An Architectural Framework For Quantum Algorithms Processing Unit (QAPU)

Mohammad Reza Soltan Aghaei, Zuriati Ahmad Zukarnain, Ali Mamat, and Hishamuddin Zainuddin

Abstract- The focus of this study is developing a framework of Quantum Algorithm Processing Unit (QAPU). The framework shows a general plan for the architecture of quantum processor that is able to run quantum algorithms. The framework is used to increase the implementation performance of quantum algorithms and design Quantum Processing Unit (QPU). QAPU can be applied as a quantum node to design quantum multicomputer. At first, the hybrid architecture is designed for the quantum algorithms. Then, the relationships between the classical and quantum parts of the hybrid algorithms are extracted and the main stages of the hybrid architecture are determined. Next, the framework of QAPU is designed and developed. Some gates and connections are projected in the framework that can be applied for future quantum algorithms. Furthermore, the framework is implemented and simulated for the existing quantum algorithms on a classic computer. It is shown that the framework is appropriate for the quantum algorithms.

Keywords: Computer Systems, Computer Architecture, Quantum Algorithm, Quantum Computing, Multi Computer.

I. INTRODUCTION

A quantum computer is a device that takes advantage of quantum mechanical effects to perform certain computations faster than a purely classical machine does. A quantum computer operates by manipulating those quantum bits or qubits with a fixed sequence of logic gates and performs it exponentially. The theory of quantum complexity determines when quantum computers may offer a computational speed-up over classical computers. At present, there are only a few general well-known techniques in the field of quantum computing and finding the problems that are amenable to quantum speedups is a high priority.

M.R. Soltan Aghaei is faculty member in Dep. of Computer Eng., Islamic Azad University of Khorasgan, Isfahan, Iran, and PhD candidate in Faculty of Computer Science and Information Technology, University Putra Malaysia 43400 UPM Serdang, Selangor, Malaysia (e-mail: msoltanaghaii@yahoo.com).

Zuriati A. Z. is with Dep. Of Communication Technology and Network, Faculty of Computer Science and Information Technology, University Putra Malaysia 43400 UPM Serdang, Selangor, Malaysia, www.fsktm.upm.edu.my (e-mail: zuriati@fsktm.upm.edu.my).

Ali Mamat is with Faculty of Computer Science and Information Technology, University Putra Malaysia 43400 UPM Serdang, Selangor, Malaysia, (e-mail: ali@fsktm.upm.edu.my).

Hishamuddin Z. is with Laboratory of Computational Sciences and Informatics, Institute for Mathematical Research, Universiti Putra Malaysia (UPM), 43400 UPM Serdang, Selangor, Malaysia (e-mail: hisham@fsas.upm.edu.my).

The Quantum Processing Unit (QPU) is the processor of quantum computer that is able to do quantum computations. A typical component in QPU is a quantum device that runs quantum algorithms; namely Quantum Algorithm Processing Unit (QAPU). This device can also be applied as a quantum node in quantum multicomputer. A quantum multicomputer is a distributed system that is composed of quantum computers connected through a quantum network. The nodes can be connected by creating an entangled state among them as well. The focus of this study is developing a framework of QAPU. QAPU can be used as a quantum node to design quantum multicomputer. At first, it was necessary to design the hybrid architecture for the quantum algorithms. This, however, required the analysis of the existing quantum algorithms as presented in the next section. Meanwhile, the relationship between the classical and quantum parts of the hybrid algorithms, and the main stages of the hybrid algorithm were determined. Furthermore, the framework was implemented and simulated for the existing quantum algorithms on a classic computer.

II. RELATED WORKS

Nowadays, many researchers focus on designing a quantum computer by implementing quantum algorithms. There are a few efficient quantum algorithms; David Deutsch and Richard Jozsa showed an algorithm that could be run in poly-log time on a quantum computer, but required linear time on a deterministic Turing machine [1]. This may have been the first example of a quantum computer being shown to be exponentially faster than a deterministic Turing machine. Unfortunately, for the quantum computer, the problem could also be solved in poly-log time in a probabilistic Turing machine, a Turing machine which is capable of making a random choice.

In 1994 Peter W. Shor showed quantum factoring algorithm is a polynomial time algorithm for prime factorization and discrete Logarithms on a quantum computer [2]. All known algorithms for factoring an *n*-bit number on a classical computer take time proportional to $O(2^n)$ time and in the best known algorithm to $O(\exp(n^{1/3}))$ time. But Shor's algorithm for factoring on a quantum computer takes time proportional to $O(n^2 \log(n))$.

Lov Grover in 1996 discovered the quantum search algorithm. It is used to solve NP-hard problem [3]. Grover's search algorithm can search an unstructured space of N possibilities in $O(\sqrt{N})$ time, and the classical computer can do search on average in $O(N)$. It is sometimes referred

ISBN: 978-988-17012-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

to as amplitude amplification and has been found to be useful for quantum counting, and as a wrapper for other algorithms [4, 5]. The speedup of Grover's algorithm is achieved by exploiting both quantum parallelism and the fact that in quantum theory a probability is the square of the probability amplitude. Bennett and co-workers [6] and Zalka [7] showed that Grover's algorithm is optimal. No classical or quantum algorithm can solve this problem faster than a time of order.

Furthermore, other algorithms are Simon's algorithm to finds the hidden string [8], Hallgren's algorithm to solve the Pell's equation [9], or the topic of quantum random walks [10, 11]. Nonetheless, efficient quantum algorithms are very limited in number and scope; no real breakthrough has yet been achieved in physical implementations. Most importantly, these algorithms are not still matured adequately to be applied in real quantum computations.

The main objective of this study was to design and analyze a framework for the processor architecture of a quantum computer which executes the quantum algorithms. At first, the quantum algorithms were analyzed to find the hybrid architecture and a framework for the quantum algorithms. Meanwhile, the hybrid architecture was designed for the quantum algorithms. Furthermore, the relationship between the classical and quantum parts of the hybrid algorithms, and the main stages of the hybrid algorithm were determined as presented in next section. Finally the designed framework was implemented and simulated for the existing quantum algorithms on a classic computer.

III. HYBRID ARCHITECTURE

In its simplest form, a quantum algorithm consists of a unitary transformation and a subsequent measurement of the resulting state. For the traditional computational tasks which include searching or mathematical calculations, efficient quantum implementations often have the form of probabilistic algorithms.

The quantum algorithms such as Shor's algorithm consist of two parts. The first part is a classical algorithm which can be run on a classical computer, while the second part is the quantum algorithms that can be run on a quantum computer or simulated on a classical computer. Figure 1 shows the relationship between the classical part and the quantum part of the hybrid architecture [12-15].

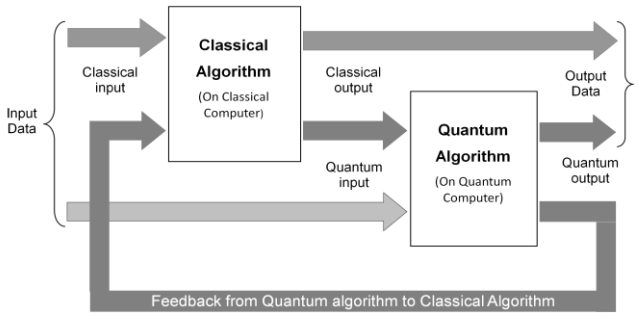

Figure 1. The relationship between the classical and quantum parts of the hybrid architecture.

The quantum algorithms are also known as the hybrid algorithms that consist of both the classical and quantum components. Moreover, the quantum portion of many algorithms is probabilistic; often need multiple runs to get the desired result. The main stages of the hybrid architecture can be done as follows:

1. Pre-calculate certain classical factors (initialize and run the classical part of the algorithm).
2. Run the quantum algorithm on the quantum circuit:
   a) Initialize the quantum node (Initialize quantum circuit and define all gates, switches and unitary function).
   b) Prepare inputs state (store inputs on target and control registers).
   c) Execute the quantum portion of the algorithm (apply gates and unitary transformation on the input data).
   d) Measure the output of Machine State (measure the output registers of the quantum circuit).
   e) Evaluate Measurement (If the desired result is retrieved, then the post-processing in step 3 is done).
   f) Exit if the desired result is obtained (If a solution is found, then exit from the quantum circuit, or else step 2 is repeated).
3. Finish post-processing (run the second classical part of the algorithm).

Steps 1 and 3 can be executed on a classical computer, while step 2 can be executed on a quantum computer using the quantum circuit. Measuring and evaluating of the quantum circuit can be done on a classical computer through a simulation work. The quantum circuit can also be simulated on a classical computer. The diagram illustrated in Figure 2 indicates the development of a general plan for the hybrid algorithms simulated on a classical computer.

IV. THE FRAMEWORK OF QAPU

In next of this study, the designed framework of QAPU is presented. This framework is shown in Figure 3. Some gates and connections are projected in the framework that can be applied for future quantum algorithms. Furthermore, the framework is setup, implemented and simulated for the existing quantum algorithms on a classic computer.

In this quantum circuit, there are two inputs $|x\rangle$ and $|y\rangle$. The inputs data require two registers that first register is used to store $|x\rangle$ named control register and second register is used to store $|y\rangle$ named target register. First register $|x\rangle$ is applied on the inputs of gate G_c and second register $|y\rangle$ is applied on the inputs of gate G_t. An n-qubit quantum register can be in a superposition of all possible 2^n states $|0\rangle$ to $|2^n - 1\rangle$ at the same time. This effect allows a quantum computer to calculate a function on all possible inputs at the same time, in a single pass. Therefore, the regular functions for G_c and G_t are Hadamard Transform and Quantum Furrier Transform (QFT). The most of the quantum algorithms is used to same functions for gate G'_c, therefore if G_c is QFT, then G'_c is QFT^{-1}.

ISBN: 978-988-17012-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

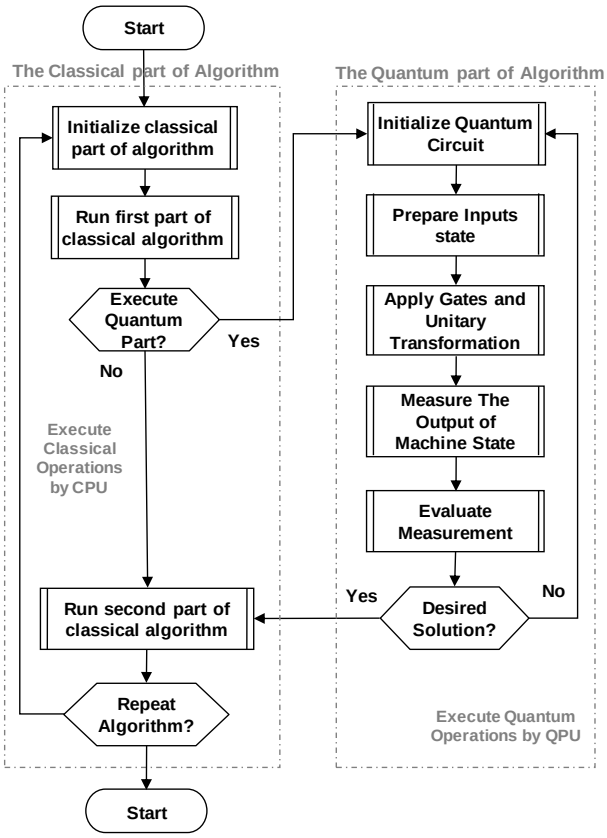

Figure 2. The Classical and Quantum parts of the Hybrid Architecture.

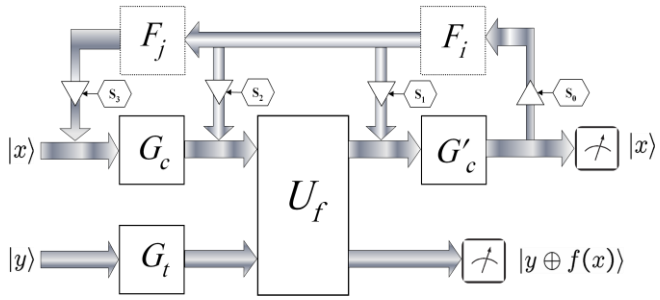

Figure 3. The framework of QAPU.

The common part in all quantum algorithms is the black box or oracle function U_f that is often used to model a subroutine of calculations and is reversible. Classically, a black-box function can be simply thought of as a box that evaluates an unknown function *f*. The input is some *n*-bit string $|x\rangle$ and the output is given by an *m*-bit string $f(x)$. In quantum, such a box can only exist if it is reversible. To create a reversible box, the input $|x\rangle$ is output together with $f(x)$. To make the box reversible, an additional *m*-bit input $|y\rangle$ is added and the output of the result is $|y \oplus f(x)\rangle$ where $\oplus$ denotes bitwise addition modulo 2. In particular, if $|y\rangle$ is fixed to be $y = 0 \ldots 0$, the output is $f(x)$. Note that U_f now induces a transformation on *n*+*m*-bit strings that can be described by a permutation of the 2^{n+m} possible strings; in particular it is unitary.

In some of the quantum algorithms, the execution of some functions or operators is repeated and there is a feedback that makes iteration. For example, in the Grover's algorithm, the operator $G = HU_{0^\perp}HU_f$ that can be applied by the following sequence of transformations and the feedback is iterated. In the framework, this feedback is implemented with switches S_0, S_1, S_2 and S_3. In the Grover's algorithm, the switches S_0 and S_2 are closed for $O(\sqrt{N})$ time, while S_1 and S_3 are always opened. On the feedback, the operators F_i and F_j are prepared for future quantum algorithms, but the existing quantum algorithms do not require these operators.

V. RESULT

In this section, the results gathered from the implementation and simulations of the framework for the existing quantum algorithms are presented. In the quantum algorithm circuit, two *n*-qubit inputs $|x\rangle$ and $|y\rangle$ are defined. The inputs data require two registers that the control register is used to store $|x\rangle$ and the target register is used to store $|y\rangle$. Furthermore, the gates G_c , G'_c , and G_t are defined for each algorithm. First, the register $|x\rangle$ is applied on the inputs of gate G_c and the second register $|y\rangle$ is applied on the inputs of gate G_t. The usual functions for G_c and G_t are Hadamard Transform and Quantum Furrier Transform (QFT). Some of the quantum algorithms are used to same functions for gate G'_c . For example, both of them are Hadamard gates or if G_c is QFT, then G'_c is QFT^{-1}. In some of the quantum algorithms, the execution of some functions or operators is repeated and there is a feedback that makes the iteration. In the framework, this feedback is implemented with switches S_0, S_1, S_2 and S_3. On the feedback, operators F_i and F_j are prepared for the future quantum algorithms, but the existing quantum algorithms do not require these operators. Table 1 shows the values of the inputs, gates and position of switches in the quantum algorithms.

The initialization and setup of the existing quantum algorithms have been demonstrated on the proposed framework. In the next, the setting up of the framework for the existing quantum algorithms is explained.

A. *Setting up the framework for Deutsch Algorithm*

Both inputs in the Deutsch's algorithm are 1-qubit. The first input |x⟩ is initialized to |0⟩ and the second input |y⟩ with |1⟩. Gates G_c , G'_c and G_t are replaced with the 1-qubit Hadamard gates. Any feedback does not require in this algorithm and the switches S_0, S_1, S_2 and S_3 are opened. The initialized framework for the Deutsch algorithm shows in Figure 4.

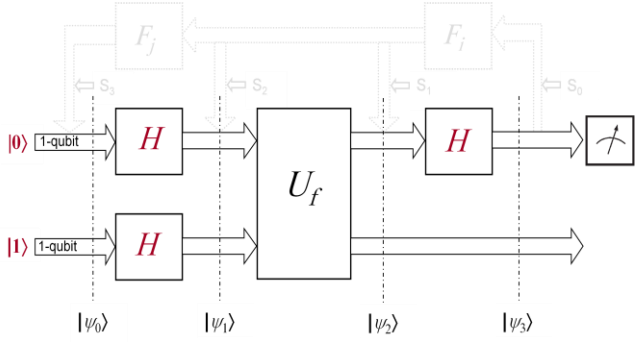

Figure 4. Setting up the framework for Deutsch algorithm.

ISBN: 978-988-17012-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

B. Setting up the framework for Deutsch-Jozsa Algorithm

The framework is initialized for Deutsch-Jozsa algorithm similar to Deutsch's algorithm. The first input $|x\rangle$ in this algorithm is n qubits and second input $|y\rangle$ is 1-qubit. The input $|x\rangle$ is initialized with $|0\rangle^{\otimes n}$ and the input $|y\rangle$ with $|1\rangle$. Any feedback does not require and the switches S_0, S_1, S_2 and S_3 are opened. The gates G_c *and* G_c' are replaced with the n-qubit Hadamard gates and G_t with the 1-qubit Hadamard gate. The initialized framework for Deutsch-Jozsa algorithm shows in Figure 5.

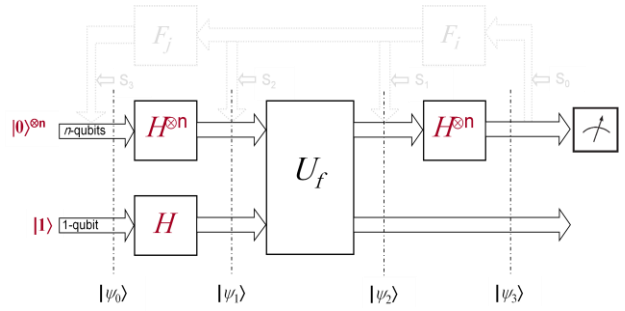

Figure 5. Setting up the framework for Deutsch-Jozsa algorithm.

C. Setting up the framework for Simon's Algorithm

The inputs in the Simon's algorithm are n-qubit. The inputs $|x\rangle$ and $|y\rangle$ are initialized with the same value $|0\rangle^{\otimes n}$. The gates G_c *and* G_c' are replaced with n-qubit Hadamard gates, but the gate G_t will be omitted on the framework, and this means the second input will be applied to black bock without any process. A feedback is needed to ensure the iterate sequence transformations of $H^{\otimes n}U_f H^{\otimes n}$ for n times. The switches S_0 and S_3 will be closed (ON) and S_1 *and* S_2 will be opened (OFF). The condition of $i<n$-1 applies to the switch S_0 to provide the feedback is connected. The switch S_3 is always closed without any condition. When S_0 is opened, then the feedback will be disconnected. The framework for Simon's algorithm shows in Figure 6.

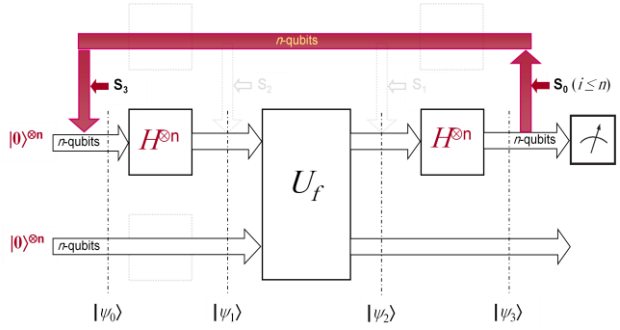

Figure 6. Setting up the framework for Simon's algorithm.

D. Setting up the framework for Eigenvalue Estimation Algorithm

The main quantum gate on the Eigenvalue Estimation Algorithm is Quantum Fourier transform (QFT). QFT is useful for the problem of quantum phase estimation, and finding the period of periodic states. The phase estimation will be applied in order to estimate eigenvalues of unitary operators, and the eigenvalue estimation algorithm will be applied in order to derive the quantum factoring algorithm, and to solve the discrete logarithm problem. The hidden subgroup problem encompasses both the order finding and discrete logarithm problem as well as many others. The framework is initialized for the circuit of eigenvalue estimation as shown in Figure 7.

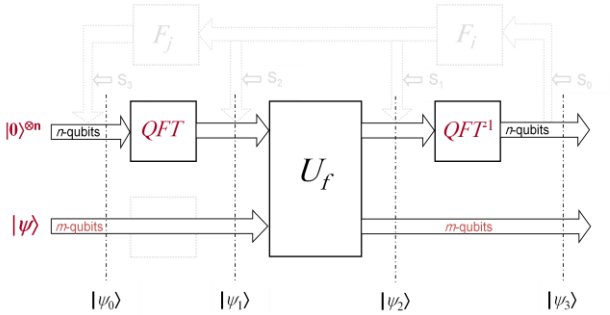

Figure 7. Setting up the framework for Eigenvalue Estimation algorithm.

As long as this algorithm does not require the feedback, the switches S_0, S_1, S_2 and S_3 are opened. An n-qubit input $|x\rangle$ is stored in the control register and an r-qubit $|y\rangle$ in the target register. As the quantum circuit of this algorithm, the n-qubit control register is initialized to $|0\rangle^{\otimes n}$ and the r-qubit target register to eigenstate $|\psi\rangle$. The gate G_c is replaced with n-qubit Quantum Fourier Transform (QFT) and the gate G_c' is replaced with n-qubit inverse of Quantum Fourier Transform (QFT^{-1}). The gate G_t will be omitted on the framework, and this means the eigenstate $|\psi\rangle$ will be applied to the black bock without any process.

E. Setting up the framework for finding orders problem for Shor's Algorithm

Shor's algorithm consists of two parts; first part is a reduction of the factoring problem to the problem of order-finding, which can be done on a classical computer. Second part is a quantum algorithm to solve the order-finding problem. The quantum part of algorithm is that in order to find a factor of a number, it is sufficient to solve a problem called period finding, the problem Shor's algorithm [2]. The quantum part circuit of Order-Finding problem is similar to the circuit of Eigenvalue Estimation algorithm, but the second input is replaced with value $|\psi\rangle=|\mathbf{1}\rangle=|00\ldots01\rangle$.

The period-finding method operates on two quantum registers, namely the control register and the function result register. In the end, the control register is actually measured in order to find the period of the function. Figure 8 shows how to initialize the framework for a quantum circuit for sampling estimates to a random integer multiple of $1/r$, which can be used to solve the order-finding problem. As long as this algorithm does not require the feedback, all the switches S_0, S_1, S_2 and S_3 are opened. An n-qubit input $|x\rangle$ is stored in the control register and an r-qubit $|y\rangle$ in the target register. As the quantum circuit of this algorithm, an n-qubit control register is initialized to $|0\rangle^{\otimes n}$ and an r-qubit target register to $|\mathbf{1}\rangle=|00\ldots01\rangle$. The gate G_c is replaced with the n-qubit Quantum Fourier Transform (QFT), while gate G_c' is replaced with the n-qubit inverse of the Quantum Fourier Transform (QFT^{-1}). The gate G_t will be omitted from the framework, and this means the second input will

ISBN: 978-988-17012-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

be applied to black bock without any process. The operator U_f is defined as c-U_a^x control.

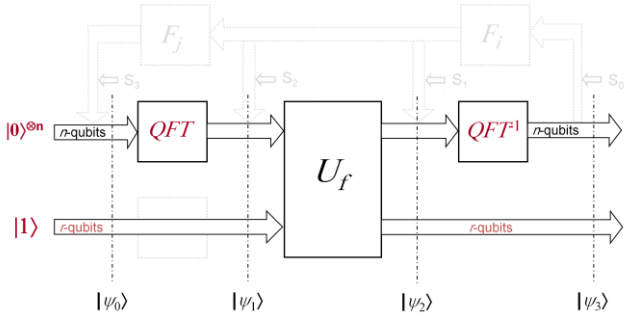

Figure 8. Setting up the framework for Shor's algorithm to solve the period finding problem.

F. *Setting up the framework for Grover's Algorithm*

The Grover's algorithm includes two different inputs, $|x\rangle$ and $|y\rangle$. The *n*-qubit input $|x\rangle$ is stored in the control register and the 1-qubit $|y\rangle$ in the target register. As the quantum circuit of this algorithm, the *n*-qubit control register is initialized to $|0\rangle^{\otimes n}$ and the 1-qubit target register to $|1\rangle$. The gate G_c is replaced with the *n*-qubit Hadamard gate, while G_t is replaced with the 1-qubit Hadamard gate. The gate G_c' is replaced with a sequence of operators $H^{\otimes n}U_{0^\perp}H^{\otimes n}$. Meanwhile, the Grover's algorithm iterates the operator $G = G_c' U_f = H^{\otimes n}U_{0^\perp}H^{\otimes n}U_f$ that is defined by the following sequence of transformations and known as the *Grover Iterate*. A feedback is needed to ensure the iteration of these sequence transformations of G for $O(\sqrt{N})$ times. This feedback can easily be implemented using switches S_0, S_1, S_2 and S_3. This can be done by closing S_0 and S_3, while leaving S_1 and S_2 open. The feedback is connected for $O(\sqrt{N})$ times that the algorithm is executed. The feedback loop can support the operators F_i and F_j for any future quantum algorithms. However, a direct feedback was employed in this design. The initialized framework for the Grover's algorithm is shown in Figure 9.

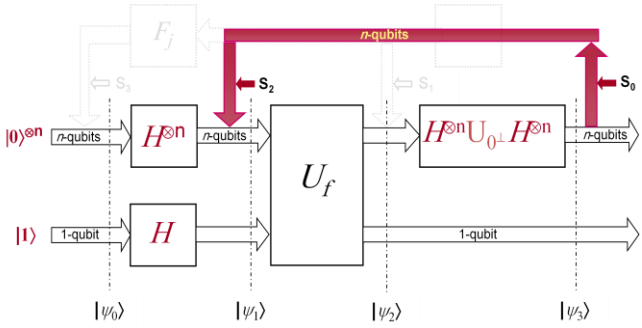

Figure 9. Setting up the framework for Grover's algorithm.

VI. CONCLUSION

This study was proposed the framework of a device which executes the quantum algorithms. This device can be applied as a unit in the quantum processing unit (QPU), namely Quantum Algorithm Processing Unit (QAPU). This device can also be applied as the quantum node in the quantum multicomputer. The quantum algorithms are known as hybrid algorithms that consist of classical and quantum components. Moreover, the quantum portion of many algorithms is probabilistic; often need multiple runs to get the desired result. Therefore, at first, the hybrid architecture was designed for the quantum algorithms. The relationship between the classical and quantum parts of the hybrid algorithms was then extracted. After that the main stages of the hybrid architecture algorithm were determined as shown in Figure 2. Next, the framework of QAPU was designed and developed. Some gates and connections were projected in the framework which can be applied for the future quantum algorithms as shown in Figure 3. Furthermore, the framework was setup, implemented and simulated for the existing quantum algorithms on a classic computer. It is shown that the framework is appropriate for the quantum algorithms. The framework is useful to design of quantum processor for the quantum computer.

Table 1. Setting up the framework for the quantum algorithms.

Gate	Deutsch Algorithm	Deutsch-Jozsa Algorithm	Simon's Algorithm	Shor's Algorithm	Grover's Algorithm
$\|x\rangle$	$\|0\rangle$ ***1***-qubit	$\|0\rangle^{\otimes n}$ ***n***-qubits	$\|0\rangle^{\otimes n}$ ***n***-qubits	$\|0\rangle^{\otimes n}$ ***n***-qubits	$\|0\rangle^{\otimes n}$ ***n***-qubits
$\|y\rangle$	$\|1\rangle$ ***1***-qubit	$\|1\rangle$ ***1***-qubit	$\|0\rangle^{\otimes n}$ ***n***-qubits	$\|1\rangle$=$\|00..1\rangle$ ***r***-qubits	$\|1\rangle$ ***1***-qubit
G_c	*H Hadamard Gate*	$H^{\otimes n}$ ***n***-qubits *Hadamard Gates*	$H^{\otimes n}$ ***n***-qubits *Hadamard Gates*	*QFT* ***n***-qubits *Quantum Furrier Tr.*	$H^{\otimes n}$ ***n***-qubits *Hadamard Gates*
G_c'	*H Hadamard Gate*	$H^{\otimes n}$ ***n***-qubits *Hadamard Gates*	$H^{\otimes n}$ ***n***-qubits *Hadamard Gates*	QFT^{-1} ***n***-qubits *Inverse QFT*	$H^{\otimes n}U_0{}^{\perp}H^{\otimes n}$ ***n***-qubits *Grover iterate*
G_t	*H Hadamard Gate*	*H Hadamard Gate*	*Null*	*Null*	*H* Hadamard Gate
S_0	*Open*	*Open*	*Close* with condition i ≤ n	*Open*	*Close for* $O(\sqrt{N})$ time, $N=2^n$
S_1	*Open*	*Open*	*Open*	*Open*	*Open*
S_2	*Open*	*Open*	*Open*	*Open*	*Close* for $O(\sqrt{N})$ time, $N=2^n$
S_3	*Open*	*Open*	*Close* with condition i ≤ n	*Open*	*Open*
F_i	*Null*	*Null*	*Null*	*Null*	*Null*
F_j	*Null*	*Null*	*Null*	*Null*	*Null*

ISBN: 978-988-17012-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

REFERENCE

[1] D. Deutsch and R. Jozsa, "Rapid Solution of Problems by Quantum Computation," *Proc. Royal Soc. London,* vol. 439, pp. 553-558, 1992.

[2] P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," in *35th Ann. Symp. Foundations of Computer Science* Los Alamitos, Calif.: IEEE Computer Society Press, 1994, pp. 124-134.

[3] L. K. Grover, "A Fast Quantum Mechanical Algorithm for Database Search," in *28th Annual ACM Symposium on the Theory of Computation* New York: ACM Press, 1996, pp. 212–219.

[4] L. K. Grover, "Fixed-point quantum search," *Physical Review Letters,* vol. 95, 2005.

[5] G. Brassard, P. Høyer, and A. Tapp, "Quantum counting," in *25th Int. Colloquium on Automata, Languages and Programming (ICALP'98)*: Springer, 1998, pp. 820–831.

[6] C. H. Bennett and others, "Strengths and Weaknesses of Quantum Computing," *SIAM J. Computing,* vol. 26, pp. 1510-1523, 2001.

[7] C. Zalka, "Grover's quantum searching algorithm is optimal," *Physical Review A,* vol. 60, pp. 2746-2751, 1999.

[8] D. Simon, "On the power of quantum computation," in *35th Ann. Symp. on Foundations Computer Science*: ACM, 1994, pp. 116–124.

[9] S. Hallgren, "Polynomial-time quantum algorithms for pell's equation and the principal ideal problem," in *34th ACM Symp. on Theory of Computing (STOC),* 2002, pp. 653-658.

[10] Y. Aharonov, L. Davidovich, and N. Zagury, "Quantum random walks," *Physical Review A,* vol. 48, pp. 1687–1690, 1993.

[11] J. Kempe, "Quantum random walks: an introductory overview," *Contemporary Physics,* vol. 44, pp. 307-327, 2003.

[12] M. R. Soltan Aghaei, Zuriati Ahmad Zukarnain, Ali Mamat, and H. Zainuddin, "A Quantum Algorithm for Minimal Spanning Tree," in *3rd Int. Sym. on Information Technology (ITsim08)* Malaysia: Proc. IEEE, 2008.

[13] M. R. Soltan Aghaei, Zuriati Ahmad Zukarnain, Ali Mamat, and H. Zainuddin, "A Hybrid Algorithm for the Shortest-Path Problem in the Graph," in *Int. Conf. on Advanced Computer Theory and Engineering (ICACTE08)* Phuket Island, Thailand: IEEE Computer Society, 2008.

[14] M. R. Soltan Aghaei, Zuriati Ahmad Zukarnain, Ali Mamat, and H. Zainuddin, "A Hybrid Algorithm for Finding Shortest Path in Network Routing," *Journal of Theoretical and Applied Information Technology,* vol. 5, 2009.

[15] M. R. Soltan Aghaei, Zuriati Ahmad Zukarnain, Ali Mamat, and H. Zainuddin, "A Hybrid Architecture Approach for Quantum Algorithms," *Journal of Computer Science,* vol. 5, pp. 725-731, 2009.

ISBN: 978-988-17012-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)