Quantum Multiplexer Designing and Optimization applying Genetic Algorithm

Debarka Mukhopadhyay[1], Amalendu Si[2]
[1]Dept of CSE ,Bengal Institute of Technology and Management, Santiniketan, West Bengal, India

[2]Dept of IT, Mallabhum Institute of Technology, Bankura, West Bengal, India

Abstract:
This paper shows how to design efficient quantum multiplexer circuit borrowed from classical computer design. The design will show that it is composed of some Toffole gates or C^2NOT gate and some two input CNOT gates. Every C^2NOT gate is synthesized and optimized by applying the genetic algorithm to get the best possible combination for the design of these gate circuits.

Keywords:
Qubit, Toffole gate, Quantum Multiplexer Circuit, Circuit Synthesis, Quantum Half adder Circuit,Genetic Algorithm.

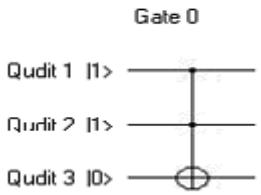
1. INTRODUCTION

Introduction: As the ever-shrinking transistor approaches atomic proportions, Moore's law must confront the small-scale granularity of the world: we can't build wires thinner than atoms. Theoretically, *quantum* computers could outperform their classical counterparts when solving certain discrete problems [15, 8].

The logical properties of qubits also differ significantly from those of classical bits[2]. Bits and their manipulation can be described using two constants (0 and 1) and the tools of Boolean algebra. Qubits [13], on the other hand, must be discussed in terms of vectors, matrices, and other linear algebraic constructions.
Quantum logic circuits [1,4,5,6,14], from a high level perspective, exhibit many similarities with their classical counterparts. They consist of quantum gates, connected by quantum wires which carry quantum bits. Moreover, logic synthesis for quantum circuits is as important as the classical case.
In this work, we focus on identifying useful quantum circuit blocks. we analyze quantum conditions and designing *quantum multiplexor* that engage CNOT[3] and Toffole gates. Here we synthesize and optimize each and every Toffole gate by applying genetic algorithm [7, 10, 11, 12].

2. Controlled Quantum Gate Operations:

Here we have engaged two controlled quantum gates i.e. C^2NOT and CNOT. C^2NOT is also known as Toffole gate. It is a three input gate. The first two inputs are the controlled inputs and the third one is the target input. This gate has a 3-bit input and output. If the first two bits are set, it flips the third bit. Following is a table over the input and output bits:



Input 1	Input 2	Input 3	Output 1	Output 2	Output 3
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

Fig1: C^2NOT gate and i/o table

3. Quantum Half Adder:

In order to construct an efficient Quantum multiplexer circuit we'll need the quantum equivalent to the classical "half adder". This is a 2-input, 2-output device with the following truth table:
Classical Half Adder.

Input		Output	
A	B	Sum	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

The sum output is the XOR of the two inputs, and the carry output is the AND of the same two inputs. The quantum equivalent is a 3-input, 3-output device with the following truth table and equations:

Quantum Half Adder

Input			Output		
C	B	A	K	S	A
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	1	0
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	1	0
1	1	1	0	0	1

S = XOR (A, B) (EQ 1)
K = XOR(C, AND (A, B)) (EQ 2)
S is the sum output of the quantum half adder, and K is the carry out xored with the ancillary input C. This can be implemented with one CNOT and one Toffole gates as follows:

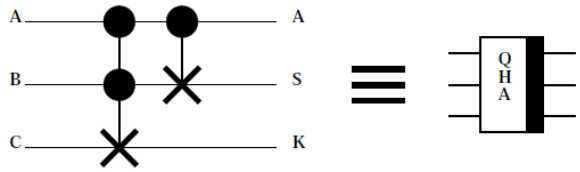


Fig 2:

4. Proposed Design of 4*1 Quantum multiplexer circuit:

Our design as in fig 3 below engages the quantum circuit like quantum half adder and two controlled not gate like C^2NOT and CNOT gate. The half adder [16] output is giving 3 output lines. The output line A is shorted from the input. The 2nd output line, S is given by the expression S = XOR (A, B) and third output line, K is given by the expression K = XOR(C, AND (A, B)).
In a classical multiplexer circuit, depending on the selection line, only one input will move to the output.
Based on this classical logic we have designed the quantum multiplexer circuit.
The first two inputs of every half adder should be treated as the two selection lines and the third line is one out of four inputs of 4 * 1 multiplexer.
When S_{00}=0 and S_{01}=0, ket I_0 will be selected, this input will move to the third output line of the first QHA and finally will move to the final QMUL output.
When S_{10}=0 and S_{11}=1, ket I_1 will be selected, this input will move to the third output line of the second QHA and finally will move to the final QMUL output.
When S_{20}=1 and S_{21}=0, ket I_2 will be selected, this input will move to the third output line of the third QHA and finally will move to the final QMUL output.
When S_{30}=1 and S_{31}=1, ket I_3 will be selected, this input will move to the third output line of the fourth QHA and finally will move to the final QMUL output.

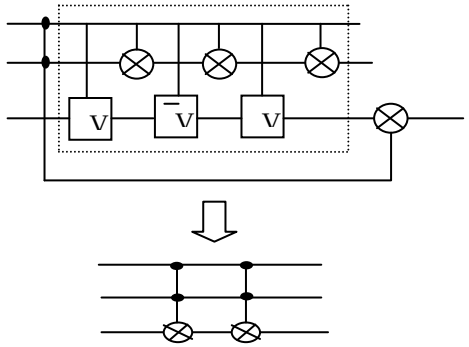
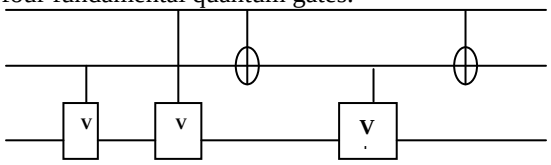


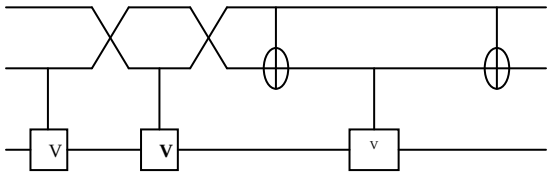
Fig 2: Fourth QHA section and its equivalent circuit

5. QUANTUM CIRCUIT SYNTHESIS

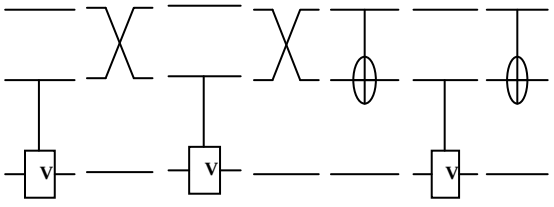
The C^2NOT Gate or Toffole Gate is a well known three input quantum gate circuit .This circuit is made up of three two input fundamental quantum logic Gate, that are CNOT, control square root of NOT Gate and conjugate control square root of NOT Gate. But if there is any overlap in this circuit, then using the SWAP gate we simplify the overlap. So this circuit is made up of four fundamental quantum gates.



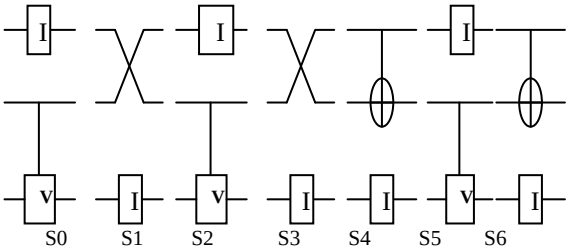
After modification of the previous circuit, we are getting the form below,



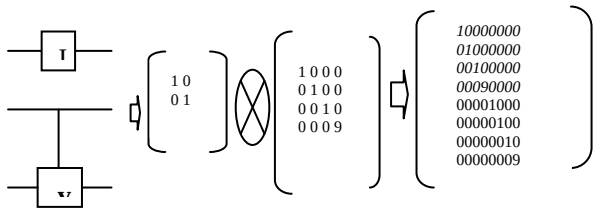
Decomposing the above circuit into seven stages, each stage is made up of a two input fundamental gate.



In the above decomposition, each circuit stage (actually four types stage present) has a fundamental gate that is two inputs but the circuit is of three inputs, then other single input is replaced by using a unitary identity gate.

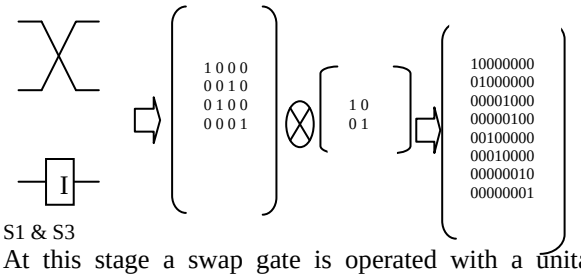


In the above circuit there are four different stages. These four different stages are,



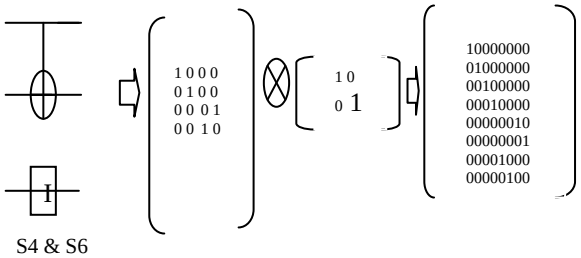
S0 & S2

At this stage a square root of not gate is operated with a unitary identity gate and produces the state matrix.



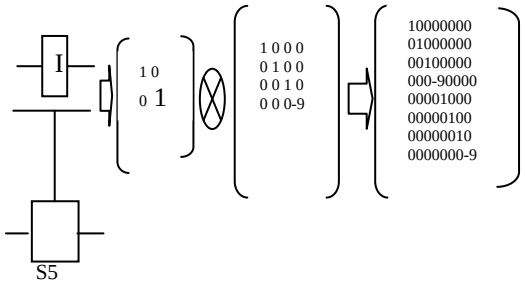
S1 & S3

At this stage a swap gate is operated with a unitary identity gate and produces the state matrix.



S4 & S6

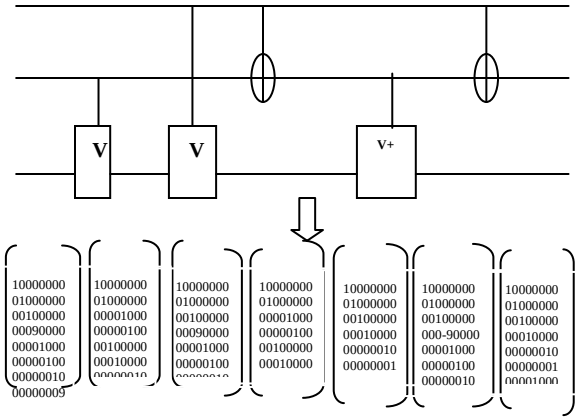
At those particular stages CNOT is operated with unitary identity gate and produces the state matrix.



S5

Here conjugate control square root of not gate is operated with unitary identity gate and produces the state matrix.

In case of square root of not gate and conjugate control square root of not gate, in their gate matrix, we are using a 9 in place of i, which is a complex number. Whenever a multiplication operation between two 9 happens, we will just replace it by -1.

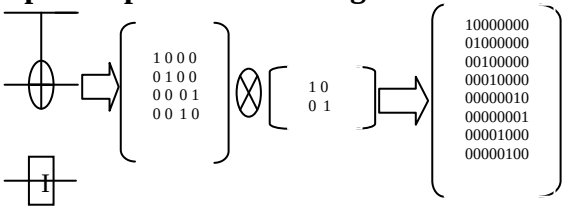


The three input fundamental quantum Gate will be constructed by the two input fundamental quantum Gate. Every two input quantum gate is transformed into three input dummy quantum gate, using kroneckar product[9] with 2*2 unit matrix. This dummy three input gate is called stage. A number of stages are cascading one by one and creating a circuit that performs an operation, that is same operation of a three input quantum fundamental gate. When the stages are in cascade, there becomes a particular sequence, otherwise the required gate matrix are not same.

In three input gate generation, using two input gate, we generate all possible three input dummy gates of two input gates using kroneckar product. Each three input dummy gate assigns an index. Using this index we can access a particular matrix. The number of stage, required to construct a gate, was previously defined.

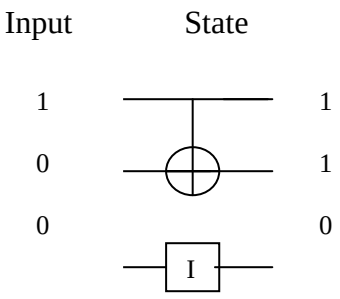
We are denoting those state matrixes by the index 0 to 6. Whenever we are generating a chromosome, we are using these seven numbers in different combinations and multiplying these state matrices according to this combination.

Input/output relation using the state matrix



The figure above shows a state of a quantum circuit. The input/output mapping can be done

using the state matrix. Here a CNOT gate is operated with unitary identity matrix and generate a state matrix.



Here we are getting the output from this particular state input. It can be shown by operating the state matrix with the input.The equivalent decimal value of the input is represented by placing a 1 at that particular postion in the input matrix. The position of 1 in the output matrix below, represents the equivalent decimal value of the output of the state.

$$\begin{pmatrix} 10000000 \\ 01000000 \\ 00100000 \\ 00010000 \\ 00000010 \\ 00000001 \\ 00001000 \\ 00000100 \end{pmatrix} * \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

6. GENETIC ALGORITHM FOR GETTING FITTEST QUANTUM CIRCUIT:

1. Read the pre defined length of chromosome(n)
2. Randomly initialize population
3. Determine fitness of population
4. repeat
   - i. select parents from population
   - ii. Generate crossover point within a range varying with length of chromosome.
   - iii. Perform crossover operation on parents.
   - iv. Generate mutation points within a range, varying with length of chromosome.
   - v. Perform mutation operation
   - vi. New generation created
   - vii. Determine fitness of new generation chromosome.
5. Until best individual is good enough.

7 . DESCRIPTION OF GENETIC ALGORITHM WITH EXAMPLE

7.1 Creating a set of random chromosome like this (taking here seven chromosomes)

chromosome [0]	S2	S3	S5	S6	S1	S0	S4
chromosome [1]	S4	S6	S1	S5	S2	S0	S3
chromosome [2]	S3	S5	S1	S0	S4	S2	S6
chromosome [3]	S6	S0	S3	S5	S1	S2	S4
chromosome [4]	S4	S0	S2	S6	S1	S5	S3
chromosome [5]	S3	S4	S6	S1	S0	S5	S2
chromosome [6]	S2	S3	S4	S5	S0	S1	S6

7.2 Now have to calculate the fitness value of each chromosome.
Here, in our problem we shall calculate the fitness value by the following procedure-

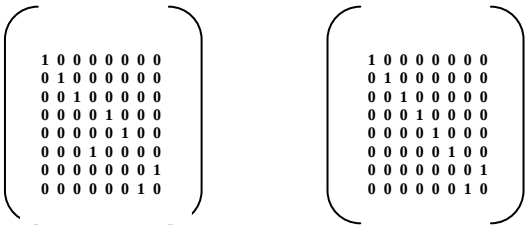
(a) Calculate the product of all the seven gene matrices.
(b) Now the resultant state matrix of chromosome is to be compared element by element with the state matrix of required gate.

(c) Set a variable “count”. Now if any element of the resultant state matrix of a randomly taken chromosome matches with the corresponding element of the state matrix of required gate, then increase the variable count by 1.

Now fitness of a chromosome = count/ number of elements in a state matrix.

Example:

The fitness value of the of the chromosome [0] can be calculated as follows



Resultant Matrix Required Matrix

Fitness value of chromosome [0]= 59/64=. 9218

7.3. Solution

If fitness value of any of the seven chromosomes attains a value which is equal to or greater than a certain value that is set by the user previously, the program ends and we get the required chromosome otherwise we have to proceed to the next stage.

7.4. Crossover and mutation for new generation

Crossover: We have to randomly generate a crossover point for each chromosome, (varying with the length of chromosome) in which the crossover should take place and the value of the crossover points should be less than the number of gene present in that chromosome. Now the seven chromosomes after crossover become,

The randomly generated seven crossover points are,

2, 0, 3, 4, 5, 2, 4

chromosome [0]	S2	S3	S4	S0	S1	S6	S5
chromosome [1]	S4	S3	S0	S2	S5	S1	S6
chromosome [2]	S3	S5	S1	S6	S2	S4	S0
chromosome [3]	S6	S0	S3	S5	S4	S2	S1
chromosome [4]	S4	S0	S2	S6	S1	S3	S5
chromosome [5]	S3	S4	S2	S5	S0	S1	S6
chromosome [6]	S2	S3	S4	S5	S6	S1	S0

Mutation: We have to randomly generate two mutation points for each chromosome (varying with the length of chromosome) in which the mutation should take place and one value of the mutation point should be less than the other point and the second point must be less than the number of gene present in the chromosome. Now the seven chromosomes after mutation become,

The randomly generated seven pair of mutation points are

(3 5) (2 6) (2 4) (3 5) (2 4) (0 5) (1 5)

chromosome [0]	S6	S5	S0	S1	S4	S3	S2
chromosome [1]	S6	S0	S2	S5	S1	S3	S4
chromosome [2]	S2	S4	S0	S1	S6	S5	S3
chromosome [3]	S2	S1	S5	S4	S3	S0	S6
chromosome [4]	S1	S3	S5	S2	S6	S0	S4
chromosome [5]	S1	S6	S0	S5	S2	S4	S3
chromosome [6]	S1	S0	S3	S4	S5	S6	S2

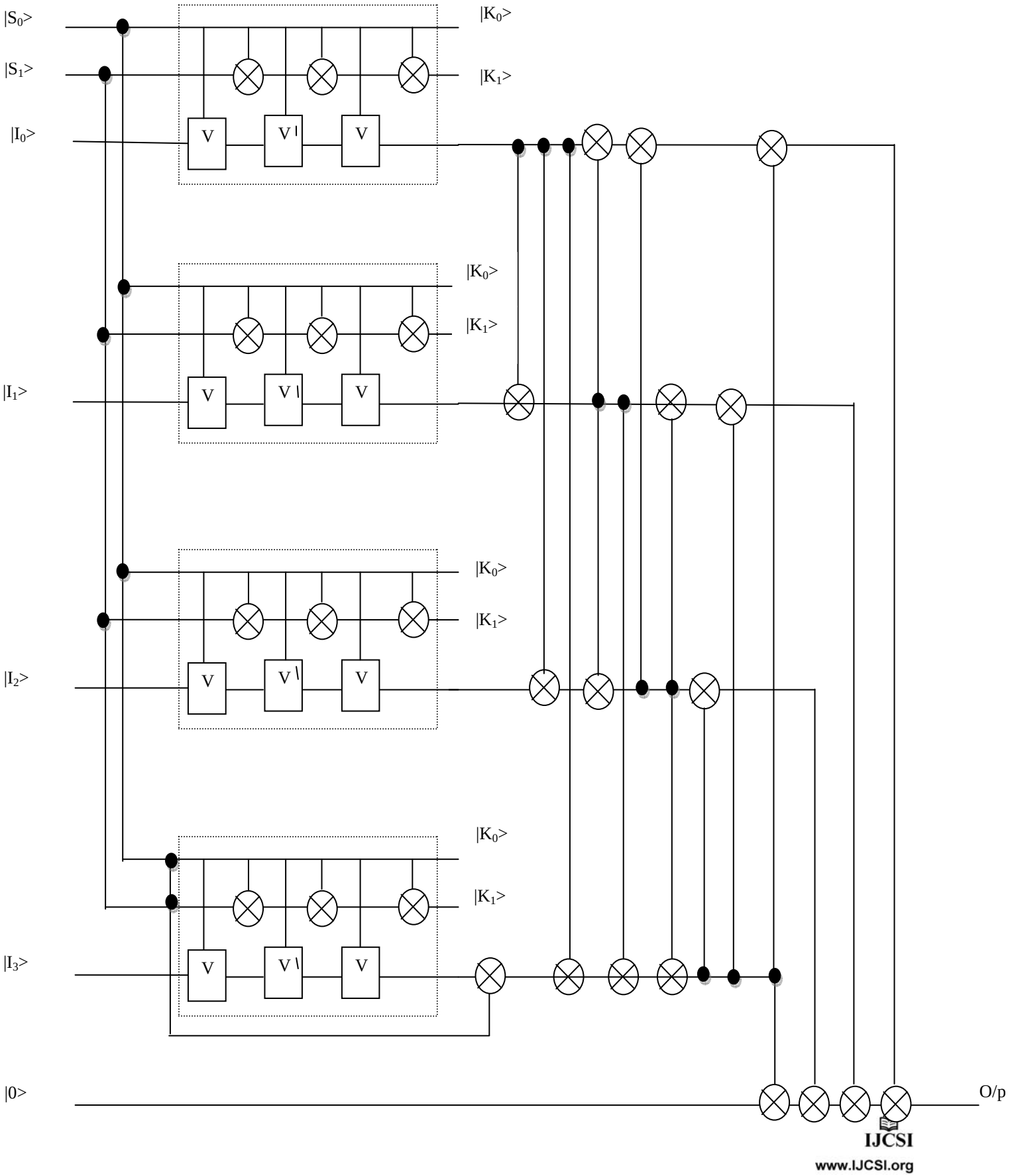
We have to calculate the fitness values of the new generation chromosomes which are generated after crossover and mutation operation. If fitness value of any of the seven chromosomes attains a value, which is equal to or greater than a certain value, that is previously set, then the program ends and we get the required chromosome otherwise we have to proceed again to the next stage.

If no new generation is generated after crossover and mutation, the program terminates.

8. CONCLUSION

Our work designing the model and optimizing a Quantum circuit, which accepts the entry of any size of 3 input quantum circuit. The GA also incorporates the means to process a given quantum circuit. Our future work will incorporate the designing of register and CPU and optimizing different section of those circuit by applying GA.

Fig 3. (Proposed design of Quantum Multiplexer circuit)



9. REFERENCES

[1] P.W. Shor, "Quantum Computing", *Documenta Mathematica - Extra Volume ICM*, Start Page 1 (1998).

[2] H.Buhrman, R. Cleve and A. Wigderson. "Quantum vs. Classical Communication and Computation," *Proceedings of the 30th Annual ACM Symposium on the Theory of Computation*, ACM Press, El Paso, Start page. 63 (1998).

[3] R.P. Feynman, "Quantum Mechanical Computers" , *Foundations of Physics* , Vol.16, Start Page.507 (1986).

[4] H.K.Lo, and S. Popescu and T. Spiller, "Introduction to Quantum Computation and Information", World Scientific, Singapore (1999).

[5] J. Preskil, "Quantum Computing: Pro and Con", Proc. Royal Society, A454, London, Start Page 469 (1998).

[6] M.A.Nielsen, and I.L. Chuang, Quantum Computation and Quantum Information, Cambridge University Press (2002).

[7] D.G.Cory, et al., "Quantum Simulations on a Quantum Computer", Physics Review Letters, start Page 5381 (1999).

[8] Ashok Muthukrishnan, **"**Classical and Quantum Logic Gates: An Introduction to Quantum Computing" Quantum Information Seminar, Friday, Sep. 3, 1999, Rochester Center for Quantum Information (RCQI)

[9] Alexander Graham. "Kronecker Products and Matrix Calculus With Applications". Halsted Press,John Wiley and Sons, NY, 1981.

[10] Genetic Algorithms - Principles and Perspectives : A Guide to GA Theory By: Reeves, Colin R.; University of Birmingham), Jonathan Rowe (School of Computer Science; CoventryUniversity), Colin R. Reeves (School of Math and IS Published By: Springer

[11] Genetic Algorithms in Search, Optimization, andMachine Learning by David E. Goldberg ISBN-13: 9780201157673,Publisher: Addison-Wesley

[12] Practical Genetic Algorithms by Randy L. Haupt, Sue Ellen Haupt, ISBN-13: 9780471455653, Wiley, John & Sons.

[13] L. K. Grover. Quantum mechanics helps with searching for a needle in a haystack. *Phys. Rev. Let.*,79:325, 1997.

[14] P. Shor. Polynomial-time algorithms for prime factorization and discrete logarithm on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[15] C. H. Bennett and G. Brassard. Quantum cryptography: Public-key distribution and coin tossing. In *Proceedings of IEEE International Conference on Computers, Systems, and Signal Processing*, page 175179, Bangalore, India, 1984. IEEE Press.

[16] Phil Gossett, "Quantum Carry-Save Arithmetic" Silicon Graphics, Inc. 2011 N. Shoreline Blvd. Mountain View, CA 94043-1389, August 29, 1998

Mr. Debarka Mukhopadhyay passed graduation in Electronics and Telecommunication Engineering (AMIETE) from IETE(New Delhi), India in 2003 and M.Tech in Computer Science and Engineering from Kalyani Government engineering College, West Bengal, India in 2007.His interest includes Quantum computing, VLSI,Image processing etc. He has published many papers in national and international Journal and Conferences. He is a life member of IETE.

Mr. Amalendu Si passed graduation in Information Technology (B. Tech) BUIE, West Bengal,India in 2005 and M.Tech in Computer Science and Engineering from Kalyani Government engineering College, West Bengal, India in 2007. .His interest includes Quantum computing, High performance computing,soft computing. He has published many papers in national and international Journal and Conferences.