Extended Abstracts of the 2001 International Conference on Solid State Devices and Materials, Tokyo, 2001, pp. 402–403

P-1-2

A Programmable SIMD Processor for Universal Quantum-Circuit Simulator

Shin-ichi O'uchi, Minoru Fujishima[1] and Koichiro Hoh[1] *

School of Engineering, The University of Tokyo
[1]School of Frontier Sciences, The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
Phone: +81-3-5841-6775, FAX: +81-3-5841-8574, E-mail: ohuchi@sf.t.u-tokyo.ac.jp

1 Introduction

Quantum computing is a promising computer model for the next generation. Currently, quantum algorithm has been studied with simulations on the conventional computer. The explosively increasing amount of computing devices or time is needed, however, when the number of quantum bits (qubits) in quantum computing increases linearly. On the other hand, the number of usable devices per chip and its operational speed are increasing due to the progress of silicon technology, year by year. The processor architecture utilizing the massive silicon devices potentially realize the operation time of quantum computing, where the architecture has to be suitable for large scale parallel computation and simple programming.

To simulate quantum circuits by taking a number of steps of the same order as a quantum computer, we present a dedicated processor based on a Single-Instruction-Multiple-Data (SIMD) architecture with a recursive structure which is suitable for scaling. We have experimentally fabricated the processor within a single chip of programmable logic device (PLD) and demonstrated a quantum Fourier transform algorithm [3].

2 Simulation Method

In our simulator, the universal-quantum-gate operations [1, 2] are realized as calculations of particular band diagonal matrices. The three elementary matrices are utilized,

$$U(\theta) = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}, \quad (1)$$

$$U(\phi_0) = \begin{bmatrix} e^{i\phi_0} & \\ & 1 \end{bmatrix}, \quad U(\phi_1) = \begin{bmatrix} 1 & \\ & e^{i\phi_1} \end{bmatrix}. \quad (2)$$

In the case of a gate operation U on i-th qubit within n-qubit quantum circuit, a whole matrix U is given by the tensor product of $E\otimes\ldots\otimes E\otimes U\otimes E\ldots\otimes E$, where E denotes a 2×2 identity matrix. This is equivalent to a band diagonal matrix which has 2^i-th off-diagonal elements. The matrix is operated on the vector $|\Psi\rangle = |\psi_{n-1}\rangle\otimes|\psi_{n-2}\rangle\otimes\ldots|\psi_0\rangle$, where $|\psi_i\rangle = \omega_0|0\rangle + \omega_1|1\rangle$. Controlled unitary operation matrix is given as $|0\rangle\langle 0|\otimes E + |1\rangle\langle 1|\otimes U$, and whole matrix is also derived by the tensor product same as 1-qubit operation. In the $(x_{n-1}x_{n-2}\ldots 0\,x_{j-1}\ldots x_0)_2$-th line (where $\{x_i|0,1\}$) of the matrix of the operation controlled by j-th qubit, the diagonal element is 1 and off-diagonal elements are 0. Plural control qubits are allowed in our simulation.

The qubit-wise observation is simulated by comparing two summations of probabilities, $\sum\langle\Psi|\ldots 0\ldots\rangle\langle\ldots 0\ldots|\Psi\rangle$ and $\sum\langle\Psi|\ldots 1\ldots\rangle\langle\ldots 1\ldots|\Psi\rangle$, where 0 or 1 represents digits of the observation-target qubit. Based on this comparing process, either group of bases are discarded from the memory as the reduction of wave packets.

3 Quantum-Circuit Processor

The matrix calculation is directly mapped to the hardware. The chip consists of processor elements (PEs), a banyan-like switch network for inter-PE communication and a broadcasting network for instruction and matrix coefficients. The PEs are configured at terminals of a recursive H tree network as shown in Fig. 2. At the branch point of H tree, a switch matrix is placed. Each PE executes the complex multiply and accumulate operation with two adders, four multipliers and complex registers. The behavior of switches for inter-PE communication is provided by the target qubit location and the behavior of switches for coefficient broadcasting is also provided by the target and control qubits location. It is noted that this hardware architecture is suitable for scaling due to recursive structure. Namely, one can realize the n-qubit processor in completely recursive structure by using 2^n PEs. The instruction set is summarized in Table 1. θ, ϕ_0 and ϕ_1 in eq.(1), (2) are decided in discrete value $\{\pi, \pi/2, \pi/4, \pi/8, \pi/16, \pi/32\}$ for simplicity. The observation process mentioned above is realized by executing PROB, PSUM and REDUCE sequentially. Owing to these three instructions, observation process is completed in the polynomial time in the number of qubits because probability summation is computed in steps of order $O(n)$.

The proposed architecture was implemented in a single PLD. Its spec is summarized in Table 2. The 5-qubit quantum Fourier transform was demonstrated in practice. The main part of demonstrated program list is shown in Fig. 3. For instance, the input state $(|0\rangle+|1\rangle)\otimes(|0\rangle+|1\rangle)\otimes(|0\rangle+|1\rangle)\otimes(|0\rangle+|1\rangle)\otimes|0\rangle$ was transformed to $(|0\rangle+|1\rangle)\otimes|0\rangle\otimes|0\rangle\otimes|0\rangle\otimes|0\rangle$ by the quantum Fourier transform, which was consistent with theory.

4 Conclusion

The simulator engine of quantum circuits was fabricated and quantum algorithm was demonstrated. In this chip, the operational speed seems dominated by the signal propagation in the network. This problem will be solved by a buffer insertion in long interconnections between PEs. The processor architecture is suitable to be

* Also a CREST Research director, Japan Science and Technology Corporation.

extended for larger numbers of qubits keeping the computational time within polynomial growth at the cost of the increase of numbers of computing device.

Acknowledgments

This work was supported by Japan Science and Technology Corporation (JST) in the program of Core Research for Evolutional Science and Technology (CREST).

References

[1] D. Deutsch, A. Barenco and A. Ekert, Proc. R. Soc. London Ser. A **449**, 669 (1995).

[2] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shore, T. Sleator, J. A. Smolin and H. Weinfurter, Phys. Rev. A **52**(5), 52 (1995).

[3] P. W. Shor, SIAM J. Computing **26**, 1484 (1997).

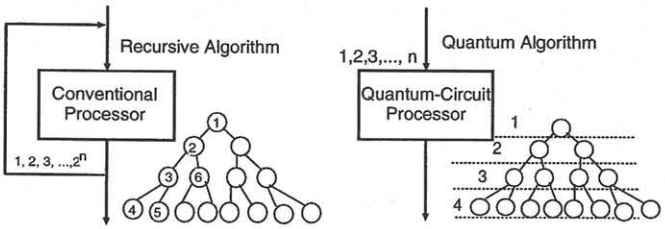

Fig. 1 Concept of quantum-circuit processor with a quantum algorithm. Parallel searching in the polynomial time is possible on our processor while conventional processors execute only recursive algorithm in the exponential time.

Table 1 Summary of Instructions and Their Function

Mnemonic	Function
PHAS(q_t, d, ϕ_d)	Phase shift of eq.(2) on the digit d of the qubit q_t.
cPHAS(q_c, q_t, d, ϕ_d)	Phase shift controlled by the qubits q_c.
ROT(q_t, θ)	Rotation of eq.(1) on the qubit q_t.
cROT(q_c, q_t, θ)	Rotation controlled by the qubits q_c.
PROB	Computation of the observing probability of single basis.
PSUM(q_t)	Summing probability register values of which only q_t-th digit is different.
REDUCE(q_t, mode)	Comparing probability register value and reducing states in 3 modes.
INIT	Initialization to $\|00\ldots0\rangle$.
HALT	Halt instruction.

Table 2 Specification of the Fabricated Chip

Type of PLD	Altera EP20K1500EFC33-3
Number of Qubits	5
Number of Gates	1.1×10^6
Clock Frequency	15 MHz

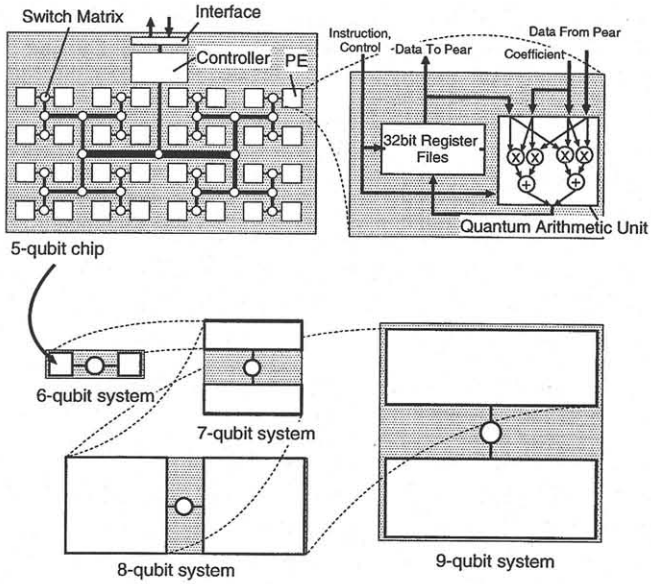

Fig. 2 5-qubit chip architecture and a concept of system extension method using this chip. For instance, 10-qubit system is composed of 32 chips in the perfect recursive structure.

```
/*qunatum FT*/
PHAS(4, 0, pi)
ROT(4, pi/2)
ROT(4, pi/4)
cPHAS(4, 3, 1, pi/2)
PHAS(3, 0, pi)
ROT(3, pi/2)
ROT(3, pi/4)
cPHAS(4, 2, 1, pi/4)
cPHAS(3, 2, 1, pi/2)
PHAS(2, 0, pi)
ROT(2, pi/2)
ROT(2, pi/4)
cPHAS(4, 1, 1, pi/8)
cPHAS(3, 1, 1, pi/4)
cPHAS(2, 1, 1, pi/2)
PHAS(1, 0, pi)
ROT(1, pi/2)
ROT(1, pi/4)
cPHAS(4, 0, 1, pi/16)
cPHAS(3, 0, 1, pi/8)
cPHAS(2, 0, 1, pi/4)
cPHAS(1, 0, 1, pi/2)
PHAS(0, 0, pi)
ROT(0, pi/2)
ROT(0, pi/4)
/*SWAP*/
cROT(4, 0, pi/2)
cPHAS(4, 0, 1, pi)
cROT(0, 4, pi/2)
cPHAS(0, 4, 1, pi)
cROT(4, 0, pi/2)
cPHAS(4, 0, 1, pi)
cROT(3, 1, pi/2)
cPHAS(3, 1, 1, pi)
cROT(1, 3, pi/2)
cPHAS(1, 3, 1, pi)
cROT(3, 1, pi/2)
cPHAS(3, 1, 1, pi)
```

Fig. 3 Programming instance. The quantum Fourier transform is executed taking 25 instruction steps.