithms by admissible controls is therefore an issue of considerable practical interest. Here it is the goal to show how quantum compilation can favourably be accomplished by optimal control: the building blocks for gate synthesis will be extended from the usual set of restricted local plus universal two-qubit gates to a larger toolbox of *scalable* multi-qubit gates tailored to yield high fidelity in short time given concrete experimental settings.

Organisation of the Application Section

Following Ref. [44], the purpose of this section is to show that optimal control theory can be put to good use for devising multi-qubit building blocks designed for

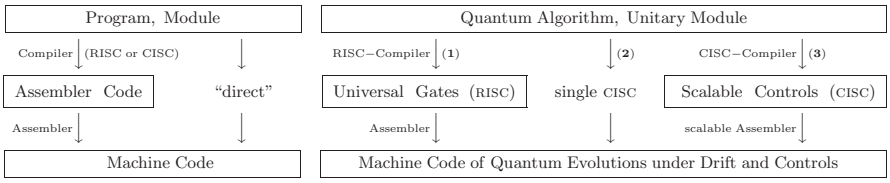

Figure 4 Compilation in classical computation (left) and quantum computation (right). Quantum machine code has to be time-optimal or protected against dissipation, otherwise decoherence wipes out the coherent superpositions. A quantum RISC-compiler (1) by universal gates leads to unnecessarily long machine code. Direct CISC-compilation into a single pulse sequence (2) exploits quantum control for a near time-optimal quantum machine code. Its classical complexity is NP, so direct compilation by numerical optimal control resorting to a classical computer is unfeasible for large quantum systems. The third way (3) promoted here pushes quantum CISC-compilation to the limits of classical supercomputer clusters and then assembles the multi-qubit complex instructions sets recursively into time-optimised or dissipation-protected quantum machine code.

scalable quantum computing in realistic settings. Note these building blocks are no longer meant to be universal *in the practical sense* that any arbitrary quantum module should be built from them (plus local controls). Rather they provide specialised sets of complex instructions tailored for breaking down typical tasks in quantum computation with substantial speed gains compared to the standard compilation by decomposition into one-qubit and two-qubit gates. Thus a CISC quantum compiler translates into significant progress towards the quantum error-correction threshold.

For demonstrating scalable quantum compilation, we choose systems with linear coupling topology, i.e., qubit chains coupled by nearest-neighbour Ising interactions. The section is then organised as follows: CISC quantum compilation by optimal control will be illustrated in three different, yet typical examples

(1) the indirect $1,n$-SWAP gate,
(2) the quantum Fourier transform (QFT) ,
(3) the generalisation of the CNOT gate to multiply-controlled NOT gates, C^{n}NOT.

For every instance of n-qubit systems, we analyse the effects of (i) sacrificing universality by going to special instruction sets tailored to the problem, (ii) extending pair interaction gates to effective multi-qubit interaction gates s, and (iii) we compare the time gain by recursive m-qubit CISC-compilation ($m \leq n$) to the two limiting cases of the standard RISC-approach ($m = 2$) on one hand and the (extrapolated) time-complexity inferred from single-CISC compliation (with $m = n$).

2.1 Quantum Compilation as an Optimal Control Task

As shown in Fig. 4, the quantum compilation task can be addressed following different principle guidelines: **(1)** by the standard decomposition into local operations and

universal two-qubit gates, which by analogy to classical computation was termed *reduced instruction set* quantum computation (RISC) [45] or **(2)** by using direct compilation into one single *complex instruction set* (CISC) [45]. The existence of a such a single effective gate is guaranteed simply by the unitaries forming a group: a sequence of local plus universal gates is a product of unitaries and thus a single unitary itself.

As a consequence, CISC quantum compilation lends itself to be treated by numerical optimal control. One thus resorts to clusters of classical computers for translating the unitary target module directly into the 'machine code' of evolutions of the quantum system under combinations of the drift Hamiltonian H_0 and experimentally available controls H_j.

In a number of studies on quantum systems up to 10 qubits, we have shown that direct compilation by gradient-assisted optimal control [3, 4, 46] allows for substantial speed-ups, e.g., by a factor of 5 for a CNOT and a factor of 13 for a Toffoli-gate on coupled Josephson qubits [46]. However, the direct approach naturally faces the limits of computing quantum systems on classical devices: upon parallelising our C++ code for high-performance clusters [2], we found that extending the quantum system by one qubit increases the CPU-time required for direct compilation into the quantum machine code of controls by grossly a factor of eight. So the classical complexity for optimal-control based quantum compilation is NP.

Therefore, here we advocate a third approach **(3)** that uses direct compilation into units of multi-qubit complex instruction sets up to the CPU-time limits of optimal quantum control on classical computers: these building blocks are to be taken as fundamental units designed such as to allow for recursive scalable quantum compilation in large quantum systems (i.e. those beyond classical computability).

Time Standards

When comparing times to implement unitary target gates by the RISC *vs* the CISC approach, we will assume for simplicity that local unitary operations are 'infinitely' fast compared to the duration of the Ising coupling evolution scaled by the coupling constant J_{ZZ} so that the total gate time is solely determined by the coupling evolutions unless stated otherwise. Let us emphasise, however, this stipulation only concerns the time standards. The optimal-control assisted CISC-compilation methods presented here are in no way limited to fast local controls. Also the assembler step of concatenating the CISC-building blocks is independent of the ratio of times for local operations *vs* coupling interactions.

Error Propagation and Relaxative Losses

As the main figure of merit we refer to a quality function q resulting from the fidelity F_{tr} and the dissipative decay with overall relaxation rate constant T_R during a duration τ

$$q = F_{\rm tr}\, e^{-\tau/T_R} \tag{8}$$

assuming independence. Moreover, for n qubits one defines as the trace fidelity of an experimental unitary module $U_{\rm exp}$ with respect to the target gate $V_{\rm target}$ (thus $U,V \in U(N)$ with $N := 2^n$)

$$\begin{aligned} F_{tr} :=&\ \tfrac{1}{N}\ \mathrm{Re}\ \mathrm{tr}\{V_{\rm target}^{\dagger} U_{\rm exp}\} \\ =&\ 1 - \tfrac{1}{2N}\ \|V_{\rm target} - U_{\rm exp}\|_2^2 \quad , \end{aligned} \tag{9}$$

which follows via the simple relation to the Euclidean distance

$$\begin{aligned} \|V-U\|_2^2 &= \|U\|_2^2 + \|V\|_2^2 - 2\,\mathrm{Re}\,\mathrm{tr}\{V^{\dagger}U\} \\ &= 2N - 2N\tfrac{1}{N}\,\mathrm{Re}\,\mathrm{tr}\{V^{\dagger}U\} \\ &= 2N(1-F_{\rm tr}) \quad , \end{aligned}$$

the latter two identities invoking unitarity.

Consider a multiqubit-interaction module (CISC) with quality $q_m = F_m\, e^{-\tau_m/T_m}$ that decomposes into r universal two-qubit gates (RISC), out of which $r' \leq r$ gates have to be performed *sequentially*. Moreover, each 2-qubit gate shall be carried out with the uniform quality $q_2 = F_2\, e^{-\tau_2/T_2}$. Henceforth we assume for simplicity equal relaxation rate constants, so $T_2 = T_m$ are identified with T_R. Then, as a first useful rule of the thumb, it is advantageous to compile the multiqubit module directly if $F_m > (F_2)^r$, or more precisely taking relaxation into account, if the module can be realised with a fidelity

$$F_m > (F_2)^r\, e^{-(r'\cdot\tau_2 - \tau_m)/T_R} \quad . \tag{10}$$

2.2 $1,n$ *SWAP*

The easiest and most basic example to illustrate the pertinent effects of optimal-control based CISC-quantum compilation is the respective indirect $1,n$-SWAP gates in spin chains of n qubits coupled by nearest-neighbour Ising interactions with J_{ZZ} denoting the coupling constant.

For the $1,2$-SWAP unit there is a standard textbook decomposition into three CNOTs. Thus for Ising-coupled systems and in the limit of fast local controls, the total time required for an $1,2$-SWAP is $3/(2J_{ZZ})$, and there is no faster implementation [3, 47–49]. Note, however, that in systems coupled by the isotropic Heisenberg interaction XXX, the $1,2$-SWAP may be directly implemented just by letting the system evolve for a time of only $1/(2J_{XXX})$. Sacrificing universality, it may thus be advantageous to regard the $1,2$-SWAP as basic unit for the $1,n$-SWAP task rather than the universal CNOT. Note, however, any $1,n$-SWAP can be built from $1,2$-SWAPs: following the most obvious scheme of Fig. 5(a) shows how to decompose an even-

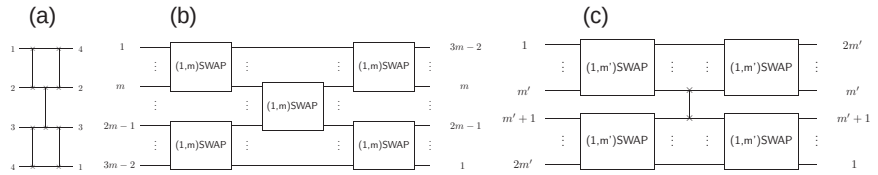

Figure 5 Assembling the $1,n$-SWAP gate from different building blocks: (a) by recursive use of $1,2$-SWAPs, (b) by three symmetric blocks of $1,m$-SWAPs or (c) by two blocks of $1,m'$-SWAPs and an interior $1,2$-SWAP.

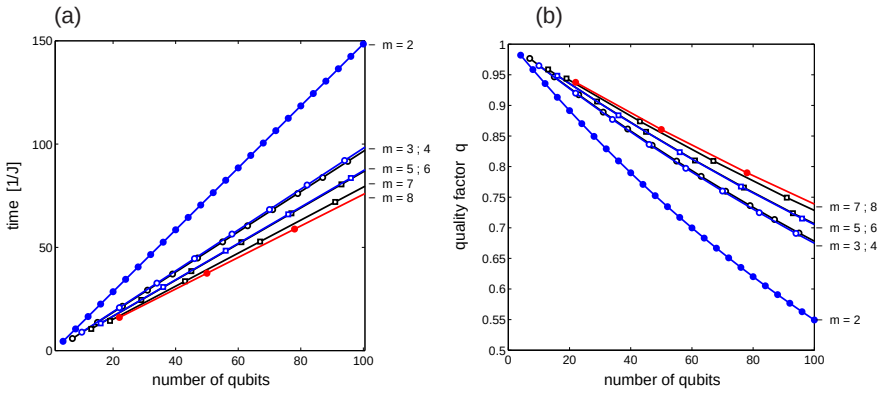

Figure 6 Left: Times required for indirect $1,n$-SWAPs on linear chains of n Ising-coupled qubits by assembling $1,m$-SWAP building blocks reaching from $m = 2$ (RISC) up to $m = 8$ (CISC). The time units are expressed as $1/J_{ZZ}$ assuming the duration of local operations can be neglected compared to coupling evolutions (details in the text). Right: Translation of the effective gate times into over-all quality figures $q = (q_m)^{r_m}$ for an effective gate assembled from r_m components of single qualities $q_m := F_m\, e^{-\tau_m/T_R}$ (with the respective component fidelities homogeneously falling into a narrow interval $F_m \in [0.99994, 0.99999]$ for $m = 3,\ldots,8$). Data are shown for a uniform relaxation rate constant of $1/T_R = 0.004 J_{ZZ}$.

order $1,2n$-SWAP into units of $1,2$-SWAPs. For the odd-order counterpart, i.e., the $1,(2n-1)$-SWAP, just omit the qubit number $2n$ and all the building blocks connected to it. Moreover, the generalisation to decomposing a $1,(3m-2)$-SWAP into three $1,m$-SWAP building blocks is also immediate, e.g., setting $m = 2$ in Fig. 5(b) reproduces part (a).

Now, the $1,m$-SWAP building blocks themselves can be precompiled into time-optimised single complex instruction sets by exploiting the GRAPE-algorithm of optimal control up to the current limits of m imposed by CPU-time allowance.

Proceeding in the next step to large n, Fig. 6 underscores how the time required for $1,n$-SWAPs decreases significantly by assembling precompiled $1,m$-SWAP building blocks as CISC units recursively up to a multi-qubit interaction size of $m = 8$, where the speed-up is by a factor of more than 1.96. Clearly, such a set of $1,m$-SWAP building blocks with $m \in \{2,3,4,5,6,7,8\}$ allows for efficiently synthesising any $1,n$-SWAP.

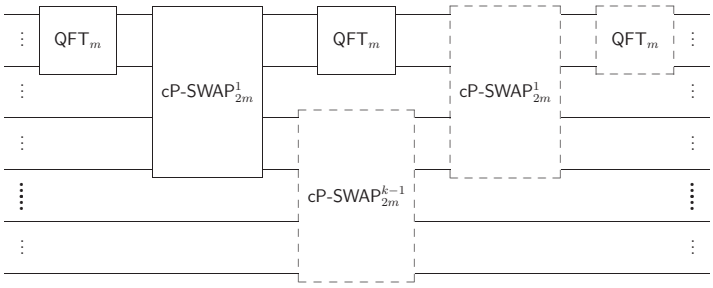

Figure 7 For $k \geq 2$, a (km)-qubit QFT can be assembled from k times an m-qubit QFT and $\binom{k}{2}$ instances of $2m$-qubit modules cP-SWAP$^j_{2m}$, where the index j of different phase-rotation angles takes the values $j = 1,2,\ldots,k-1$. The dashed boxes show the induction $k \mapsto k+1$.

However, deducing from Fig. 6 that the time complexity of $1,n$-SWAPs is linear is premature: although the slopes seem to converge to a non-zero limit, numerical optimal control may become systematically inefficient for larger interaction sizes m. Therefore, on the current basis of findings, a logarithmic time complexity cannot ultimately be excluded.

Summarising the results for the indirect SWAPs in terms of the three criteria described in the introduction, we have the following: (i) in Ising coupled qubit chains, there is no speed-up by changing the basic unit from the universal CNOT into a $1,2$-SWAP, whereas in isotropically coupled systems the speed-up amounts to a factor of three; (ii) extending the building blocks of $1,m$-SWAPs from $m = 2$ (RISC) to $m = 8$ (CISC) gives a speed-up by a factor of nearly two (1.96) even under Ising-type couplings; (iii) the numerical data are consistent with a time complexity converging to a linear limit for the $1,n$-SWAP task in Ising chains, however, there is no proof for this yet.

2.3 Quantum Fourier Transform (QFT)

Since many quantum algorithms take their advantage by efficiently solving some hidden subgroup problem, the quantum Fourier transform plays a central role.

In order to realise a QFT on large qubit systems, our approach is the following: given an m-qubit QFT, we show that for obtaining a $(k \cdot m)$-qubit QFT by recursively using m-qubit building blocks, a second type of module is required, to wit a combination of controlled phase gates and SWAPs, which henceforth we dub m-qubit cP-SWAP for short.

One arrives at the desired block decomposition of a general $(k \cdot m)$-qubit QFT as shown in Fig. 7: it requires k times the same m-qubit QFT interdispersed with $\binom{k}{2}$ times an $2m$-qubit cP-SWAP, out of which $k-1$ show different phase-rotation angles. For all m and $j = 1,2,\ldots,(k-1)$, one finds (i) a cP-SWAP$^j_{2m}$ takes as least as

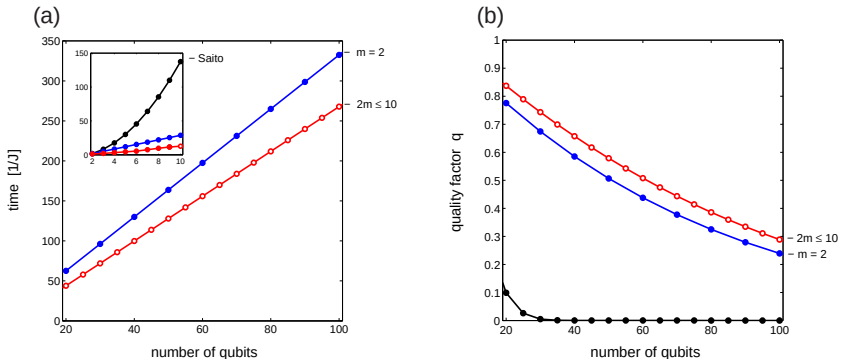

Figure 8 Comparison of CISC-compiled QFT (red) with standard RISC compilations following the scheme by Saito [50] (black) or Blais [51] (blue): (a) times for implementation translate into quality factors (b) for a relaxation rate constant of $1/T_R = 0.004 J_{ZZ}$.

long as a QFT_m ; (ii) a QFT_m takes as least as long as a $\mathrm{cP\text{-}SWAP}_m^j$; (iii) a $\mathrm{cP\text{-}SWAP}_m^j$ takes least as long as a $\mathrm{cP\text{-}SWAP}_m^{j+1}$. Thus the duration of a $(k \cdot m)$-qubit QFT built from m-qubit and $2m$-qubit modules amounts to

$$\tau_{k\cdot m}^{\mathrm{QFT}} = 2\cdot\tau(\mathrm{QFT}_m) + (k-1)\cdot\tau(\mathrm{cP\text{-}SWAP}_{2m}^1) + (k-2)\cdot\tau(\mathrm{cP\text{-}SWAP}_{2m}^2) \quad . \tag{11}$$

In the following, we consider the overall quality of a $(k \cdot m)$-qubit QFT in terms of its two types of building blocks, namely the basic m-qubit QFT as well as the constituent $2m$-qubit cP-SWAPs with their respective different rotation angles. We will neglect rotations as soon as their angle falls short of a threshold of $\pi/2^{10}$. This approximation is safe since it is based on a calculation of a 20-qubit QFT, where the truncation does not introduce any relative error beyond 10^{-5}. Following the block decomposition of Fig. 7, thus three variants of cP-SWAPs are left, since all $\mathrm{cP\text{-}SWAP}_{10}^j$ elements with $j \geq 3$ boil down to mere SWAP gates due to truncation of small rotation angles.

With these stipulations, the task of assembling an $(k \cdot 10)$-qubit QFT translates into using 10-qubit cP-SWAP building blocks ($2m = 10$) and the 5-qubit QFT ($m = 5$) in the sense of a $(2k \cdot 5)$-qubit QFT. Its duration $\tau(\mathrm{QFT}_{2k\cdot 5})$ is then readily obtained as in Eqn. 11 thus giving an overall quality of

$$\begin{aligned} q^{\mathrm{QFT}_{2k\cdot 5}} &= \\ &= (F_{\mathrm{tr}}^{\mathrm{QFT}_5})^{2k}\,(F_{\mathrm{tr}}^{\mathrm{cP\text{-}SWAP}_{10}^1})^{2k-1}\,(F_{\mathrm{tr}}^{\mathrm{cP\text{-}SWAP}_{10}^2})^{2k-2}\,(F_{\mathrm{tr}}^{\mathrm{cP\text{-}SWAP}_{10}^3})^{\binom{2k}{2}-4k+3}\,e^{-\tau_{2k\cdot 5}^{\mathrm{QFT}}/T_R}\,. \end{aligned} \tag{12}$$

Based on this relation, Fig. 8 shows the numerical results of the current calculations on the HLRB-II cluster, where the quality of a CISC-compiled $(k \cdot 10)$-qubit QFT is notably superior to the standard RISC versions [50, 51].

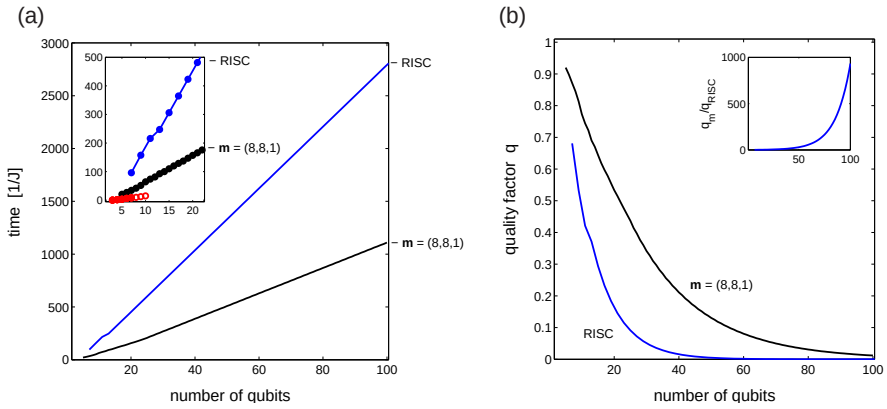

Figure 9 Comparison of implementations of the C^{n}NOT on linear Ising-coupled spin chains using 2-, 3- and up to 6-qubit building blocks in terms of time (a) and quality factors (b) under relaxation with a rate constant of $1/T_R = 0.004 J_{ZZ}$.

2.4 Multiply-Controlled NOT Gate (C^{n}NOT)

Multiply-controlled CNOT gates generalise Toffoli's gate, which is C^{2}NOT to C^{n}NOT. They frequently occur in error-correction schemes hence their practical relevance.

Here we address the task of decomposing a C^{n}NOT into C^{m}NOTs and $1, m$ SWAP gates given the topology of a linear chain of $n+2$ qubits coupled by nearest-neighbour Ising interactions. The reason for $n+2$ qubits being an ancilla qubit that turns the problem to linear complexity [52]. As described elsewhere [44], there is an analogous, yet more elaborate induction to prove recursive scalable assembling schemes. Here, we just present the results. Fig. 9 convincingly demonstrates that CISC-compilation to 10-qubit building blocks is a significant time saver thus translating into superior over-all qualities under realistic conditions. These results are of great practical importance, since the C^{n}NOT gates are a cornerstone in quantum error correction.

3 Conclusion

By numerical developments using, e.g., Chebychev expansions for a fast matrix exponential and Strassen-type matrix multiplication techniques, a fully parallelised programme package with parallel matrix operations on clusters has been provided. Quantum control theory is a powerful framework for devising algorithms to steer quantum devices with optimal figures of merit. Controlling quantum systems experimentally is central to many branches of quantum technology including nanotechnology, quantum information processing and spectroscopy. However, to find

such steerings is a–classically–computationally demanding task, as the resource requirements grow exponentially with the size of the quantum system. We have exploited recent progress allowing to use high-end parallel clusters. Building upon these achievements, the HLRB-II cluster has been used for obtaining progress in the following fields:

1. an optimal-control based quantum CISC compiler that recursively uses medium-sized modules for addressing quantum systems of dimensions that are too large to be handled otherwise;
2. a CISC compiler allows for assembling optimal quantum controls protected against decoherence [43].

The cutting-edge applications of a quantum CISC compiler are based on parallel matrix operations for clusters. They pave the way to another frontier of research: optimising the quantum assembler task on the extended toolbox of quantum CISC-modules with effective many-qubit interactions. It is anticipated [53] that methods developed in classical computer science, e.g., for fast Fourier transforms [19, 20, 54], can also be put to good use for systematically optimising quantum assemblers.

Acknowledgements

This work was supported in part by the integrated EU project QAP and by *Deutsche Forschungsgemeinschaft*, DFG, within the incentive SFB-631. Via project h1051 access to the high-performance parallel cluster HLRB-II at *Leibniz Rechenzentrum* of the Bavarian Academy of Science is gratefully acknowledged.

References

[1] J. Dowling and G. Milburn, Phil. Trans. R. Soc. Lond. A **361**, 1655 (2003).
[2] T. Gradl, A. K. Spörl, T. Huckle, S. J. Glaser, and T. Schulte-Herbrüggen, Lect. Notes Comput. Sci. **4128**, 751 (2006), Proceedings of the EURO-PAR 2006.
[3] T. Schulte-Herbrüggen, A. K. Spörl, N. Khaneja, and S. J. Glaser, Phys. Rev. A **72**, 042331 (2005).
[4] N. Khaneja, T. Reiss, C. Kehlet, T. Schulte-Herbrüggen, and S. J. Glaser, J. Magn. Reson. **172**, 296 (2005).
[5] C. Moler and C. van Loan, SIAM Rev. **20**, 801 (1978).
[6] C. Moler and C. van Loan, SIAM Rev. **45**, 3 (2003).
[7] T. J. Rivlin, *The Chebyshev Polynomials* (Wiley-Interscience, New York, 1974).
[8] M. Veshtort and R. G. Griffin, J. Magn. Reson. **178**, 248 (2006).
[9] M. S. Paterson and L. J. Stockmeyer, SIAM J. Comp. **2**, 60 (1973).
[10] R. E. Ladner and M. J. Fischer, J. ACM **27**, 831 (1980).
[11] R. P. Feynman, Int. J. Theo. Phys. **21**, 467 (1982).
[12] R. P. Feynman, *Feynman Lectures on Computation* (Perseus Books, Reading, MA., 1996).
[13] P. W. Shor, in *Proceedings of the Symposium on the Foundations of Computer Science, 1994, Los Alamitos, California* (IEEE Computer Society Press, New York, 1994), pp. 124–134.

[14] P. W. Shor, SIAM J. Comput. **26**, 1484 (1997).
[15] C. H. Papadimitriou, *Computational Complexity* (Addison Wesley, Reading, MA., 1995).
[16] R. Jozsa, Proc. R. Soc. A. **454**, 323 (1998).
[17] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca, Proc. R. Soc. A. **454**, 339 (1998).
[18] M. Ettinger, P. Høyer, and E. Knill, Inf. Process. Lett. **91**, 43 (2004).
[19] J. W. Cooley and J. W. Tukey, Math. Comput. **19**, 297 (1965).
[20] T. Beth, *Verfahren der schnellen Fourier-Transformation* (Teubner, Stuttgart, 1984).
[21] S. Lloyd, Science **273**, 1073 (1996).
[22] D. Abrams and S. Lloyd, Phys. Rev. Lett. **79**, 2586 (1997).
[23] C. Zalka, Proc. R. Soc. London A **454**, 313 (1998).
[24] C. Bennett, I. Cirac, M. Leifer, D. Leung, N. Linden, S. Popescu, and G. Vidal, Phys. Rev. A **66**, 012305 (2002).
[25] L. Masanes, G. Vidal, and J. Latorre, Quant. Inf. Comput. **2**, 285 (2002).
[26] E. Jané, G. Vidal, W. Dür, P. Zoller, and J. Cirac, Quant. Inf. Computation **3**, 15 (2003).
[27] D. Deutsch, Proc. Royal Soc. London A **400**, 97 (1985).
[28] J. Dodd, M. Nielsen, M. Bremner, and R. Thew, Phys. Rev. A **65**, 040301(R) (2002).
[29] M. Bremner, D. Bacon, and M. Nielsen, Phys. Rev. A **71**, 052312 (2005).
[30] G. Vidal, K. Hammerer, and J. I. Cirac, Phys. Rev. Lett. **88**, 237902 (2002).
[31] A. M. Childs, H. L. Haselgrove, and M. A. Nielsen, Phys. Rev. A **68**, 052311 (2003).
[32] R. Zeier, M. Grassl, and T. Beth, Phys. Rev. A **70**, 032319 (2004).
[33] P. Wocjan, D. Janzing, and T. Beth, Quant. Inf. Comput. **2**, 117 (2002).
[34] V. Ramakrishna and H. Rabitz, Phys. Rev. A **54**, 1715 (1995).
[35] T. Schulte-Herbrüggen, *Aspects and Prospects of High-Resolution NMR* (PhD Thesis, Diss-ETH 12752, Zürich, 1998).
[36] S. J. Glaser, T. Schulte-Herbrüggen, M. Sieveking, O. Schedletzky, N. C. Nielsen, O. W. Sørensen, and C. Griesinger, Science **280**, 421 (1998).
[37] U. Helmke, K. Hüper, J. B. Moore, and T. Schulte-Herbrüggen, J. Global Optim. **23**, 283 (2002).
[38] R. R. Tucci (1999), e-print: http://arxiv.org/pdf/quant-ph/9902062.
[39] D. Williams, *Quantum Computer Architecture, Assembly Language and Compilation* (Master's Thesis, University of Warwick, 2004).
[40] V. V. Shende, S. Bullock, and I. L. Markov, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **25**, 1000 (2006).
[41] K. M. Svore, A. V. Aho, A. W. Cross, I. Chuang, and I. L. Markov, Computer **25**, 74 (2006).
[42] R. R. Tucci (2007), e-print: http://arxiv.org/0706.0479.
[43] T. Schulte-Herbrüggen, A. Spörl, N. Khaneja, and S. J. Glaser (2006), e-print: http://arxiv.org/pdf/quant-ph/0609037.
[44] T. Schulte-Herbrüggen, A. Spörl, and S. J. Glaser (2007), e-print: http://arxiv.org/pdf/quant-ph/0712.3227.
[45] G. D. Sanders, K. W. Kim, and W. C. Holton, Phys. Rev. A **59**, 1098 (1999).
[46] A. K. Spörl, T. Schulte-Herbrüggen, S. J. Glaser, V. Bergholm, M. J. Storcz, J. Ferber, and F. K. Wilhelm, Phys. Rev. A **75**, 012302 (2007).
[47] N. Khaneja and S. J. Glaser, Chem. Phys. **267**, 11 (2001).
[48] N. Khaneja, R. Brockett, and S. J. Glaser, Phys. Rev. A **63**, 032308 (2001).
[49] N. Khaneja, S. J. Glaser, and R. Brockett, Phys. Rev. A **65**, 032301 (2002).
[50] A. Saito, K. Kioi, Y. Akagi, N. Hashizume, and K. Ohta (2000), http://arxiv.org/pdf/quant-ph/0001113.
[51] A. Blais, Phys. Rev. A **64**, 022312 (2001).
[52] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. W. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, Phys. Rev. A **52**, 3457 (1995).
[53] R. Zeier, personal communication (2007).
[54] M. Clausen and U. Baum, *Fast Fourier Transforms* (Bibliographisches Institut, Mannheim, 1993).