Quantum FPGA Architecture Design

Jialin Chen[1,2], Lingli Wang[1], Bin Wang[2]*
[1] State Key Laboratory of ASIC & System, Fudan University, P. R. China
e-mail address: llwang@fudan.edu.cn
[2]Department of Electrical Engineering, Fudan University, P. R. China

***Abstract*— A Quantum FPGA (QFPGA) architecture is presented for programmable quantum computing, which is a hybrid architecture combining the advantages of the measurement-based quantum computation and the qubus system. QFPGA consists of Quantum Logic Blocks (QLBs) and Quantum Routing Channels (QRCs). The QLB is used to realize a small quantum logic while the QRC is to combine them properly for larger logic realization. There are two types of buses in QFPGA, the local bus in the QLB and the global bus in the QRC, which are to generate the cluster states and general multiqubit rotations around the z axis respectively. However for some applications such as Grover's algorithm and *n*-qubit quantum Fourier transform, one QLB can be configured for four-qubit phase shift module and four-qubit quantum Fourier transform respectively.**

Keywords—quantum FPGA architecture; QLB; QRC; Grover's algorithm; Quantum Fourier Transform.

I. Introduction

Using various quantum mechanical effects such as entanglement and superposition, quantum computation can provide massive performance speedup for some important problems such as integer factorization, database search, global binary optimization and so on[1]. Much of the effort thus far has been focused on the research of quantum algorithms, architecture and physical implementation. In the meantime, several methods have been proposed [2-3] to emulate for quantum computation using FPGAs.

In this paper, however, our goal is to extend classical FPGA to quantum FPGA (QFPGA) to realize universal quantum computing that can be appropriately programmed. The proposed QFPGA architecture is shown in Fig. 1, in which we utilize the MBQC [4-5] with embedded classic bits to realize the programmable quantum computation with the aid of the qubus system [6].

As we focus on the programmable architecture, our main contributions are:

1. A quantum FPGA architecture is proposed for universal quantum computation which consists of two parts: "quantum routing channels" (QRC) and "quantum logic block" (QLB).
2. The proposed QFPGA is a hybrid architecture which combines MBQC and qubus system. By using qubus to design a scheme for dynamic cluster state generation, we will show that our architecture is feasible as the error probability ϵ is below 10^{-2}, which is an accepted threshold of the MBQC

This research is supported by National Natural Science Foundation of China (61131001, 61171011) and National 863 Program of China (2009AA012201).

II. Preliminary

A. Notations

Throughout this paper, we use H represents a Hadmard gate, $R_i, i \in \{x, y, z\}$ represents a single qubit rotation gate around x, y or z axis, and the identity operator by I. Moreover, we use either X, Y, Z or $\sigma_x, \sigma_y, \sigma_z$ to represent the Pauli-X matrix, Pauli-Y matrix and Pauli-z matrix respectively [1].

B. Quantum logic synthesis

Two quantum circuits are equivalent if matrix products of their corresponding gates are identical. Here, we use a method based on CS and QS decompositions [8] to decompose an arbitrary *n*-qubit gate into a circuit containing six generic *(n-1)*-qubit gates and three multi-qubit diagonal gates D_n as shown in Fig. 2. By using CS and QS decompositions recursively, any $U(2^n)$ can be decomposed into several $U(4)$ and D_k gates ($2 \le k \le n$). By the way, we choose U(4) because it is suitable for the QLB realization which will be discussed in Section III.

III. The quantum programmable architecture

As described in Sec. II B, any quantum circuits can be decomposed recursively into a sequence of D gates and two-

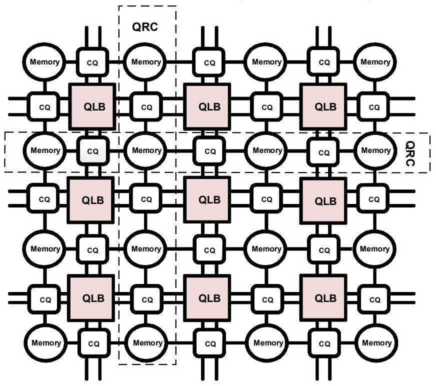


Fig. 1. Quantum FPGA architecture..

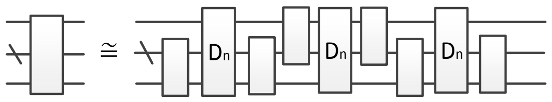


Fig. 2. The decomposition of a general n-qubit quantum gate.

qubit quantum gates. Correspondingly, our architecture consists of two parts: quantum routing channels (QRCs) and quantum logic blocks (QLBs). A QRC consists of the memory qubits (long-lived qubits) and channel qubits (ancilla qubits), while QLBs are short-lived qubits that possess short coherence time. The different choices of qubits are due to that the running time of a QRC is often longer than a QLB.

In next three parts, we will describe the details of QRCs, QLBs and how they are combined to build a programmable architecture.

A. Quantum routing channel (QRC)

The main purpose of QRCs is to realize D gates, as one of the advantages of qubus is that we can use the bus to more efficient cluster-state construction, especially the star graph state. Consider the following sequence of unitary operators for n qubits to generate star graph states:

$$\prod_{l=1}^{n} D(\beta\sigma_{z,l})D(-i\beta\sigma_{z,a})\prod_{l=1}^{n} D(-\beta\sigma_{z,l})D(i\beta\sigma_{z,a})$$
$$= \exp\left[2i\beta^2\sigma_{z,a}\left(\textstyle\sum_{l=1}^{n}\sigma_{z,l}\right)\right] \qquad (1)$$

where $\beta = \sqrt{\pi/8}$. Then, we use this method to construct D_n gates. Obviously, every D_n gate has 2^{n-1} independent variables from (4), it can be written into the expression:

$$D_n = e^{i\left(Z\otimes I\cdots\otimes I\alpha_1 + Z\otimes I\cdots\otimes Z\alpha_2 + \cdots + Z\otimes Z\cdots\otimes Z\alpha_{2^{n-1}}\right)} \qquad (2)$$

Where Z and I represent Pauli-Z matrix and identity matrix, and $\alpha_1, \alpha_2, \ldots \alpha_{2^{n-1}}$ are 2^{n-1} independent variables. Therefore, $2^{n-1}-1$ multi-qubit rotation gates are enough to perform a D_n gate, and to realize these gates, each of them needs a certain star graph states to be measured [9].

Another advantage of the qubus system is that we can reuse the bus[6], i.e. to construct several star graph states in one bus. Because in each step we could couple a new channel qubit and couple or decouple some memory qubits to opposition quadrature of the bus. The reader may easily verify that if this method is used to implement n-qubit D gate where $n>3$, some memory qubits must be visited more than once.

In summary, the detail steps of implementing a general D_n gate are as follows:

1. Prepare all required ancilla qubits in the state $|+\rangle$ as channel qubits.
2. Couple all n memory qubits to position (momentum) quadrature of the bus and a channel qubit to momentum (position) quadrature of the bus.
3. Measure the channel qubit in certain basis $\left(\theta, -\frac{(1+m)\pi}{2}\right)$, where m is the degree of this channel qubit.
4. Decouple or re-couple a certain memory qubit and simultaneously couple a new channel qubit to the bus.
5. Repeat steps 3 and 4 until all desired star graph states are generated and measured.

An n-qubits D_n gate requires $2^{n-1}-1$ channel qubits at most, thus as long as required channel qubits can be obtained, QRCs are efficient to implement any D_n gates.

B. Quantum logic block (QLB)

The main function of QLBs is to realize the simple quantum logic: a general two-qubit quantum gate. As discussed in Sec. II, the decomposition of a general two-qubit quantum gate is shown in Fig. 3.

In principle, any two-qubit quantum gate can be realized in certain cluster states via the traditional MBQC method. However, as we use the qubus method to generate entanglements between two qubits, a slightly modified cluster states could be adopted, i.e. using qubus to generate three R_{zz} gates directly, three ancilla qubits can be saved. Therefore, the final cluster states needed is shown in Fig. 4, where three dotted line mean three $R_{zz}(\theta)$ gates between two qubits and numbers in circles represent the order of measurements and two gray circles represent two output qubits (since some qubits will be measured at the same cycle, a capital letter is appended after the numbers for discrimination). In total, 14 qubits are needed to realize a general two-qubit quantum gate. Because of the adaptive measurements, theoretically at least it will take 8 clock cycles of measurements and three $R_{zz}(\theta)$ gates to complete the computation. Furthermore, two output qubits can be ignored, as they can be memory qubits inside a QRC.

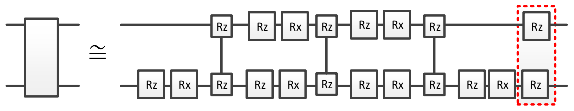


Fig. 3. The general decomposition of a two-qubit quantum gate.

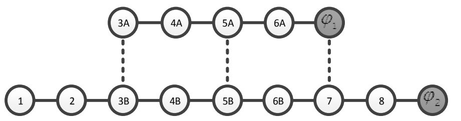


Fig. 4. The cluster states needed to realize a general two-qubit quantum gate.

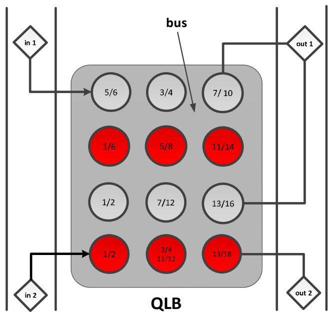


Fig. 5. (Color in electronic version) Schematic of the QLB (shadow).

Then we focus on how to realize the rest of 12-qubit cluster states within a QLB. Suppose that these 12 qubits form a 4× 3 lattice and a local bus within it, we can use it to generate desired cluster states.

As shown in Fig. 5, grey qubits are coupled to position quadrature of the bus and red qubits are coupled to momentum quadrature of the same bus, this is because the displacement operators on the qubus allow a qubit on one quadrature to become entangled with all qubits on the other. For example to generate the cluster states of Fig. 4, if qubits 1 and 3B couple to the momentum quadrature of the bus, then qubit 2 should be coupled to the position quadrature and so on. To avoid the unwanted entanglement in the QLB, four qubits at most can remain on the bus: one position-quadrature qubit and three momentum-quadrature qubits or vice versa. By using this method properly, the desired cluster states of QLB can be realized.

From Fig. 5, we can say that every qubit of the QLB has a unique clock cycle as its programmable point, like "1/2", which means this qubit couple to the bus at 1st clock cycle and decoupled at 2nd clock cycle. Therefore, Fig. 5 is just one QLB configuration only, as it is the clock cycles of each qubit that decide the shape of cluster states rather than its position of the lattice.

In summary, the programmability of QLB can be implemented through a programmable local bus which can interact with these qubits in the suitable sequence

C. *Error analysis*

In this part, we calculate the dephasing error of our architecture. For N bus operations, the probability of dephasing is [7]:

$$\epsilon = \frac{1}{2}[1 - \exp(-N\gamma\tau - 4C\eta\beta^2)] \qquad (3)$$

Where τ is the time to perform for one bus operation and γ is the dephasing rate for qubits, C is the number of CZ gates constructed per bus and η is the loss parameter for the bus. Consider the construction of the QLB in Fig. 5, then N=32 and C=15. If we take $\gamma\tau = 5 \times 10^{-4}$ and $\eta = 10^{-4}$ for practice [7], then the error of dephasing is $\epsilon = 0.0091$ which is below the error threshold 10^{-2}. On the other hand, coupling 17 qubits (N=34) and creation of 20 CZ gate per bus are the limitation under the error threshold, thus we can efficiently generate a 17-qubit star graph state, i.e. for a rotation gate larger than 16 qubits, we might use two or more buses in the QRC to generate a certain star graph state.

IV. Application of the QFPGA

A. *Grover's Algorithm*

On a classical computer, if there are *N* possible candidates and only one is correct, it obviously takes $O(N)$ operations to find the right answer. Remarkably, Grover's quantum search algorithm (GA) enables this search method to be sped up substantially, requiring only $O(\sqrt{N})$ [1]. The core of the GA is the implementation of the *Grover iteration* or *Grover operator,* which we denote *G*. It can be decomposed into two reflection operations: an oracle O and a diffusion gate *D.* In this section, we will show how to implement GA in the QFPGA. The main idea is to implement oracle O in the QRC and D in the QLB as shown in Fig. 6.

The action of O can be written:

$$O^j = I - 2|j\rangle\langle j| \qquad (9)$$

where $1 \le j \le N$ and we take $N = 2^n$ for simplicity. For example, if *N=16*, then 15 multi-qubit rotation gates are needed to realize this oracle, and the rotation valule of each gate is either $\pi/16$ or $-\pi/16$. These 15 rotation gates are {*IIIZ, IIZI, IIZZ, IZII, IZIZ, IZZI, IZZZ, ZIII, ZIIZ, ZIZI, ZIZZ, ZZII, ZZIZ, ZZZI, ZZZZ*} respectively, e.g. "*IIIZ"* represents a rotation gate $R^{1234}_{IIIZ}(\theta)$, etc. In the QFPGA, as qubus itself can realize any two-qubits rotation around z axis, that is {*IIZZ, IZIZ, ZIIZ, ZIZI, ZZII*} can be directly realized using displacement operator, thus only 9 channel qubits are needed to generate required cluster states effeciently within the QRC. Any other oracle can be derived by perfoming X gates on the relevant qubits before and after peforming the gates (9). Of course, the answer to "Which of the oracles is executed by the black box?" is hidden from us [9].

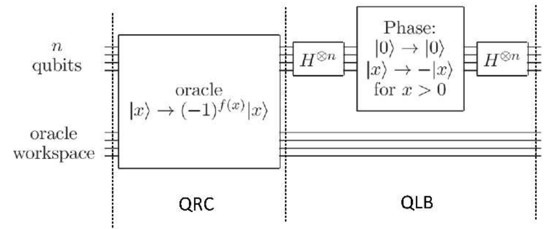
The diffusion gate *D* has the form:

$$D = H^{\otimes n}(2|0\rangle\langle 0| - I)H^{\otimes n} \qquad (10)$$

The details of the optimized way to realize *D* as follows:

1. Generate required cluster states for {*ZZII, ZZZI, ZZZZ, ZIZZ, ZIIZ, ZZIZ, IZZZ, IZZI*} in order.
2. Using displacement operation directly to realize {*IZIZ, ZIZI, IIZZ*} in order and only another four bus operations needed on qubit 1 and 3.
3. Connect four input qubits with four memory qubits by QRC bus to realize {*ZIII, IZII, IIZI, IIIZ*} followed by *H* gates.

Using this method, the QLB-based realization of D is shown in Fig. 7, 32 bus operations and 25 CZ gates are carried out in one bus and its error probability is 0.0099 according to (3). In contrast, the native method which use 5 ancilla qubits will carry out 36 bus operations and 22 CZ gates and its error probability is 0.0106.



n qubits
oracle $|x\rangle \to (-1)^{f(x)}|x\rangle$
oracle workspace
$H^{\otimes n}$
Phase: $|0\rangle \to |0\rangle$ $|x\rangle \to -|x\rangle$ for $x > 0$
$H^{\otimes n}$
QRC
QLB

Fig. 6. Block diagram of the Grover iteration.

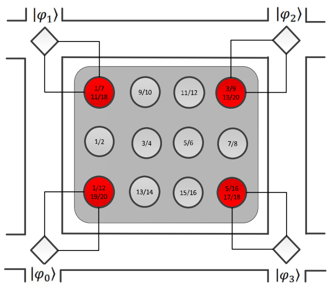


Fig. 7. The QLB-based realization of the diffusion operator D.

B. *Quantum Fourier Transform.*

Quantum Fourier transform [1] is the key of many quantum algorithm such as Shor's factoring algorithm. In this section, we will realize the QFT-4 (QFT on four qubits) in one QLB. For a general QFT, how to combine these QFT-4 module can be seen in [10].

The quantum circuit of QFT-4 are given in Fig. 8, where necessary swap operations are excluded for clarity. In fact, as the output qubits of a QLB are only decided by the sequence of bus operations, e.g. their programmable points, no explicit swap gates are needed. It is easy to find its MBQC version, as H gate following a R_z gate can be realized on the two-qubit cluster states while R_z gate alone can be realized on the three-qubit cluster states. Thus the required cluster states is shown in Fig. 9(a), where dotted lines represent certain controlled-phase gates between two qubits and inputs and outputs should be memory qubits in the QRC. Thus 11 qubits are needed but at the cost of six additional $R_{zz}(\theta)$ gates, shown as dotted lines in Fig. 9 (a). However, these two-qubit rotation gates are not expected to be realized by measurements because the qubus system is a better way to implement them, hence the QLB which combines a local bus and cluster states is an ideal candidate module for the QFT realization.

Then, the QLB-based realization of the QFT-4 is given in Fig. 9 (b), where arrows toward the QLB represent quantum wires at the stage of inputs. Compare to the standard configuration of a QLB (see Fig. 5), one of the differences between them is that here we have 4 inputs memory qubits and they are also served as output qubits simultaneously, as the memory qubits can be reused again. Another difference is that now eight qubits are programmed to couple to the momentum quadrature of the bus and four qubits couple to the position quadrature of the bus.

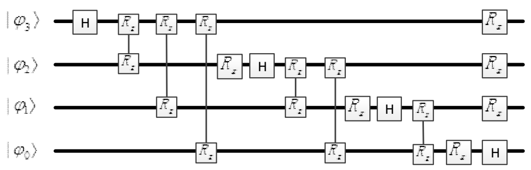


Fig. 8. Quantum circuit of QFT on four qubits

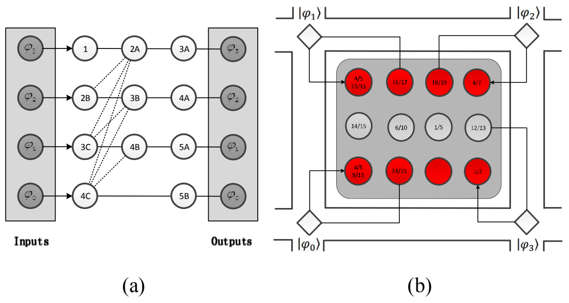


Fig. 9. (Color in electronic version) (a) The required cluster states of the QFT-4 (b) Schematic of the QLB-based realization of the QFT-4.

V. CONCLUSION AND OUTLOOK

We have proposed a programmable architecture called QFPGA for universal quantum computing, which consists of QLBs and QRCs. As any large unitary gate can be decomposed into a sequence of 2-qubit gates and D gates, they will be realized in QLBs and QRCs respectively. Compared with original MBQC, the QFPGA architecture combines small blocks of cluster states by using global buses, so as to make the overall structure more scalable, flexible and also clear. To guarantee an acceptable error, we have described an efficient method for generating cluster states in the QLB, this means that the QFPGA approach is feasible, as fully scalable operation can be achieved. Furthermore, by realizing two applications, we have shown the advantages of this architecture.

VI. Reference

[1] M. Nielsen and I. Chuang, Quantum computation and quantum information, Cambridge University Press, 2000.

[2] Khalid, A.U., Zilic, Z. and Radecka, K., "FPGA Emulation of Quantum Circuits", ICCD 2004, Proceedings. IEEE International Conference on.

[3] José F. Rivera-Miranda et al., "Hardware Emulation of Quantum Fourier Transform", Circuits and Systems (LASCAS), 2011 IEEE Second Latin American Symposium on.

[4] R. Raussendorf, D. E. Browne, and H. J. Briegel, "Measurement-based quantum computation on cluster states", Phys. Rev. A 68, 022312 (2003).

[5] Dan E. Browne, Hans J. Briegel, "One-way Quantum Computation - a tutorial introduction", arXiv:quant-ph/0603226.

[6] K. L. Brown, Ph.D. Thesis, "Using the qubus for quantum computing", University of Leeds, 2011.

[7] C. Horsman, Katherine L. Brown,William J. Munro and Vivien M. Kendon, "Reduce, reuse, recycle for robust cluster-state generation", Phys. Rev. A 83, 042327 (2011).

[8] Vivek V. Shende, Stephen S. Bullock, Igor L. Markov, "Synthesis of Quantum Logic Circuits", IEEE Trans. on CAD 25, 2006.

[9] A. Sehrawat, D. Zemann and B. Englert, "Hybrid Quantum Computation", Phys. Rev. A 83, 022317 (2011).

[10] J. L. Chen, L. L. Wang, E. Charbon, B. Wang, "A programmable architecture for quantum computing", Phys. Rev. A 88, 022311 (2013).