Quantum Inf Process (2017) 16:152
DOI 10.1007/s11128-017-1603-1

Quantum arithmetic with the quantum Fourier transform

Lidia Ruiz-Perez[1] · **Juan Carlos Garcia-Escartin**[1]

Received: 25 November 2016 / Accepted: 17 April 2017 / Published online: 28 April 2017

Abstract The quantum Fourier transform offers an interesting way to perform arithmetic operations on a quantum computer. We review existing quantum Fourier transform adders and multipliers and comment some simple variations that extend their capabilities. These modified circuits can perform modular and non-modular arithmetic operations and work with signed integers. Among the operations, we discuss a quantum method to compute the weighted average of a series of inputs in the transform domain. One of the circuits, the controlled weighted sum, can be interpreted as a circuit to compute the inner product of two data vectors.

Keywords Quantum Fourier transform · Quantum adder · Quantum multiplier

1 Introduction: quantum arithmetic

The discovery of Shor's algorithm for efficient quantum factoring [1] awakened an interest on the quantum implementation of the modular arithmetic operations that are the building blocks of the quantum factorization circuit. Since then, there have been many proposals on how to build the required quantum modular adders, multipliers and exponentiators using a set of elementary quantum gates.

The first suggested circuits were reversible versions of known classical implementations [2,3]. Many subsequent proposals have been improvements and modifications of reversible generalizations of the adders and multipliers of classical digital logic [4–15].

✉ Lidia Ruiz-Perez
lruiper@ribera.tel.uva.es

[1] Dpto. de Teoría de la Señal y Comunicaciones, ETSI de Telecomunicación, Universidad de Valladolid, Campus Miguel Delibes, Paseo Belén 15, 47011 Valladolid, Spain

There are also solutions with more of a "quantum flavour" such as teleportation-based operations [16], measurement-based schemes on cluster states [17], repeat-until-success circuits [18] or implementations that restrict to experimentally achievable quantum operations like the nearest-neighbour interaction [19].

A particularly elegant quantum alternative is the quantum Fourier transform, QFT, adder of Draper [20] and its generalizations in a variety of QFT adders and multipliers [21–24].

In this paper, we study those systems and their applications to modular and non-modular arithmetic and discuss a QFT-based circuit to compute weighted sums. The controlled version of this circuit can be used to implement inner products.

Section 2 describes the phase encoding that permits implementing arithmetic operations in the transform domain. Section 3 introduces the basic QFT adder, including modified circuits that compute non-modular additions and work with signed integers, which provides a circuit for subtraction.

Section 4 analyses the implementation of QFT adders for qubits. It takes the general adder circuit and decomposes it into a group of elementary gates acting on a collection of two-level quantum systems (qubits). The implementation rests on QFT gates and controlled rotation gates.

Section 5 proposes a modified QFT adder to compute the arithmetic mean of a list of integers. Section 6 generalizes that circuit and gives a weighted adder that sums the integers of a list after multiplication with constant weights. We also study multiplication by a constant as a particular case of a weighted sum.

Section 7 describes a QFT multiplier for qubits based on the combination of modified QFT adders. The QFT multiplier is adapted in Sect. 8 to put forward a qubit implementation of a programmable weighted adder which works with arbitrary input integers and weights. The presented quantum circuit can be used to implement different inner products between two vectors.

We conclude the paper with a discussion of the applications of these circuits in Sect. 9.

2 The quantum Fourier transform and distributed phase encoding

The quantum Fourier transform, QFT, provides an alternative way to perform arithmetic operations on a quantum computer. Consider a d-dimensional system with states $|x\rangle$ from the computational basis $\{|0\rangle, |1\rangle, \ldots, |d-1\rangle\}$. In this basis, we define the QFT operation as

$$\mathrm{QFT}\,|x\rangle = \frac{1}{\sqrt{d}} \sum_{k=0}^{d-1} \mathrm{e}^{i\frac{2\pi xk}{d}} |k\rangle\,. \tag{1}$$

The QFT allows us to encode a number x in the relative phases of the states of a uniform superposition consisting in the sum of all the states $|k\rangle$ in the computational basis, each with the same $\frac{1}{\sqrt{d}}$ amplitude.

Imagine we want to work with natural numbers from 0 to $d-1$. One possible encoding is mapping number x into state $|x\rangle$. With the QFT, we can take the information into the phases $\mathrm{e}^{i\frac{2\pi xk}{d}} = \omega^{xk}$ that appear together with each state $|k\rangle$ of the

superposition. The QFT can be interpreted as a change of basis. We call $|\phi(x)\rangle$ to the state QFT $|x\rangle$ that encodes x in this new transformed basis.

We can equally define an inverse quantum Fourier transform operator IQFT so that

$$\text{IQFT}\,|k\rangle = \frac{1}{\sqrt{d}}\sum_{x=0}^{d-1} e^{-i\frac{2\pi xk}{d}}\,|x\rangle\,. \tag{2}$$

With the direct and the inverse Fourier transforms, we can move back and forth between the computational basis and the phase representation. In our notation, this conversion from the phase encoding to the computational basis is written as

$$\text{IQFT}\,|\phi(x)\rangle = \text{QFT}^{-1}\text{QFT}\,|x\rangle = |x\rangle\,. \tag{3}$$

This phase encoding is the basic common element of all existing proposals for QFT arithmetic [20–24].

3 QFT adders

QFT addition provides a simple example of how operations with the phase encoding work. Once in the transform domain, we need quantum operators that act on the distributed phases of our quantum states. The basic element in these operators is the controlled phase gate, CZ. We start from the well-known controlled Pauli Z gate which, for two input qubits $|x\rangle$ and $|y\rangle$, gives

$$\text{CZ}\,|x\rangle\,|y\rangle = \mathrm{e}^{i\pi xy}\,|x\rangle\,|y\rangle\,. \tag{4}$$

We can generalize the gate for d-dimensional systems (qudits) so that

$$\text{CZ}\,|x\rangle\,|y\rangle = \mathrm{e}^{i\frac{2\pi xy}{d}}\,|x\rangle\,|y\rangle\,. \tag{5}$$

As it should, when $d = 2$ we recover the qubit gate.

We can also define a modified version of the controlled phase shift gate

$$\text{CZ}^F\,|x\rangle\,|y\rangle = \mathrm{e}^{i\frac{2\pi xy}{Fd}}\,|x\rangle\,|y\rangle \tag{6}$$

that introduces a factor F in the divisor which will be useful later. All the CZ and CZ^F gates we use correspond to controlled rotation gates $\text{CR}(\theta)$ for different rotation angles. These gates are basic building blocks in many quantum arithmetic constructions, and there are multiple proposals for their implementation with different quantum information units [25–28].

These ingredients are enough to give a modulo d adder. We can add two numbers that are originally encoded in the computational basis by taking one of them into phase encoding and then applying a controlled phase shift. The adder comes from the

sequence of operations

$$\mathrm{IQFT}_2 \cdot \mathrm{CZ} \cdot \mathrm{QFT}_2 \, |x\rangle \, |y\rangle = |x\rangle \, |x + y \pmod d\rangle \, . \tag{7}$$

Here and in the following equations, when we apply a quantum gate on only a subset of all the possible input states, we introduce subindices to show on which states the gates are acting. The first operation

$$|x\rangle \, |y\rangle \xrightarrow{\mathrm{QFT}_2} \frac{1}{\sqrt{d}} \sum_{k=0}^{d-1} \mathrm{e}^{i \frac{2\pi y k}{d}} \, |x\rangle \, |k\rangle \tag{8}$$

encodes number y into the phase basis. The phase gate introduces a phase shift that is equivalent to a modulo d addition in that basis, so that

$$\frac{1}{\sqrt{d}} \sum_{k=0}^{d-1} \mathrm{e}^{i \frac{2\pi y k}{d}} \, |x\rangle \, |k\rangle \xrightarrow{\mathrm{CZ}} \frac{1}{\sqrt{d}} \sum_{k=0}^{d-1} \mathrm{e}^{i \frac{2\pi y k}{d}} \mathrm{e}^{i \frac{2\pi x k}{d}} \, |x\rangle \, |k\rangle \, . \tag{9}$$

Finally, the inverse QFT takes the result back into the computational basis with

$$\begin{aligned} &\frac{1}{\sqrt{d}} \sum_{k=0}^{d-1} \mathrm{e}^{i \frac{2\pi (x+y) k}{d}} \, |x\rangle \, |k\rangle \xrightarrow{\mathrm{IQFT}_2} \frac{1}{d} \sum_{k,l=0}^{d-1} \mathrm{e}^{i \frac{2\pi (x+y) k}{d}} \mathrm{e}^{-i \frac{2\pi k l}{d}} \, |x\rangle \, |l\rangle \\ &= |x\rangle \, |x + y \pmod d\rangle \, . \end{aligned} \tag{10}$$

The adder can be extended to any number of inputs. Imagine we have N integers $x_1, x_2, \ldots, x_N$ encoded into the state $|x_1\rangle \, |x_2\rangle \ldots |x_N\rangle$. Then we can repeat the sum in Eq. 7 with the operation

$$\mathrm{IQFT}_N \cdot \mathrm{CZ}_{1,N} \ldots \mathrm{CZ}_{N-2,N} \ldots \mathrm{CZ}_{N-1,N} \cdot \mathrm{QFT}_N \, |x_1\rangle \, |x_2\rangle \ldots |x_{N-1}\rangle \, |x_N\rangle \tag{11}$$

that produces an output state

$$|x_1\rangle \, |x_2\rangle \ldots |x_{N-1}\rangle \, |x_1 + x_2 + \ldots + x_N \pmod d\rangle \, . \tag{12}$$

Here, the subindices in $\mathrm{CZ}_{c,t}$ give the indices of the control state, c, and of the target state, t. Each controlled phase shift adds an integer in the phase encoding. This operation uses the minimum possible number of qudits, but it can also be interesting to preserve all the input states and store their sum in an ancillary qudit. In that case, we can apply the procedure to an initial state $|x_1\rangle \, |x_2\rangle \ldots |x_N\rangle \, |0\rangle$. The result is the sum of the N integers plus 0, which gives the same result as in the compact version.

There are a few additional modifications worth noticing. First, if we want to perform arithmetic, non-modular, additions instead of modulo d addition, we can always encode the data in a system of a larger dimension d' where modulo d' addition and regular arithmetic addition are the same for our range of values. For instance, for two integers x and y between 0 and $d-1$, the sum will always stay between 0 and $2d-2$ and a system

of dimension $d' = 2d-1$ will suffice. We can take an input state $|x\rangle_d |y\rangle_d |0\rangle_{2d-1}$ with systems of dimensions d, d and $2d-1$, respectively, and use the QFT for $d' = 2d-1$ and CZ operations that can also be defined for inputs of a different size as

$$\mathrm{CZ}\,|x\rangle_d |y\rangle_{2d-1} = \mathrm{e}^{i\frac{2\pi xy}{2d-1}} |x\rangle_d |y\rangle_{2d-1}\,. \tag{13}$$

Similarly, if we sum N numbers, arithmetic addition requires a system with dimension $d' = Nd - N + 1$ and we need to adapt the QFT and CZ circuits to this new dimension.

These circuits can also perform signed addition for numbers up to $d/2$. We just need to encode positive numbers $x < d/2$ into states $|x\rangle$ and negative numbers $-x$ into states $|d-x\rangle$, in both cases in the computational basis. After the QFT, positive numbers are associated with phases $\mathrm{e}^{i\frac{2\pi x}{d}}$ below π, which correspond to a phase $\mathrm{e}^{i\frac{2\pi xk}{d}}$ accompanying each state $|k\rangle$, and negative numbers are associated with negative phases (equivalent to phases above π for $k = 1$). The QFT adder will then perform signed addition, which gives an implementation for subtraction.

4 QFT adder: qubit implementation

The most common implementations of quantum logic use two-dimensional systems (qubits). In this section, we consider the addition of numbers encoded in n qubits.

We first describe a variant on Draper's QFT adder [20] to compute full arithmetic additions instead of modular additions and give its implementation with elementary gates. We consider a system composed of a collection of two-level systems (qubits).

Let a, b, which are integers from 0 to $2^n - 1$, be the numbers to add. Let $a_1 a_2 \ldots a_n$ and $b_1 b_2 \ldots b_n$ be the binary representations of a and b, where $a = a_1 2^{n-1} + a_2 2^{n-2} + \ldots + a_n 2^0$ and $b = b_1 2^{n-1} + b_2 2^{n-2} + \ldots + b_n 2^0$. Then $|a\rangle = |a_1\rangle \otimes |a_2\rangle \otimes \ldots \otimes |a_n\rangle$ and $|b\rangle = |b_1\rangle \otimes |b_2\rangle \otimes \ldots \otimes |b_n\rangle$.

Draper's circuit first computes the quantum Fourier transform of a, evolving $|a\rangle$ into $|\phi(a)\rangle$:

$$|\phi(a)\rangle = \mathrm{QFT}\,|a\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \mathrm{e}^{i\frac{2\pi ak}{N}} |k\rangle\,, \tag{14}$$

where $N = 2^n$. Then the circuit computes the sum, using the n qubits that represent the number b to take $|\phi(a)\rangle$ into $|\phi(a+b)\rangle$. To perform the addition, the circuit decomposes the CZ gates presented in Sect. 3 into conditional rotation phase gates of the form:

$$R_l = \begin{bmatrix} 1 & 0 \\ 0 & \mathrm{e}^{\frac{2\pi i}{2^l}} \end{bmatrix}. \tag{15}$$

These gates are controlled by the n qubits that represent the number b. The combined effect of all the gates is to introduce a total phase $\mathrm{e}^{\frac{2\pi ibk}{N}}$ for each state $|k\rangle$, so that the qubits containing b keep the same value while the qubit register containing the QFT of a now stores $|\phi\,(a+b)\rangle$.

We can extend the scheme to perform non-modular additions by encoding a into a larger register. We represent the number a using $n+1$ qubits so that $|a\rangle =$

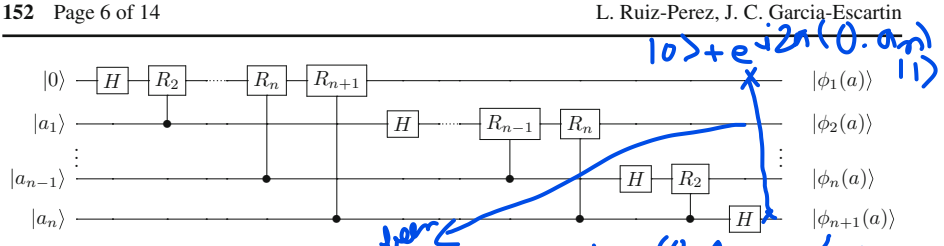

Fig. 1 QFT of the state $|0\rangle\,|a\rangle$

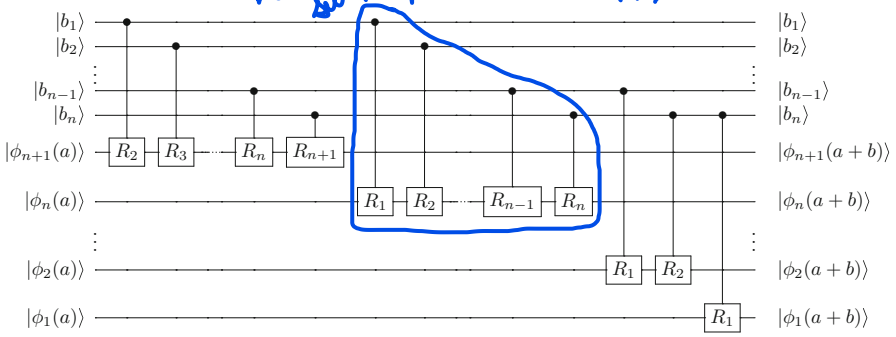

Fig. 2 Arithmetic (non-modular) sum in the transform domain

$|0\rangle\,|a_1\rangle\,|a_2\rangle\ldots|a_n\rangle$. The second step is computing the QFT of $|a\rangle$,

$$\mathrm{QFT}\,|a\rangle = \frac{1}{\sqrt{2^{n+1}}}\sum_{k=0}^{2^{n+1}-1} \mathrm{e}^{i\frac{2\pi ak}{2^{n+1}}}\,|k\rangle\,, \tag{16}$$

with the QFT circuit shown in Fig. 1, where the states $\left|\phi_j(a)\right\rangle$ represent the jth qubit of the phase state $|\phi(a)\rangle$ encoding a. For simplicity, in this figure we have omitted the sequence of SWAP gates needed to invert the order of the output qubits [29]. Alternatively, since we know the order of the qubits, we can connect them to the next stage in the right order.

Once we have $|\phi(a)\rangle$, we add the number b using controlled phase rotation gates as in Draper's scheme. We add a and b by applying the conditional phase rotation

$$\mathrm{e}^{2\pi i\frac{(a_j+b_j)2^{n-j}k_s2^{n+1-s}}{2^{n+1}}} = \mathrm{e}^{2\pi i\frac{(a_j+b_j)k_s}{2^{j+s-n}}} \tag{17}$$

that depends on the jth qubits of the representation of the numbers to be added and is applied on the sth qubit in the transformed register containing superpositions of states $|k\rangle = |k_1\rangle\otimes\cdots\otimes|k_{n+1}\rangle$. The gate is controlled by the jth qubit of $|b\rangle$ and only produces a change if $b_j = 1$. We choose the conditional phase rotation gates $R_l = R_{j+s-n}$ when $j+s-n > 0$. If $j+s-n \leq 0$, we are applying the phase $\mathrm{e}^{2\pi 2^{n-j-s}i} = 1$ and the state remains unaltered. The resulting circuit is shown in Fig. 2.

As a result, the register containing the QFT of a now stores $|\phi\,(a+b)\rangle$. We still perform a modular addition, but, by adding an ancillary qubit to encode a, we avoid

overflow and leave space to recover the integer addition of a and b. When we perform the inverse QFT and go back to the computational basis, the qubit register with the sum has the correct result. This QFT adder has the minimum possible size to contain the result and needs no additional ancillary qubits, unlike some reversible adders based on classical schemes [4,5,7,9].

5 Computing the mean with the QFT

A simple extension to the quantum adder can compute the arithmetic mean of a set of integers. We consider again N integers $x_1, x_2, \ldots, x_N$ encoded into a state $|x_1\rangle\,|x_2\rangle \ldots |x_N\rangle$ and an ancillary $|0\rangle$ qudit. If we replace the CZ gates in Eq. (11) by the CZ^N gates defined in Eq. (6), we have the evolution

$$\begin{aligned} &\mathrm{IQFT}_{N+1}\left(\prod_{m=1}^{N}\mathrm{CZ}^N_{m,N+1}\right)\mathrm{QFT}_{N+1}\,|x_1\rangle\,|x_2\rangle \ldots |x_N\rangle\,|0\rangle \\ &\quad = |x_1\rangle\,|x_2\rangle \ldots |x_N\rangle \left|\frac{1}{N}\sum_{m=1}^{N} x_m \pmod d\right\rangle, \end{aligned} \tag{18}$$

which produces the desired average.

Notice that, in this case, the arithmetic mean is always equivalent to the modular addition. The mean of numbers from 0 to $d-1$ is always between 0 and $d-1$. However, there appears a new problem. In general, the result is not an integer and the phase to computational basis transition of the inverse QFT shown in Eq. (2) does not return an integer in the computational basis. To solve this, we can expand the state space and encode the numbers in the computational basis with a fixed point representation. In Sect. 8, we give the details of the circuit in terms of qubits for a general weighted sum. For that scenario, with $\log_2(Nd)$ qubits we recover the correct mean value.

Alternatively, we can use the methods of phase estimation quantum algorithms which, for $d = 2^m$, give the best possible m-bit approximation to any arbitrary phase ϕ between 0 and 1 in a term $\mathrm{e}^{i2\pi\phi}$ with a probability of, at least, $4/\pi^2$, which can be improved at the cost of a larger circuit [30].

6 Weighted sums and multiplication by a constant

The method of the previous section can be modified to compute any weighted sum

$$\sum_{m=1}^{N} a_m x_m. \tag{19}$$

We start by encoding the numbers into a state

$$|x_1\rangle\,|x_2\rangle \ldots |x_N\rangle\,|0\rangle \tag{20}$$

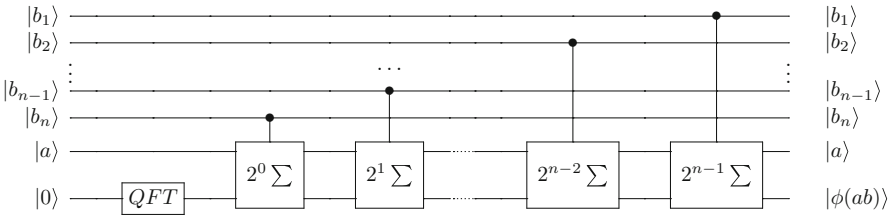

Fig. 3 QFT multiplier as a concatenation of controlled weighted sum blocks

and then apply the gate sequence

$$\mathrm{IQFT}_{N+1}\left(\prod_{m=1}^{N}\mathrm{CZ}_{m,N+1}^{\frac{1}{a_m}}\right)\mathrm{QFT}_{N+1}. \tag{21}$$

The resulting state

$$|x_1\rangle\,|x_2\rangle\ldots|x_N\rangle\,|a_1x_1+a_2x_2+\ldots+a_Nx_N \pmod d\rangle \tag{22}$$

returns the modulo d weighted sum. If we want to obtain the non-modular weighted sum or add signed numbers, we might need to choose a different dimension for the ancillary qudit. We can recycle the encoding and circuit changes we used to modify adders in Sect. 3.

A particular case happens when all the a_m are positive and $\sum_m a_m = 1$, like in the example of the arithmetic mean where $a_m = \frac{1}{N}$ for all m. Then the result is guaranteed to be between 0 and $d-1$ and the modulo d sum and the total weighted sum are always equal. However, if the a_m are not integers we need either to increase the state space and use a fixed point representation or to include a phase estimation stage to recover our result.

Multiplication by a constant can be seen as a particular case of weighted sum. We can multiply two numbers x and b, with b constant and x any integer from 0 to $d-1$, using the binary decomposition of b. If b has n bits, we can write the product bx as the sum

$$(b_1 2^{n-1}\cdot b_2 2^{n-2}\cdots b_{n-1}2^1\cdot b_n 2^0)x = \sum_{m=1}^{n} b_m 2^{n-m}x, \tag{23}$$

which is a weighted sum with integer coefficients $a_m = b_m 2^{n-m}$ and where all the x_m are equal. Section 7 describes a variation of this method that gives a QFT multiplier.

7 QFT multiplier

We can design a quantum circuit to multiply two n-bit numbers by performing n consecutive controlled QFT additions. The result will be a $2n$-qubit register encoding the number $a \cdot b$. The circuit is shown in Fig. 3.

The first adder block, labelled as $2^0\Sigma$, takes as input the n qubits representing a number a and $2n$ qubits representing the number 0. Before starting, we prepare an initial ancillary state taking the quantum Fourier transform of number 0, *i.e.* $|\phi(0)\rangle$, and then, the $2^0\Sigma$ block applies a series of conditional phase rotation gates to evolve the state into $|\phi(0+a)\rangle$. The block is controlled by the least significant qubit of $|b\rangle$ so it produces the output state

$$|a\rangle \left|\phi(0+b_n 2^0 a)\right\rangle. \tag{24}$$

The next step is a second QFT adder controlled by b_{n-1}. Now the phase addition is scaled by a factor 2^1 so that the output state will be

$$\left|\phi(0+b_n 2^0 a + b_{n-1}2^1 a)\right\rangle. \tag{25}$$

We now proceed in a similar fashion with the remaining blocks. When the last QFT adder is applied, the output state is

$$\left|\phi(0+b_n 2^0 a + b_{n-1}2^1 a + \ldots + b_2 2^{n-2} a + b_1 2^{n-1} a)\right\rangle = |\phi(0+ab)\rangle = |\phi(ab)\rangle\,. \tag{26}$$

The key to compute the product $a \cdot b$ is to select the proper conditional phase rotation gates to implement each QFT adder block. After computing the QFT of 0, we obtain the output state

$$\text{QFT}\,|0\rangle = \frac{1}{\sqrt{2^{2n}}} \sum_{k=0}^{2^{2n}-1} \mathrm{e}^{i\frac{2\pi 0 k}{2^{2n}}} |k\rangle = |\phi(0)\rangle\,, \tag{27}$$

where $k = k_1 2^{2n-1} + k_2 2^{2n-2} + \ldots + k_{2n} 2^0 = \sum_{s=1}^{2n} k_s 2^{2n-s}$. In order to take $|\phi(0)\rangle$ to $\left|\phi(0+b_j 2^{n-j} a)\right\rangle$, we need to use phase rotation gates controlled by b_j and by each a_i, chosen so that they apply a phase rotation

$$\mathrm{e}^{i\frac{2\pi(a_i 2^{n-i} b_j 2^{n-j}) k_s 2^{2n-s}}{2^{2n}}} = \mathrm{e}^{i\frac{2\pi a_i b_j k_s}{2^{i+j+s-2n}}}. \tag{28}$$

Therefore, we select conditional rotation gates of the form $R_l = R_{i+j+s-2n}$, where $i + j + s - 2n > 0$, to implement the QFT adder block controlled by b_j.

In this circuit, we have chosen the size of the ancillary register so that we get the exact value of $a \cdot b$ instead of a modular multiplication. We can vary the size of the ancillary register and modify the R_l gates accordingly to obtain any desired modular multiplication in moduli that are powers of two of the size of the ancillary register.

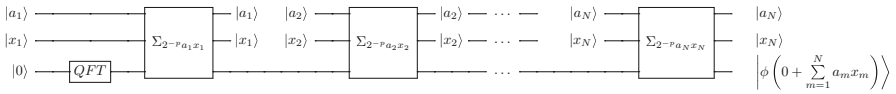

Fig. 4 Controlled weighted sum (block diagram)

8 Controlled weighted sum

Using the same methods, we can implement a quantum circuit to compute the weighted sum

$$\sum_{m=1}^{N} a_m x_m \tag{29}$$

for any combination of input weights a_m and numbers x_m. The weights and input integers can be in a superposition of different values. To build such a circuit, we can use an architecture similar to the QFT multiplication block introduced in Sect. 7.

Each qubit of a_m controls how to add the contribution of each qubit of x_m. If we directly use the circuit of Fig. 3, we compute the weighted sum for integer weights. However, the discrete weights a_m can be adjusted to any range of interest simply by introducing the appropriate factor in the corresponding CZ^F gates. We define a precision variable p so that the weights a_m are the integers represented by each binary string encoded in the weight qubits divided by 2^p. Each input weight in the computational basis can be interpreted as a fixed point binary number $|a\rangle = \left|b_1 \cdots b_{q-p}.b_{q-p+1} \cdots b_q\right\rangle$. These non-integer values are encoded into the phase representation, and when, at the end of the computation, we take the inverse QFT, we recover the correct result in the computational basis in the corresponding fixed point encoding. If we use q qubits to store each weight a_m, we can obtain weights a_m in a range $0 \leq a_m \leq 2^{q-p} - 2^{-p}$ with a precision 2^{-p}. The values of p and q can be adjusted to define any desired range of values with the required precision.

The circuit can be implemented using N modified versions of the QFT multiplication blocks as shown in Fig. 4. We first compute the quantum Fourier transform of $|0\rangle$. Then we apply the first multiplication block, which takes the input state

$$|\mathbf{a_1}\rangle\,|\mathbf{x_1}\rangle \otimes \cdots \otimes |a_N\rangle\,|x_N\rangle\,|\phi(0)\rangle \tag{30}$$

and returns the output state

$$|\mathbf{a_1}\rangle\,|\mathbf{x_1}\rangle \otimes \cdots \otimes |a_N\rangle\,|x_N\rangle\,|\phi(0+\mathbf{a_1x_1})\rangle\,. \tag{31}$$

The second multiplier acts in a similar manner, taking the input state

$$|a_1\rangle\,|x_1\rangle\,|\mathbf{a_2}\rangle\,|\mathbf{x_2}\rangle \otimes \cdots \otimes |a_N\rangle\,|x_N\rangle\,|\phi(0+a_1x_1)\rangle \tag{32}$$

and returning the output state

$$|a_1\rangle\,|x_1\rangle\,|\mathbf{a_2}\rangle\,|\mathbf{x_2}\rangle \otimes \cdots \otimes |a_N\rangle\,|x_N\rangle\,|\phi(0+a_1x_1+\mathbf{a_2x_2})\rangle\,. \tag{33}$$

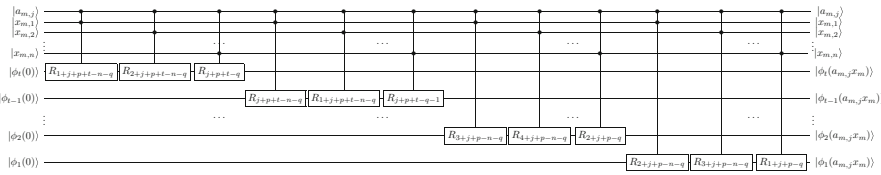

Fig. 5 Example block of a controlled weighted sum

After applying all the multipliers, we get the final output state

$$|a_1\rangle\,|x_1\rangle\,|a_2\rangle\,|x_2\rangle\otimes\cdots\otimes|a_N\rangle\,|x_N\rangle\,|\phi(0+a_1x_1+a_2x_2+\ldots+a_Nx_N)\rangle\,. \quad (34)$$

Figure 5 shows an example of the gates inside each of the multiplication blocks. The figure describes the circuit controlled by the jth qubit of the mth weight and how it controls the R_l operations on the qubits of the mth value. The subindices are written for a system that computes the weighted sum modulo 2^t for weights that have q bits with a precision 2^{-p} and input integer numbers x_m with n bits. If we want to find the non-modular weighted sum and there are N values to be added, the ancillary register must have $t = \lceil \log_2(N2^q2^n)\rceil = \lceil (q+n)\log_2(N)\rceil$ qubits so that we can maintain the precision and there is no overflow.

The block on Fig. 5 must be repeated for all the qubits of $|a_m\rangle$. The gate acting on the uth qubit of the ancillary register that is controlled by the ith qubit of $|x_m\rangle$ and the jth qubit of $|a_m\rangle$ must produce a phase shift

$$\mathrm{e}^{i\frac{2\pi x_i 2^{n-i}a_j 2^{q-p-j}k_u 2^{t-u}}{2^t}} = \mathrm{e}^{i\frac{2\pi x_i a_j k_u}{2^{i+j+u+p-n-q}}}, \quad (35)$$

which corresponds to the conditional phase rotation gate $R_l = R_{i+j+u+p-n-q}$.

Notice that many from these gates cancel. Any gate for which $i+j+u+p \le n+q$ introduces a phase that is an integer multiple of 2π and can be eliminated from the final scheme.

The general weighted sum circuit has applications in both its modular and non-modular forms. The non-modular controlled weighted sum gives the dot product $\vec{a}\cdot\vec{x}$ of two vectors $\vec{a} = (a_1,\ldots,a_m)$ and $\vec{x} = (x_1,\ldots,x_m)$. The described modular operation for qubits gives an inner product in a vector space V over a field consisting in a collection of m-tuples from $F^m_{p^k}$ where $p^k = 2^n$. The m elements x_i and a_i, which are each n-bit binary strings, can be seen as vectors $\vec{x}, \vec{a} \in V$ and the controlled weighted sum modulo 2^n as their inner product in V.

For instance, when we consider a system where the result is stored in one qubit, the resulting modular weighted sums recover the inner product modulo 2, $\sum_{i=1}^n a_i \cdot x_i \mod 2$, which appears in many quantum information protocols [31,32]. The default version of our circuit, which performs modular weighted sums, generalizes this inner product to other moduli.

9 Discussion

The quantum Fourier transform offers a versatile way to perform modular and non-modular arithmetic on a quantum computer. We have discussed how QFT adders and multipliers are compact circuits for quantum arithmetic that need no ancillary qubits and put forward a few modifications to accommodate general non-modular operations, signed numbers and different moduli. We have also discussed the qubit implementation of both QFT adders and multipliers. We can implement a QFT adder using $O(n^2)$ gates, while the multiplier would need $O(n^3)$ gates for integers encoded with n bits.

We have also shown that certain operations, like the arithmetic mean or any weighted average, can be implemented with the same number of gates as a basic QFT addition. If the elementary gates can be classically programmed, this allows for a flexible quantum modular weighted sum calculator that avoids computing each weight-value product. If we sum N integers, the obvious classical implementation would require N multiplications and N sums with a complexity $O(N(M + A))$ where M and A are the complexities of modular multiplication and addition, respectively. For a good choice of multiplication and addition methods, the bottleneck is M and we can obtain a complexity $O(Nn \log^2 n \log\log n)$ for Montgomery multiplication [33] with the Schönhage–Strassen algorithm [34]. For the our modular quantum weighted sum with integers and fixed integer weights, the complexity is comparable to N sums plus the QFT and its inverse at the beginning and the end. The total complexity is $O(Nn+n^2)$. For a few long integers, the dominant factor is the QFT overhead. Each sum takes $O(n)$ gates, and they only become important if N is of the order of the number of bits of each integer or greater. The QFT method is therefore particularly interesting when we compute the weighted sums of a large list of integers.

Additionally, we have presented a quantum circuit that computes the weighted sum for both quantum weights and values. It can be implemented using $O(Ntqn)$ gates, where n and q are the number of bits used to encode each number $|x_i\rangle$ and each weight $|a_i\rangle$, respectively. Provided $n = q = t$, the implementation of the scheme would require a number of gates $O(Nn^3)$. The circuit for N numbers takes as many gates as N multipliers but needs no additions. In all the cases, there is an overhead in the form of the direct and inverse QFT with $O(t^2)$ gates for modulo 2^t operations.

The controlled quantum weighted adder opens many applications. Optimizing weighted sums is a problem that appears in data processing and network planning among others. Many machine learning algorithms need to compute weighted sums [35]. For instance, neural network training requires choosing a set of weights that minimizes a weighted sum of the samples. A quantum weighted adder that has a uniform superposition of all the possible discrete weights for a given register size as its input can be combined with the quantum algorithm for finding the minimum [36] to obtain a quadratic speedup in the optimization problem. For a good enough weight precision, this can be very helpful.

While many quantum machine learning algorithms encode the weights in the probability amplitudes of a superposition [37,38], if we want to avoid the still challenging problem of creating the initial amplitude superposition [39] and enter the weights and the data as states, our weighted adder gives a reversible implementation that can work

with superpositions. As an alternative, we can just take direct translations of classical machine learning circuits and use methods based on Grover's algorithm [40] to give more modest, but reliable for any general case, quadratic speedups.

The presented weighted sum block can also be used whenever we need an inner product inside a quantum algorithm.

All these circuits expand the available QFT-based arithmetic operations available for quantum computers and show the potential of operations in a phase encoding.

Acknowledgements L. Ruiz-Perez has been funded by the FPI fellowship programme of the Spanish Ministry of Economy, Industry and Competitiveness (Grant BES-2015-074514). J.C. Garcia-Escartin has been funded by Project TEC2015-69665-R (MINECO/FEDER, UE) and Junta de Castilla y León Project No. VA089U16.

References

1. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. **26**(5), 1484 (1997)
2. Vedral, V., Barenco, A., Ekert, A.: Quantum networks for elementary arithmetic operations. Phys. Rev. A **54**(1), 147–153 (1996)
3. Beckman, D., Chari, A.N., Devabhaktuni, S., Preskill, J.: Efficient networks for quantum factoring. Phys. Rev. A **54**, 1034–1063 (1996)
4. Gossett, P.: Quantum carry-save arithmetic. arXiv:quant-ph/9808061v2 (1998)
5. Cuccaro, S.A., Draper, T.G., Kutin, S.A., Moulton, D.P.: A new quantum ripple-carry addition circuit. arXiv:quant-ph/0410184v1 (2004)
6. Van Meter, R., Itoh, K.M.: Fast quantum modular exponentiation. Phys. Rev. A **71**, 052320 (2005)
7. Draper, T.G., Kutin, S.A., Rains, E.M., Svore, K.M.: A logarithmic-depth quantum carry-lookahead adder. Quantum Inf. Comput. **6**(4), 351–369 (2006)
8. Álvarez-Sánchez, J.J., Álvarez-Bravo, J.V., Nieto, L.M.: A quantum architecture for multiplying signed integers. J. Phys. Conf. Ser. **128**(1), 012013 (2008)
9. Takahashi, Y., Kunihiro, N.: A fast quantum circuit for addition with few qubits. Quantum Inf. Comput. **8**(6), 636–649 (2008)
10. Takahashi, Y., Tani, S., Kunihiro, N.: Quantum addition circuits and unbounded fan-out. Quantum Inf. Comput. **10**(9&10), 0872–0890 (2010)
11. Markov, I.L., Saeedi, M.: Constant-optimized quantum circuits for modular multiplication and exponentiation. Quantum Inf. Comput. **12**(5&6), 361–394 (2012)
12. Thapliyal, H., Ranganathan, N.: Design of efficient reversible logic-based binary and BCD adder circuits. ACM J. Emerg. Technol. Comput. Syst. (JETC) **9**(3), 17 (2013)
13. Nguyen, T.D., Van Meter, R.: A resource-efficient design for a reversible floating point adder in quantum computing. ACM J. Emerg. Technol. Comput. Syst. (JETC) **11**(2), 13 (2014)
14. Davies, J.T., Rickerd, C.J., Grimes, M.A., Guney, D.O.: An n-bit general implementation of Shor's quantum period-finding algorithm. Quantum Inf. Comput. **16**(7&8), 700–718 (2016)
15. Babu, H.M.H.: Cost-efficient design of a quantum multiplier-accumulator unit. Quantum Inf. Process. **16**(1), 30 (2017)
16. Meter, R.V., Munro, W.J., Nemoto, K., Itoh, K.M.: Arithmetic on a distributed-memory quantum multicomputer. J. Emerg. Technol. Comput. Syst. **3**(4), 2:1–2:23 (2008)
17. Trisetyarso, A., Van Meter, R.: Circuit design for a measurement-based quantum carry-lookahead adder. Int. J. Quantum Inf. **08**(05), 843–867 (2010)
18. Wiebe, N., Roetteler, M.: Quantum arithmetic and numerical analysis using repeat-until-success circuits. Quantum Inf. Comput. **16**(1&2), 134–178 (2016)
19. Choi, B.-S., Van Meter, R.: A $\Theta(\sqrt{n})$-depth quantum adder on the 2D NTC quantum computer architecture. J. Emerg. Technol. Comput. Syst. **8**(3), 24:1–24:22 (2012)
20. Draper, T.G.: Addition on a quantum computer. arXiv:quant-ph/0008033v1 (2000)
21. Beauregard, S.: Circuit for Shor's algorithm using 2n+3 qubits. Quantum Inf. Comput. **3**(2), 175–185 (2003)

22. Beauregard, S., Brassard, G., Fernandez, J.M.: Quantum arithmetic on Galois fields. arXiv:quant-ph/0301163v1 (2003)
23. Pavlidis, A., Gizopoulos, D.: Fast quantum modular exponentiation architecture for Shor's factoring algorithm. Quantum Inf. Comput. **14**(7& 8), 649–682 (2014)
24. Maynard, C., Pius, E.: A quantum multiply-accumulator. Quantum Inf. Process. **13**(5), 1127–1138 (2014)
25. Daboul, J., Wang, X., Sanders, B.C.: Quantum gates on hybrid qudits. J. Phys. A Math. Gen **36**(10), 2525–2536 (2003)
26. Fushman, I., Englund, D., Faraon, A., Stoltz, N., Petroff, P., Vučković, J.: Controlled phase shifts with a single quantum dot. Science **320**(5877), 769–772 (2008)
27. Nam, Y.S., Blümel, R.: Robustness of the quantum Fourier transform with respect to static gate defects. Phys. Rev. A **89**(4), 769–772 (2014)
28. Hirose, M., Cappellaro, P.: Coherent feedback control of a single qubit in diamond. Nature **532**(7597), 77–80 (2016)
29. Nielsen, M., Chuang, I.L.: Quantum Computation and Quantum Information. Cambridge University Press, Cambridge (2000)
30. Cleve, R., Ekert, A., Macchiavello, C., Mosca, M.: Quantum algorithms revisited. Proc. R. Soc. Lond. A **454**, 339–354 (1998)
31. Bernstein, E., Vazirani, U.: Quantum complexity theory. SIAM J. Comput. **26**(5), 1411–1473 (1997)
32. Terhal, B.M., Smolin, J.A.: Single quantum querying of a database. Phys. Rev. A **58**(3), 1822–1826 (1998)
33. Montgomery, P.L.: Modular multiplication without trial division. Math. Comput. **44**(170), 519–521 (1985)
34. Schönhage, A., Strassen, V.: Schnelle Multiplikation großer Zahlen. Computing **7**(3), 281–292 (1971)
35. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Data Mining, Inference, and Prediction. Springer Series in Statistics, Springer, New York (2009)
36. Dürr, C., Hoyer, P.: A quantum algorithm for finding the minimum. eprint arXiv:quant-ph/9607014 (1996)
37. Rebentrost, P., Mohseni, M., Lloyd, S.: Quantum support vector machine for big data classification. Phys. Rev. Lett. **113**(13), 130503 (2014)
38. Schuld, M., Sinayskiy, I., Petruccione, F.: An introduction to quantum machine learning. Contemp. Phys. **56**(2), 172–185 (2015)
39. Aaronson, S.: Read the fine print. Nat. Phys. **11**(4), 291–293 (2015)
40. Grover, L.K.: Quantum mechanics helps in searching for a needle in a haystack. Phys. Rev. Lett. **79**(2), 325–328 (1997)