2015 IEEE International Symposium on Nanoelectronic and Information Systems

Synthesis of Vertex Coloring Problem using Grover's Algorithm

Amit Saha, Avik Chongder, Sudhindu Bikash Mandal, Amlan Chakrabarti
A. K. Choudhury School of Information Technology, University of Calcutta, Kolkata - 700 009, India

***Abstract*—Many applications such as Graph coloring, Triangle finding, Boolean satisfiability, Traveling salesman problem can be solved by Grover's search algorithm, which is one of the remarkable quantum computing algorithm. To design a quantum circuit for a given quantum algorithm, which involves Grover's search, we need to define an oracle circuit specific to the given algorithm and the diffusion operator for amplification of the desired quantum state. In this paper, we propose a quantum circuit implementation for the oracle of the vertex coloring problem based on Grover's algorithmic approach. To the best of our knowledge, this is a first of its kind approach in regards to the quantum circuit synthesis of the vertex coloring oracle in binary quantum domain. We have performed the synthesis of the proposed oracle circuit for the six commonly available Physical Machine Description (PMD) gate libraries using the Fault Tolerant Quantum Logic Synthesis (FTQLS) tool. The synthesis results have been presented to understand the cost estimation of the oracle circuit for the various PMDs in terms of number of quantum operations and number of cycles.**

***Index Terms*—Vertex coloring, Grover's Search, PMD**

I. Introduction

Quantum computation is a remarkable technological area based on basic concepts of quantum mechanics. In the last two decades, quantum algorithms have received considerable attention regarding performance speedup in the theoretical computer science community. Quantum algorithms offer numerous algorithmic speedup than their classical counterpart by using some quantum operations like superposition, interference, entanglement. These exclusive properties of quantum system make the quantum algorithms richer than the conventional classical algorithms and make them distinguishable. The quantum analogous of the classical bit is the qubit. A qubit is a quantum system, which can be completely defined by the superposition of two orthonormal basis states $|0\rangle$ and $|1\rangle$ in Hilbert space [1].

Shor's algorithm is first of the kind quantum algorithms for integer factorization capable of breaking RSA cryptosystem in polynomial time. There are a lot of examples of quantum algorithms like Deutsch-Jozsa algorithm, Bernstein-Vazirani algorithm Grover's algorithm etc, which show how a problem can be solved in lesser time compared to a classical computer for the same problem [1] [2].

Quantum computing techniques are commonly applied to solve search problems. One of the well-known search algorithms is Grover's algorithm, which can easily search a marked item in an unsorted database quadratically faster compared to its classical counterpart. The traveling salesman problem, Boolean satisfiability problem, Graph coloring problem can be easily solved using the remarkable properties of Grover's search problem [2].

Vertex coloring problem finds whether in a given graph every two vertices linked by an edge have different colors. It is hard to find appropriate solution for large graphs on a classical computer. Suppose n is the number of nodes, c is the maximum number of colors, then to find exact solution using classical algorithm require $O(2^{n*logc})$ number of steps. Whereas, using Grover's algorithm, finding the exact solution requires $O\sqrt{N}$ number of iterations where N is 2^{n*logc}. Previously in [3] [4], Graph coloring problem using Grover's algorithm has been discussed in the context of ternary quantum system. But, we plan to synthesize the circuit of vertex coloring problem in binary quantum system for the first time by using the decision oracle and the diffusion operator of Grover's algorithm. Oracle block marks the searched state by inverting its amplitude, diffusion operator amplifies the amplitude of the searched state and after $O(\sqrt{N})$ number of iterations the amplitude of the searched state will be maximum, while the amplitude of the other states will be reduced to almost zero. We further simulate our proposed circuit with the help of MATLAB simulator [5], followed by quantum cost analysis of the proposed oracle block using FTQLS [6]. FTQLS cost results estimate the implementation cost of the proposed oracle circuit for the various quantum technologies [9].

The synopsis of Grover's algorithm [2], Functional view of oracle, Synthesis of oracle, Different qubit gates, Diffusion operator and the simulation of our vertex coloring algorithm are given in section II. Finally, cost of our proposed algorithm is estimated in Section III. Concluding remarks appear in Section IV.

II. Synthesis of Vertex Coloring Problem

A. Grover's Algorithm

As previously mentioned Grover's algorithm has two parts, namely oracle and diffusion operator. The oracle depends on the specific instance of the search problem. The diffusion operator block is also known as inversion about the average operator and it amplifies the amplitude of the marked state to increase its measurement probability. The block diagram of a typical Grover's algorithm is shown in Figure 1.

B. Functional View of Oracle for Vertex Coloring Problem

The functional view of vertex coloring problem using Grover's search has been presented in this section. We present an algorithm for vertex coloring, which takes a disjoint graph

978-1-4673-9692-9/15 $31.00 © 2015 IEEE
DOI 10.1109/iNIS.2015.55

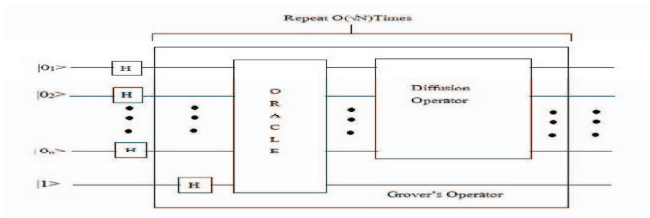

Fig. 1. Generalized Circuit for Grover's algorithm [2]

as an input. This algorithm checks every vertex such that each pair of vertices, which are connected by an edge, are of different colors. Neighbor input variables represent vertices as groups. Coloring of a vertex is illustrated as a binary encoding of the set of qubits corresponding to this node. The colors are differentiated by a unique number. Let $n = \lceil \log N \rceil$ so it takes n bits to represent the vertices of the graph. As we are considering c as the maximum number of colors that we need to color the graph, hence it is same as that of the number of vertices.

Let a, b, $v_c(a)$ and $v_c(b)$ be $4n-$bit string input where a, b are two connected vertices and let $v_c(a)$ and $v_c(b)$ are the color code for vertex a and b respectively. Then the action of the quantum black box U associated with the graph is

$$U|a, b, v_c(a), v_c(b)\rangle = |a, b, v_c(a), v_c(a) \oplus v_c(b)\rangle \qquad (1)$$

where $\oplus$ denotes bitwise addition modulo 2. Assume that we have access to the c unitary transformations one for each of the c possible colors. Clearly, this operation can be performed using the oracle (1). In fact, by checking $v_c(a)$ and $v_c(b)$, it is also straightforward to extend the Hilbert space by a single qubit and perform each of the c transformations

$$V_c|a, b, v_c(a), v_c(b), r\rangle = |a, b, v_c(a), v_c(a) \oplus v_c(b), r \oplus f_c(a)\rangle \qquad (2)$$

where

$$f_c(a) = \begin{cases} 0 & \text{if } v_c(a) = v_c(b); \\ 1 & v_c(a) \neq v_c(b). \end{cases} \qquad (3)$$

by using (1). In summary, we may view the oracle step as a process performing each of the c unitary transformations V_c given by (2), where a, b, $v_c(a)$ and $v_c(b)$ generates a $4n-$bit string. The value of $f_c(a)$ is '1' when two connected vertices have different colors else the value is '0' and r is the single ancilla qubit as output.

C. Synthesis of Oracle for Vertex Coloring Problem

Before obtaining the oracle function we should have the adjacency matrix of the given graph in our hand. With the help of adjacency matrix we can easily understand which two vertices are connected and which aren't. Firstly, we have to make a comparison between two connected vertices by using CNOT gate to check that the two vertices are differently coloured or not. Similarly, we have to compare for all the connected vertices. Then, we have to apply NOT gate on the target qubit line following by a Toffoli gate to an ancilla qubit $|1\rangle$. If the ancilla qubit $|1\rangle$ changes to zero that means any two or more connected vertices are having the same color. The gate level representation of a n-qubit oracle is shown in Figure 6. To synthesize our oracle we require NOT gate, Controlled-NOT gate, Toffoli gate, SWAP gate, Hadamard gate and Multi Control Toffoli gate(MCT). All the mentioned gates [7] are defined in next subsections.

1) NOT Gate:: It is a one qubit basic quantum gate, which is represented by following the matrix:

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

2) Controlled-NOT Gate:: The Controlled-NOT gate has two input bits and two output bits, as illustrated in Figure 2. First bit is the control bit that is unaffected by the action of the Controlled-NOT gate. The second bit is the target bit that is flipped if control bit is set to 1, and otherwise is left alone.

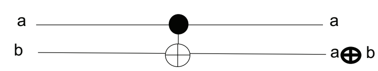

Fig. 2. Symbolic representation of Controlled-NOT gate

3) Toffoli Gate:: The Toffoli gate has three input bits and three output bits, as illustrated in Figure 3. First two bits are the control bits. The third bit is the target bit that is flipped if both the control bits are set to 1, and otherwise is left alone.

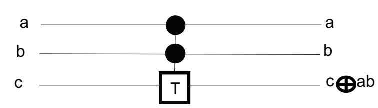

Fig. 3. Symbolic representation of Toffoli gate

4) SWAP Gate:: It is a two qubit gate that interchanges the state of the two qubits.

5) Hadamard Gate:: Hadamard gate is a one qubit basic quantum gate. Figure 4 represents the matrix form of the Hadamard gate.

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix}$$

Fig. 4. Matrix Representation of Hadamard Gate

6) Multi-Controlled Toffoli Gate:: There are n number of inputs and outputs in an n-bit MCT. This MCT gate passes the first $n-1$ inputs, which are referred as control bits to the output unaltered. It inverts the n^{th} input, which is referred as the target bit if the first $n-1$ inputs are all ones. An MCT gate is shown in Figure 5 Black dots $\bullet$ represent the control bits and the target bit is denoted by a $\oplus$.

Next, we present a schematic design of an oracle for the vertex coloring problem. Our designed oracle checks whether a complete graph of three vertices (K3) and four vertices (K4)

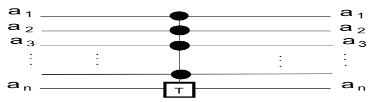

Fig. 5. Multi-Control Toffoli Gate

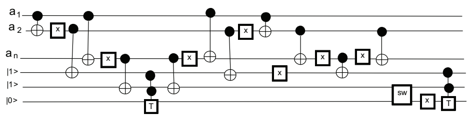

Fig. 6. Gate level representation of n-qubit Oracle

are properly colored or not. Our oracle has to compare every adjacent vertex, which is connected by an edge to complete this task. If every adjacent vertex is having different colors, then only our proposed oracle marks these states. We calculate the gate count for n number of vertices, for this oracle we need $n(n-1)(2*log(n)+1)+3n-5$ number of gates where $n \geq 3$. With the help of this realization we first design generalized oracle for K3 and K4 graphs, which are shown in Figure 7 and Figure 8 respectively. Gate count estimation of generalized oracle is given below :

For n number of vertices the 'First gate pattern' in Figure 7 will be repeated $n-2$ times and the total gate count of the 'First gate pattern' will be $(n{-}2){*}\{(n{+}1){*}(2{*}\lceil\log n\rceil{+}1){+}1\}$ and the total number of gate count of the 'Second gate pattern' in Figure 7 for n number of vertices will be $2{*}(2{*}\lceil\log n\rceil){+}1$.

In Figure 7 and Figure 8 Block 1 containing CNOT gate, NOT gate and Toffoli gate or MCT gate represents a particular gate pattern present in the 'First Gate Pattern' for three vertices and four vertices respectively. Therefore, for n vertices this block contains $2*\lceil\log n\rceil+1$ gates. As a result for n number of vertices the block gets repeated $(n-2)(n+1)$ times. Also the Toffoli gate or the MCT gate present only on the ancilla qubit in the 'First Gate Pattern' gets repeated $n-2$ times for the same n number of vertices. So to conclude the total number of gate count of the 'First Gate Pattern' computes to $(n-2)*\{(n+1)*(2*\lceil\log n\rceil+1)+1\}$.

In Figure 7 and Figure 8 another gate pattern containing CNOT gates and NOT gates is represented in the 'Second Gate Pattern' shown in Block 2. For n number of vertices Block 2 contains $2*\lceil\log n\rceil$ gates. This block gets repeated twice for irrespective number of vertices. A Toffoli gate or a MCT gate is also present in the 'Second Gate Pattern'. Therefore, the total number of gate count of the 'Second Gate Pattern' is $2*(2*\lceil\log n\rceil)+1$.

For n = 3, Number of Swap gates = 1

For n = 4, Number of Swap gates = 2

Similarly for n vertices, Number of Swap gates = $n-2$

For n = 3, Number of NOT gates on ancilla qubit = 2

For n = 4, Number of NOT gates on ancilla qubit = 3

Similarly for n vertices, Number of NOT gates on ancilla qubit = $n-1$

At the last part of the circuit, a single Toffoli or MCT gate is applied on that ancilla qubit, which is irrespective of the number of vertices.

$$GC(GateCount) = \underbrace{(n-2)*\{(n+1)*(2*\lceil\log n\rceil+1)+1\}}_{\text{For first gate pattern}} + \underbrace{2*(2*\lceil\log n\rceil)+1}_{\text{For second gate pattern}} + \underbrace{n-1}_{\text{NOT gate on ancilla qubit}} + \underbrace{n-2}_{\text{Swap gate}} + \underbrace{1}_{\text{Toffoli or MCT gate at the last of the circuit}} \quad (4)$$

After simplification of (4) we have following number of total gate count shown in (5) :

$$GC = n*(n-1)*(2*\lceil\log n\rceil+1)+3*n-5 \quad (5)$$

In case of minimized oracle we have to consider the minimum number of colors, which can be easily obtained by analyzing the adjacency matrix of the provided graph. At least three colors are required for K3 graph that are encoded as $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$ as no color and for K4 graph all the possible $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$ colors are needed. In Figure 9, we have minimized our oracle circuit considering the minimum number of color for K3 graph to properly color the graph by adding six Toffoli gates and one MCT at the start of the oracle to bypass the color code $|11\rangle$ as it is an invalid color. This minimization step is not required for K4 graph as the number of color is same as the possible number of color.

We also have to decompose all the MCT into 2-qubit and 1-qubit gates because as we map our oracle circuit in different PMDs it gives least cost circuit if the circuit is having upto 2-qubit gates. To make our circuit PMD specific we decompose all the MCT. This minimization technique depends on the graph, which has been provided. So, the gate count of the minimized oracle varies from one graph to another. In Figure 10, we decompose all the MCT into 2-qubit and 1-qubit gates to minimize our oracle for complete three graph and similarly described in Figure 11 for complete four graph.

D. Diffusion Operator

This operator inverts the amplitude of the input states about their mean value of amplitude. The gate level representation for six qubits diffusion operator is shown in Figure 12, where we use NOT gate, Hadamard gate and 5 control Toffoli gate (MCT). NOT gate, Hadamard gate and Multi controlled Toffoli gate are already discussed in previous section.

E. Simulation of Vertex Coloring Algorithm

We have taken K3 and K4 graph as example to describe our algorithm. We use two qubits to represent the color of a vertex. So the available combinations for color are $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$. We require only three combinations among these four for K3 graph as we take $|00\rangle$, $|01\rangle$, $|10\rangle$ as our three color and $|11\rangle$ as an invalid color and we take all possible colors for K4 graph. So the states of the system for K3 are $|000000\rangle$, $|000001\rangle$ ..... $|111111\rangle$ and for K4 are $|00000000\rangle$, $|00000001\rangle$

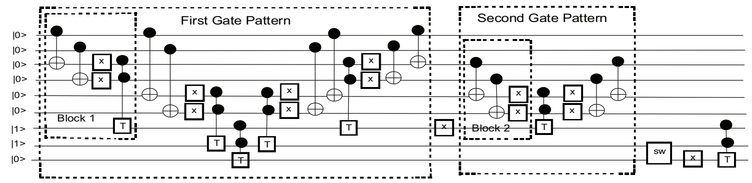

Fig. 7. Gate level representation of 6-qubit generalized oracle

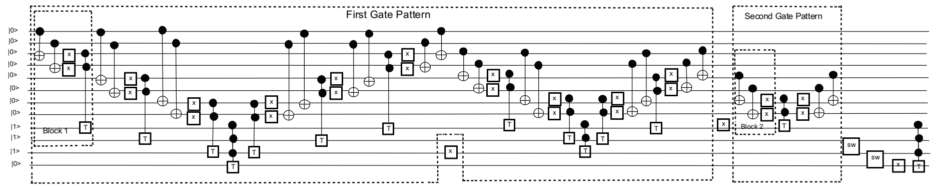

Fig. 8. Gate level representation of 8-qubit generalized oracle

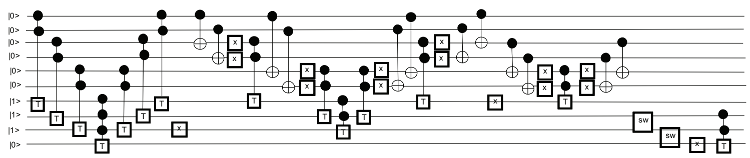

Fig. 9. Gate level representation of 6-qubit oracle considering minimum color

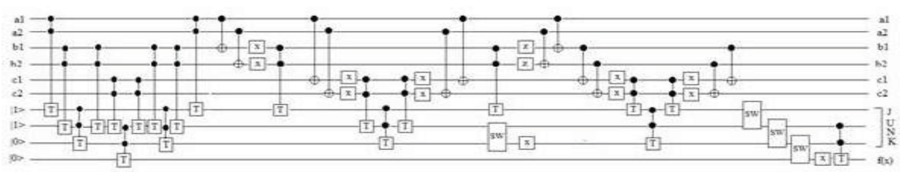

Fig. 10. Gate level representation of 6-qubit oracle

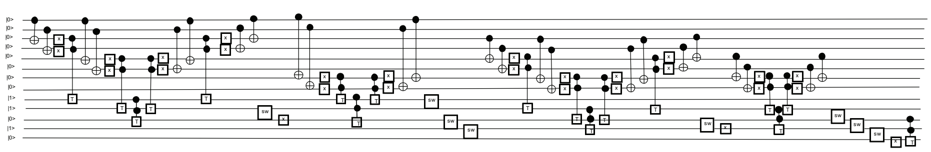

Fig. 11. Gate level representation of 8-qubit oracle

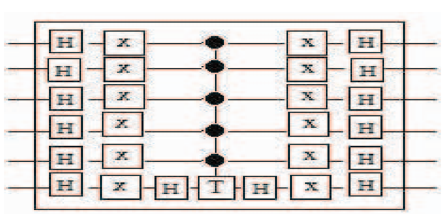

Fig. 12. Six qubit Diffusion Operator

..... $|11111111\rangle$. We have to identify those states that satisfy the constraints of the vertex coloring problem and also do not contain any invalid color combination for any vertex. So our database contains 2^6 elements for K3 and 2^8 for K4. The gate level representation of Grover's circuit is shown in Figure 13 for K3 and in Figure 14 for K4.

The description of simulation steps of our algorithm are given below :

Step 1: At first we have to initialize $n-$qubits with $|0\rangle$ as we initialized six qubits and eight qubits with $|0\rangle$ in Figure 13 and Figure 14 respectively.

Step 2: Then we have to apply Hadamard gate on $n-$ qubits as we apply the $H^{\otimes 6}$ and $H^{\otimes 8}$ gate to create all possible 2^6 (64) states ($|000000\rangle$, $|000001\rangle$ ..... $|111111\rangle$) and 2^8 (256) states ($|00000000\rangle$, $|00000001\rangle$ ..... $|11111111\rangle$) respectively.

Step 3: The oracle compares the color of every adjacent vertex and inverts the amplitude for true elements like six elements for K3 graph which are $|000110\rangle$, $|001001\rangle$, $|010010\rangle$,

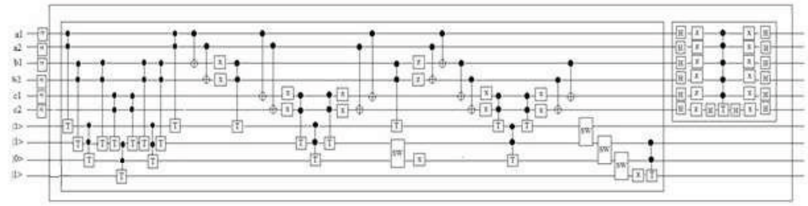

Fig. 13. Gate level representation of Grover's circuit for Vertex Coloring of K3 graph

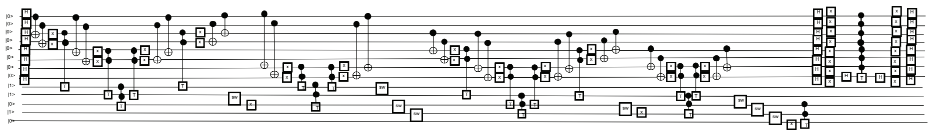

Fig. 14. Gate level representation of Grover's circuit for Vertex Coloring of K4 graph

$|011000\rangle$, $|100001\rangle$, $|100100\rangle$ and similarly 24 elements for K4 graph respectively.

Step 4: The output of the oracle is acted upon by the diffusion operator which amplifies the amplitude of the marked states of step 3.

Step 5: Step 4 is repeated for $\pi/4\sqrt{64/6}$ times for K3 graph and similarly K4 graph it requires $\pi/4\sqrt{256/6}$ (For multiple solution of Grover's operator having N = number of elements in the database and M = number of marked states we need $\pi/4\sqrt{N/M}$ number of iterations [8]).

The output of Step 2 is shown Figure 15 and Figure 16. The output of Step 3 is shown in Figure 17 and Figure 18. The final output, i.e. the output of step 5 is shown in Figure 19 and Figure 20. We can verify the location of the searched states by subtracting 1 from the index value of figure and then converting it to its equivalent binary value. For example, if we take index value 10 from figure and after subtracting 1 from it, converting it to its equivalent binary value we get the string $|001001\rangle$, which is one of the searched state.

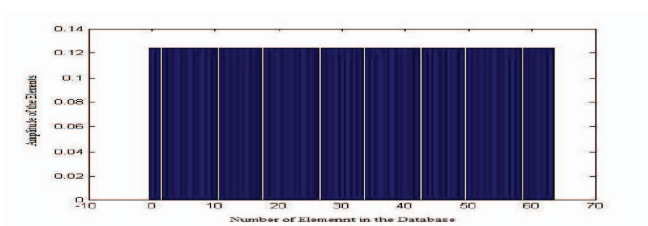

Fig. 15. Simulation output of $H^{\otimes 6}$

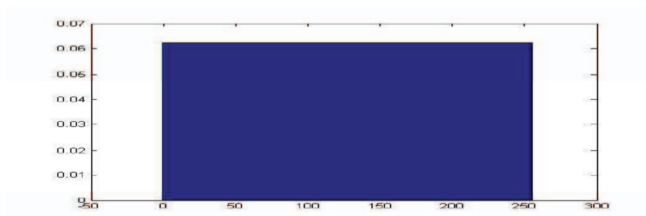

Fig. 16. Simulation output of $H^{\otimes 8}$

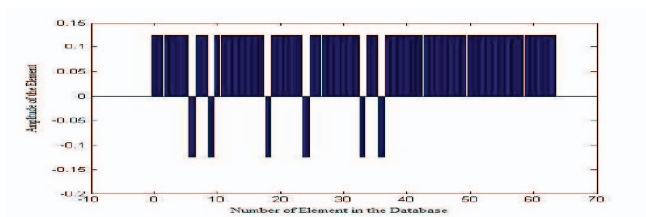

Fig. 17. Simulation output of oracle

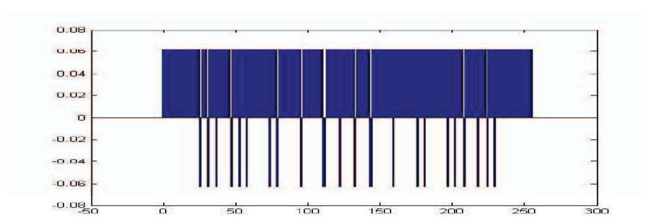

Fig. 18. Simulation output of oracle of four vertex

III. Experimental Results

In order to estimate the implementation cost of the synthesized quantum circuit we have to map the quantum gate operations to the primitive quantum operations supported by the PMDs [9]. The PMD specific cost estimation of the quantum circuits can lead to the understanding of favoring some PMDs compared to the others in terms of optimized quantum implementation. We have used the FTQLS synthesis tool to evaluate the cost of our algorithm for vertex coloring problem for six different PMDs [9] which are Quantum dots(QD), Ion trap(IT), Superconducting(SC), Neutral atom(NA), Nonlinear photonics(NP), Linear photonics(LP). FTQLS gives an optimized and fault tolerant description of the quantum circuit in terms of circuit netlist involving primitive PMD specific

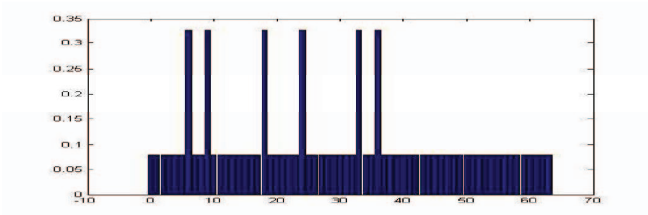

Fig. 19. Output after three iterations of three vertex

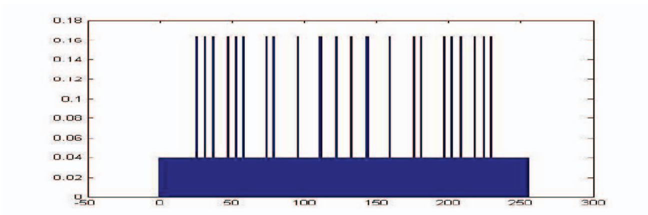

Fig. 20. Output after three iterations

quantum operations. FTQLS estimates the quantum cost of the quantum circuit in terms of (i) operational cost and (ii) cycle cost. The first column in each major column indicates original cost (orig.), the second column the optimized cost (opt.) generated by FTQLS, and the third column describes the percentage reduction in cost values (red.).

A. Results of six qubit oracle

The quantum cost in terms of #ops and #cycles of completely minimized K3 graph for all PMDs are shown in Table 1. #ops is least for LP and the number of #ops is 295. #cycles is least for NA and the number of #cycles is 179.

TABLE I
COST ANALYSIS FOR FAULT TOLERANCE OF SIX QUBIT ORACLE

	#ops			#cycles		
PMD	orig	opt	red(%)	orig	opt	red(%)
QD	1347	440	67.3	1978	1141	42.3
IT	1098	359	67.3	431	191	55.7
SC	1342	475	64.6	3021	2322	23.1
NA	1243	379	69.5	350	**179**	48.9
NP	464	324	30.2	1312	1248	4.8
LP	472	**295**	37.5	1127	1051	6.7

B. Results of eight qubit oracle

We now illustrate the quantum cost in terms of #ops and #cycles of completely minimized K4 graph for all the PMDs are shown in Table 2. For our algorithm, LP gives the least number of #ops which is 286 and NA gives the least number of #cycles and the number of #cycles is 160.

Finally, we compare the quantum cost in terms of #ops of six-qubit oracle and eight-qubit oracle for the different PMDs which is shown in Figure 21.

TABLE II
COST ANALYSIS FOR FAULT TOLERANCE SYNTHESIS OF EIGHT QUBIT ORACLE

	#ops			#cycles		
PMD	orig	opt	red(%)	orig	opt	red(%)
QD	1375	471	65.7	1694	993	41.38
IT	1078	380	64.74	378	166	56.08
SC	1378	490	64.4	2560	1998	21.95
NA	1206	386	68	306	**160**	47.71
NP	436	319	26.8	1178	1123	4.66
LP	442	**286**	35.3	994	942	5.23

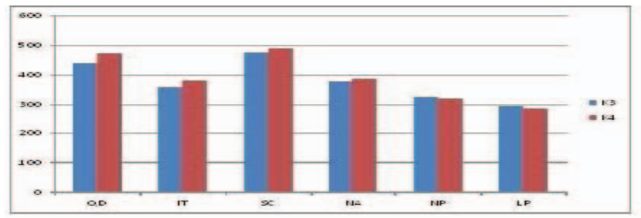

Fig. 21. Cost comparison of six qubit oracle and eight qubit oracle

IV. CONCLUSION

In our paper, we proposed quantum circuit for the vertex coloring algorithm of a graph as an example application of Grover's algorithm. We have designed a gate level synthesis of the vertex coloring oracle and also verified the circuit instances for various qubit sizes through simulation. PMD specific synthesis of these oracle circuits have been performed using the FTQLS tool for evaluating their quantum cost. We have compared the quantum cost in terms of #quantum operations and #cycles for six and eight qubit vertex coloring oracle circuits for the various PMDs. We found that LP gives the least cost in terms of #quantum operations and #cycles is least for NA.

ACKNOWLEDGMENT

This work has been supported by the grant from TEQIP phase 2 of University of Calcutta providing fund for this research.

REFERENCES

[1] M. A. Nielsen and I. L. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, 2002.
[2] Lov. K. Grover, A fast quantum mechanical algorithm for database search.
[3] Y. Wang, M. Perkowski, Improved Complexity of Quantum Oracles for Ternary Grover Algorithm for Graph Coloring, IEEE International Symposium on Multiple- Valued Logic,2011.
[4] S. B. Mandal, A. Chakrabarti, S. Sur-Kolay, Synthesis of Ternary Grovers Algorithm, IEEE 44th International Symposium on Multiple-Valued Logic (ISMVL), 2014.
[5] MATLAB, R2011a, The MathWorks, Natick, MA, 2011.
[6] Chia-Chun Lin, A. Chakrabarti, and N. K. Jha, FTQLS: Fault-Tolerant Quantum Logic Synthesis, IEEE Transactions on very large scale integration (VLSI) systems.
[7] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, Elementary gates for quantum computation, Phys.Rev.A, vol. 52, no. 5, pp. 34573467,Nov. 19.
[8] Michel Boyer, Gilles Brassard, Peter Hyer, Alain Tapp, Tight bounds on quantum searching , arXiv:quant-ph/9605034, 1996.
[9] C. C. Lin, A. Chakrabarti, and N. K. Jha, Optimized quantum gate library for various physical machine descriptions, IEEE Trans. Very Large Scale Integer. (VLSI) Syst., no. 99, Jan. 2013.