FPGA-Based High-Speed Emulator of Quantum Computing

Minoru Fujishima
School of Frontier Sciences, The University of Tokyo
fujf@k.u-tokyo.ac.jp

Abstract

Quantum computers are believed to perform high-speed calculations, compared with conventional computers. However, the quantum computer has inherent issues. Firstly, it solves NP (non-deterministic polynomial) problems at a high speed only when a periodic function can be used in the process of calculation. Secondly, it is extremely difficult to increase the problem scale to be solved, which is determined by the number of quantum bits. To overcome the restrictions stemming from the quantum algorithm, we are studying the emulation by a FPGA (field programmable gate array). In this paper, first, it is explained why a periodic function is required for the algorithm of a quantum computer. Then, it is shown that the hardware emulator can solve NP problems at a high speed without using a periodic function.

1. Introduction

The quantum computer is attracting the attention of various researchers since it can rapidly solve problems which require much calculation time using a conventional computer. In particular, it is expected that a quantum computer may be able to solve NP problem at a high speed. Currently, however, Shor’s algorithm [1] is the only one that can solve an NP problem in polynomial time using the quantum computer. We have investigated the essence of the restrictions inherent in the quantum computer, and studied the feasibility of overcoming the limitations in the case of the quantum computer by an emulator without using quantum mechanics. In this paper, an efficient emulation using FPGA will be proposed in the following sections.

2. High-speed solution of NP problems

A problem, in which the polynomial-time algorithm using the deterministic Turing machine is not obtained with increasing the scale of the question, is called NP problem. In other words, the calculation time of NP problem increases exponentially with increasing scale of the question when the conventional computer is used. For example, factorization and the traveling-salesperson’s problem are considered as NP problems, as shown in Figure 1. Here, assume that the output $y = f(x)$ can be calculated in polynomial time according to the scale of an input $x$. In the case of NP problems, the input $x$ for the output $y$ has to be calculated, and it is not necessarily easy to obtain. The factorization problem can be expressed in this way, with $y = f(x) = N \mod x$; the problem is then to find a value of $x$ other than 1 that gives $y = 0$. To obtain $y$ from $x$, it is only necessary to divide once. On the contrary, in order to obtain $x$ for given $y = 0$, the operation time increases exponentially according to the scale of $N$. In the case of a traveling salesperson’s problem, one has to obtain $x_1, x_2, \ldots$ giving the minimum of $y = \sum x_i$, where $x_i, x_2, \ldots$ are route vectors. Although the total distance $y$ is easy to find if route $x_1, x_2, \ldots$ vectors are given, a group of $x_1, x_2, \ldots$ making $y$ the minimum is not easy to find. The quantum computer was considered to provide a solution at a high speed to such a problem [2].

General procedures to solve NP problem in polynomial time are summarized in Figure 2. Here, $f^{-1}(y)$ is not calculated directly because $f^{-1}(y)$ is not obtained analytically or requires exponential time to calculate. Consequently, another strategy is applied. At first, the table of the entire input $x$ and corresponding output $y$ is created. Its calculation is completed in polynomial time if this calculation is performed for the entire $x$ in parallel. Then, the target $y$ is searched and the corresponding $x$ is given as the answer. NP problem can be solved in polynomial time as a whole if the time to search also ends in polynomial time.
3. Solving NP problem using a quantum computer

3.1. Search issue

The issue is how to realize the high-speed solution for NP problem using a quantum computer. That the quantum computer executes calculations in parallel using quantum superposition is well known. To create the input-and-output table for the given function \( f \), conventional computers using the scalar processor have to perform calculations one by one. Therefore, the creation of the table requires exponential time. On the other hand, since outputs are simultaneously calculated from the group of all inputs in the case of a quantum computer, the calculation ends in polynomial time. As compared with the conventional computer, the quantum computer requires a much shorter calculation time. However, NP problem cannot be solved in polynomial time solely by performing parallel operation in polynomial time.

Quantum Fourier transform (QFT)
- Only applied to periodic function
- The solution is found in polynomial time

Grover’s algorithm
- Any function can be applied
- The solution is found in the order of \( \sqrt{N} \); not polynomial

Figure 3. Two algorithms for the search using the quantum computer.

To solve NP problem in polynomial time, it is also necessary to complete the search in polynomial time. Two kinds of search methods are proposed using quantum computing as shown in Figure 3. One is the quantum Fourier transform (QFT) [1]. It is known for the quantum computer that Fourier transform can be completed in polynomial time. If the required information is encoded in the period of the outputs of the table calculated beforehand, information can be acquired in polynomial time. Namely, in order to apply the QFT, the outputs of the table need to be periodic; otherwise, Grover’s algorithm is used [3]. Since Grover’s algorithm can search for a specific value from the input-and-output table, the outputs do not need to be periodic. However, it is known that the calculation time increases proportionally to \( N \) when using Grover’s algorithm, where the scale of the table is \( N \). Since the search time is not proportional to \( \log N \) the search cannot be completed in polynomial time. Therefore, finding an appropriate periodic function becomes the key to solving NP problem in polynomial time.

Shor’s algorithm is famous as the algorithm which can solve factorization in polynomial time. It is considered to solve an NP problem in polynomial time since factorization is one of NP problems. The key of Shor’s algorithm is not to calculate the residue directly, but to calculate a substitute function in which the information about the factor is encoded in a period. It is noted that it is the only algorithm which can solve an NP problem in polynomial time. This implies that it is difficult to find the substitute function encoding the required information in the period.

The polynomial-time solution using the quantum computer for other NP problems than the factorization has not yet been found. Here, a satisfiability (SAT)
problem is taken for instance. The SAT problem is the problem which finds the group of variables satisfying a given binary equation. If a quantum computer is used, the input-and-output table can be created in polynomial time using the quantum superposition. However, since the created table is not periodic, only Grover’s algorithm can be used. Since the search is not completed in polynomial time, a quantum computer cannot solve a SAT problem in polynomial time.

![Figure 4](image4.png)

**Figure 4.** Separate qubits are required for input and output registers in a quantum computer.

### 3.2. Quantum-state issue

Many quantum bits (qubits) are required in quantum computing for practical algorithms. Taking the function \( b = f(a) \) as an example, in a quantum computer, separate qubits must be assigned for input “a” and output “b”, because all outputs must correspond with inputs, and the same qubit cannot be used for an output and an input. The assignment of qubits to realize the function with 2-bit input and 2-bit output using a quantum computer is shown in Figure 4. In this case, 4 qubits are required for the input and the output. Similarly, during calculation, separate qubits are required according to the number of temporary variables when temporary variables are necessary in a quantum computer. It must be noted that the realization of quantum computing with large-scale qubits is difficult not only with physical quantum-mechanical processes but also with software simulation.

![Figure 5](image5.png)

**Figure 5.** Conceptual view of the binary search.

### 4. Efficient emulation of NP problems using FPGA

#### 4.1. Search algorithm

We have studied the hardware emulator to solve the problems in search of the quantum computer [4]. Parallel operation can be performed using many devices in the FPGA. Furthermore, binary search in the FPGA can complete the operation in polynomial time even if the function is not periodic, as shown in Figure 5. Therefore, NP problems, where even the quantum computer takes long computation time, can be finished in polynomial time provided the input-and-output table is generated in polynomial time.

![Figure 6](image6.png)

**Figure 6.** The process of solving a problem using a quantum computer can be divided into three stages.

In order to realize a quantum computer for solving practical problems, development of software as well as hardware is necessary, although qubit expansion using physical quantum-mechanical processes has been a focus for research on quantum computing. Moreover, in order to develop software efficiently, a system which can emulate large-scale problems at high speed is required.

---

1 The SAT problem has to find the input corresponding the final output of 0. If the answer could be searched with methods other than Grover’s algorithm at high speed, a quantum computer would solve a SAT problem at high speed.
initialize, equal probability amplitudes are distributed to the entire inputs. Then, the input-and-output table is created in polynomial time by performing parallel operation. Finally, information to obtain the target output is extracted and the corresponding input becomes the answer. The probability distribution and the parallel operation in these three stages are emulated using FPGA without any cost with respect to operation time, since they are completed in polynomial time in all cases using the quantum computer. However, the FPGA does not emulate the extraction of the last answer to overcome the restrictions on the function using the quantum computer.

4.2. Algorithmic optimization

4.2.1. Logic quantum processor. For general quantum algorithms, operations proceed as follows. In the first stage, the probabilities are distributed equally to the quantum states which correspond to the candidates for the answer. In the second stage, these probabilities are exchanged between quantum states in order to execute the calculations required for the given application. Until the second stage, only two probabilities appear, which are 0 and 1/m, when the number of candidates for the answer is m. As a result, binary code can be used to represent the quantum states although conventional quantum computing simulation requires complex numbers in order to represent them. Therefore, we have proposed a logic quantum processor (LQP) that expresses probabilities using binary code [4]. It is noted that the quantum command is described by the unitary matrix in principle, and that the sum-of-products operation of a complex number is required to support arbitrary transformation [5][6]. Required hardware resources such as processing elements and memory can be considerably reduced compared with the resources required for the conventional quantum computing emulator.

4.2.2. Quantum index processor. When the location of “1” in the quantum states is used instead of the values of the quantum states, memory usage can be reduced further since most quantum states stored in memories remain “0” in the LQP. The location of “1” in the quantum states is called the quantum index, hereafter. The processor holding only quantum indexes instead of quantum states is named a quantum index processor (QIP) [8]. Figure 7 shows a comparison of memory usage in the LQP and the QIP proposed in this paper. Internal data transitions in the processor are compared in Figure 8 when quantum operations are executed. As shown in this figure, while probabilities in the quantum states are exchanged in each quantum operation in the LQP, only the index numbers are changed in the QIP.

Figure 7. Comparison of the expressions of quantum states using the LQP and the QIP

Figure 8. Comparison of internal data transitions when NOT is performed in the LSB.
Figure 9. Block diagram and specification of the QIP

5. Measurements and discussions

The block diagram of the QIP is shown in Figure 9. The QIP has 2048 processing elements (PEs) and emulates massive parallel computing in the quantum computer. Each PE handles one candidate for the answer and executes the calculations using quantum indexes which correspond to the quantum operations. Each PE has a logic unit (LU), a memory block and a temporary register. 64-qubit index data are stored in the memory. The QIP can also simulate the effect of quantum bit errors [7] at high speed by generating quantum NOTs stochastically.

We have implemented the QIP on a field programmable gate array (FPGA) with 1.5 million gates. 2048 (=2^11) PEs are arrayed on one chip, which corresponds to 11 qubits. Each PE has 64 quantum index bits, which correspond to 64 qubits. In total, our proposed system realizes quantum operation with 75 qubits. The screenshot of the software and the photograph of the hardware of the QIP are shown in Figure 10. Upon implementation, the clock frequency of the QIP is 80 MHz. A comparison between computation time of the LQP and the QIP is shown in Figure 11, where Shor’s factoring algorithms are used. It is noted that the computation time of the LQP is predicted value by simulation instead of experimental measurement.

Figure 10. Screenshot of the software and photograph of the hardware of the QIP

The FPGA emulator can also execute the SAT algorithm. The calculation time for the SAT problem using the emulator is proportional to the number of parentheses in a formula, and is not related to the number of variables. On the other hand, it is known that the calculation time increases exponentially with the number of variables when using the conventional scalar processor. The SAT cannot be solved in polynomial time by the quantum computer either, as stated previously.

The algorithm using the emulator for the SAT problem is shown in Figure 12. Assume that the following equation is given for the SAT problem:

\[ \left( a + c + d \right) \left( b + e \right) = \left( a + d + f + h \right) \cdots \left( k + g \right). \]  \( \text{(1)} \)

Then, negate both sides of the SAT equation and set the right-hand side of the equation to binary 0.

\[ \left( a \cdot c \cdot d \right) \left( b \cdot e \right) \left( a \cdot d \cdot f \cdot h \right) \cdots \left( k \cdot g \right) = 0. \]  \( \text{(2)} \)

First, the outputs corresponding to all variables are initialized to 0. The expression enclosed in a pair of parentheses is called a node, hereafter. Since the variable group generating 1 in each node is not a solution, it is excluded from the solution by incrementing the counter, which is in the lower part of the quantum circuit in Figure 12. This procedure continues for all nodes. Finally, the input corresponding to the output with 0 is investigated with a binary search, and an answer will be obtained. As a result, the solution is obtained in polynomial time. The counter is used in order not to return the output to 0.
even if the group of a specific variable is 1 in two or more nodes. If an irreversible command is used to set the specific quantum bit to 0, one quantum bit can hold the judgment result for the solution instead of a counter. Thus, in the fabricated emulator, not only the command for performing the emulation of a quantum computer, but also irreversible commands called SET0 and SET1, are supported.

\[(a + c + d)(b + e)(a + d + f + h) \cdots (k + g) = 1\]

**Figure 12. Quantum circuit of the SAT problem.**

**Table 1. Comparison of the quantum computer and FPGA emulator**

<table>
<thead>
<tr>
<th></th>
<th>Quantum Computer</th>
<th>FPGA Emulator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware</td>
<td>Polynomial</td>
<td>Exponential</td>
</tr>
<tr>
<td>(with error correction)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel Computing</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Search</td>
<td>Polynomial (periodic)</td>
<td>Exponential (otherwise)</td>
</tr>
<tr>
<td></td>
<td>Polynomial</td>
<td></td>
</tr>
</tbody>
</table>

**6. Summary**

The comparison between the quantum computer and the FPGA emulator is summarized in Table 1. The quantum computer is attractive from the viewpoint that hardware requirements increase in polynomial with increasing scale of the question. The quantum computer and the FPGA emulator can perform parallel operations to create an input-and-output table. In the case of the quantum computer, the QFT cannot be used and search cannot be completed in polynomial time if the output of the table is not periodic. It is a severe restriction if a periodic function must be found to solve NP problem in polynomial time using the quantum computer. A breakthrough is required for the search algorithm in the quantum computer. On the other hand, in the case of the FPGA emulator, searching can be completed in polynomial time independently of the periodicity of the output of an input-and-output table. As a result, NP problems which even the quantum computer takes computation time can be solved at high speed if the FPGA emulator, which does not have restrictions in the algorithm, is used.

Moreover, to cope with the large scale problem, the system for emulating large-scale quantum computing was proposed by calculating the quantum index during the quantum operation. As a result of hardware implementation with an FPGA, the system realized the emulation of a 75-qubit quantum algorithm. It is found that the computation time of the QIP is $10^{18}$ times faster than that of the conventional emulator. Therefore, the proposed system will be a powerful tool for the development of quantum algorithms.

**Acknowledgement**

The “Core Research for Evolutional Science and Technology” of the Japan Science and Technology Corporation has supported this study.


