APPLICATION OF QUANTUM MACHINE LEARNING TO VLSI PHYSICAL DESIGN

A Project Report

Submitted by:

KOPPARTHIJWALAPATHI NARENDRA REDDY (2114119)
TUMATI PAVAN KUMAR (2114101)
GUDIVADA VAISHNAVI (2114109)

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHONOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

at

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR,
SILCHAR, ASSAM (INDIA)-788010

DECEMBER 2024

DECLARATION

We hereby declare that the project entitled "**Application of Quantum Machine Learning to VLSI Physical design"** submitted for the B. Tech. (ECE) degree is my original work and the project has not formed the basis for the award of any other degree, diploma, fellowship or any other similar titles.

Place:

Date:

……

(Signature)

K.J.Narendra Reddy(2114119)

……

(Signature)

T.Pavan Kumar (2114101)

……

(Signature)

G.Vaishnavi (2114109)

CERTIFICATE

This is to certify that the project titled **"APPLICATION OF QUANTUM MACHINE LEARNING TO VLSI PHYSICAL DESIGN"** is the bona fide work carried out by us, students of B Tech (ECE) of National Institute of Technology Silchar (An Institute of National Importance under MHRD, Govt. of India), Silchar, Assam, India during the academic year 2024-25, in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology (Electronics and communication Engineering) and that the project has not formed the basis for the award previously of any other degree, diploma, fellowship or any other similar title.

Name and Signature of the Supervisor

Place:

Date:

Name and Signature of the Co-Supervisor

Place:

Date:

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude and special thanks to **Dr. Kavicharan Mummaneni, Assistant Professor** in the Electronics and Communication Engineering Department at NIT Silchar, for his invaluable mentorship during our project. His guidance, encouragement, and support have been instrumental in our success.

Throughout the duration of our project, he provided us with his expert advice and direction that greatly influenced our research. His insightful suggestions and constructive feedback helped shape the trajectory of our work, ensuring its quality and depth. We are truly indebted to his for imparting her wisdom and knowledge, which have been pivotal in our intellectual growth. His guidance and co-operation are sincerely acknowledged.

KOPPARTHIJWALAPATHI NARENDRA REDDY(2114119)

TUMATI PAVAN KUMAR(2114101)

GUDIVADA VAISHNAVI(2114109)

ABSTRACT

Recent advances in quantum computing have allowed the field to tackle complex problems with large solution spaces, even with current noisy and limited hardware. Because of these limitations, hybrid quantum-classical methods have become essential. In these methods, a quantum computer generates data that is then used by classical algorithms to adjust control parameters, creating a continuous loop of optimization. One potential area of application is in VLSI (Very Large-Scale Integration) placement problems, which involve arranging components on a chip to minimize the length of connecting wires. Traditionally, algorithms like Kernighan-Lin (KL) have been used to find solutions, but they have limitations. In this context, the Variational Quantum Eigen solver (VQE), a hybrid quantum classical algorithm, is applied to create a recursive Balanced Min-Cut (BMC) approach. This algorithm aims to divide a circuit into smaller sections in an optimized manner. The use of quantum machine learning techniques in this process can reduce error rates and help the algorithm converge to the best solution more quickly. This is crucial because reducing errors and speeding up convergence are significant challenges in quantum computing, especially when dealing with complex problems like VLSI placement.

LIST OF FIGURES

Figure 1: Logic toy circuit consisting of 8 gates 12

Figure 2: Visualization of iteration over the edges of the graph 14

Figure 3: Example of Netlist Partitioning 15

Figure 4: Global placement 16

Figure 5: Toy circuit modeled into a graph 18

Figure 6: Initial circuit partition using Kernighan-Lin 18

Figure 7: Initial partition of circuit using VQE to two subgraphs of 4 gates each 21

Table of Contents

Title Page 1

Declaration of the Student 2

Certificate 3

Acknowledgement 4

Abstract 5

List of Figures 6

1.INTRODUCTION 8

1.1 Problem Definition 8

1.2 Project Overview 8

1.3 Software Specifications 9

2.LITERATURE SURVEY 9

3.SYSTEM ANALYSIS &IMPLEMENTATION STEPS 9

3.1 Requirement Specification 10

3.2 Implementation Steps 11

4.RESULTS/OUTPUTS 16

5.CONCLUSION 21

6.REFERENCES 21

1. INTRODUCTION

1.1 Problem Definition

The goal of this project is to leverage quantum machine learning, specifically the Variational Quantum Eigensolver (VQE) and the Quantum Approximate Optimization Algorithm (QAOA), to address optimization problems in EDA, such as the Max-Cut and Max-SAT problems. These problems are critical in optimizing electronic circuit designs and can be modeled as combinatorial optimization tasks.

This research also aims to integrate machine learning techniques with quantum algorithms to refine their performance despite the limitations of current quantum hardware. By combining quantum computing's ability to handle complex calculations with classical machine learning's optimization power, the project seeks to demonstrate practical applications of quantum machine learning for circuit design in the NISQ era.

1.2 Project Overview

The project focuses on applying quantum machine learning techniques to solve EDA-related optimization problems using hybrid quantum-classical models. The Variational Quantum Eigen solver (VQE) and Quantum Approximate Optimization Algorithm (QAOA) are central to the study.

- **Key Objectives**:
    1. Model EDA problems like Max-Cut and Max-SAT as combinatorial optimization tasks suitable for quantum algorithms.
    2. Use hybrid computation models where quantum devices generate data that classical processors refine using machine learning techniques.
    3. Address the challenges of noise and errors in NISQ devices by optimizing algorithmic parameters with classical machine learning tools, such as meta-learning through recurrent neural networks and gradient-free optimizers like COBYLA.

The project will explore the practical applicability of VQE and QAOA to solve graph-based optimization problems. Special emphasis will be placed on recursive graph partitioning for tasks like balanced Min-Cut, which is essential in electronic circuit placement. The results will be benchmarked against classical algorithms such as Kernighan-Lin to assess performance improvements.

1.3 Software Specification

The implementation of this project requires a combination of quantum and classical software tools to execute hybrid quantum-classical algorithms effectively.

Primary Software Tools:

1. **Qiskit**:

- An open-source quantum computing library developed by IBM.
    - Used for designing and simulating quantum circuits, implementing VQE, and running QAOA.
    - Provides backends like the state vector simulator for efficient testing.

2. **Python**:
    - The primary programming language for developing and integrating the hybrid quantum-classical model.
    - Includes libraries such as NumPy and SciPy for classical computations.
3. **Classical Optimization Libraries**:
    - **SciPy**: Includes optimization algorithms such as COBYLA for parameter tuning in VQE and QAOA.

2. LITERATURE SURVEY

[1], QCOpt is a novel open-source framework that optimizes quantum circuit decompositions by transforming arbitrary unitary gates into hardware-native sequences, offering optimality guarantees. It achieves up to 57% reduction in gate count for small circuits (≤4 qubits) with fast runtimes on commodity hardware. QCOpt supports various quantum hardware platforms, including IBM, Rigetti, and Google, facilitating improved circuit design and exploration for quantum processors.

[2], This paper introduces a quantum algorithm for combinatorial optimization that produces approximate solutions, with the approximation quality improving as a parameter p increases. The quantum circuit has a locality matching the objective function, with depth growing linearly in p and the number of constraints. Applied to Max-Cut on regular graphs, for p=1, it consistently finds cuts at least 0.6924 times the optimal cut on 3-regular graphs.

[3], This paper proposes using Conditional Value-at-Risk (CVaR) as an aggregation function in hybrid quantum/classical variational algorithms for combinatorial optimization. Unlike traditional methods that minimize the expectation of the Hamiltonian, CVaR leads to faster convergence and better solutions, as shown in both classical simulations and quantum hardware experiments. Analytical results are provided to explain the improved performance compared to other variational algorithms.

[4], This collects and extends classic results relating partitioning problems to Ising spin glasses, well as work describing exact covering algorithms and satisfiability. In each case, the state space isat most polynomial in the size of the problem, as is the number of terms in the Hamiltonian. This work may be useful in designing adiabatic quantum optimization algorithms.

[5], This book "VLSI Physical Design: From Graph Partitioning to Timing Closure "introduces and compares algorithms that are used during the physical design phase of integrated-circuit design, wherein a geometric chip layout is produced starting from an abstract circuit design. The emphasis is on essential and fundamental techniques, ranging from hypergraph partitioning and circuit placement to timing closure.

[6], Recent advances in noisy, near-term quantum devices have expanded the application of quantum computing to problems with large solution spaces, but limitations in scale and noise necessitate hybrid quantum-classical approaches. In VLSI placement problems, traditional heuristics like Kernighan-Lin (KL) are combined with the Variational Quantum Eigensolver (VQE) to implement a recursive Balanced Min-Cut (BMC) algorithm. Quantum machine learning techniques are proposed to reduce error rates and accelerate convergence to optimal solutions**.**

3. SYSTEM ANALYSIS & IMPLEMENTATION

3.1 Requirement Specifications

Quantum Algorithm Implementation: Implement Variational Quantum Eigensolver (VQE) with RY ansatz and linear entanglement to solve balanced min-cut problems.

Classical Optimization Integration: Integrate a classical optimizer like COBYLA to optimize parameters for the trial quantum state.

Graph Partitioning: Enable recursive graph splitting for combinatorial optimization problems, such as balanced min-cut.

Data Processing: Compute and aggregate eigenvalues using methods like Conditional Value-at-Risk (CVaR) for improved optimization performance.

Simulation Backend: Utilize Qiskit's statevector simulator to execute and validate quantum circuits.

Scalability: Ensure the system can handle both small toy circuits and larger benchmark circuits (e.g., MCNC benchmarks).
Accuracy: Maintain a high degree of accuracy in graph partitioning by reducing local minima issues.
Performance: Optimize circuit depth (e.g., depth of 3 for VQE) to ensure low computational overhead.
Compatibility: The system should work seamlessly on NISQ devices and classical hardware with Qiskit

support.
Reliability: Ensure robustness to noisy outputs from quantum hardware by using hybrid quantum- classical approaches.
Quantum Hardware: NISQ-compatible quantum processors or state vector simulators for testing.
Classical Hardware: Multi-core CPU and high-performance GPU for simulations and classical optimization tasks.
Software:
Qiskit: For quantum circuit design, simulation, and optimization.
Python: Programming language for implementing quantum and classical algorithms.
Classical Optimizers: COBYLA or any gradient-free optimizer.

3.2 Implementation steps

The implementation of the quantum placement algorithm using the Variational Quantum Eigen solver (VQE) is divided into several well-defined steps. Each step is crucial for successfully performing the placement and ensuring the quantum algorithm's performance is comparable to or better than classical approaches.

Step 1: Setup of Qiskit Environment

Qiskit, an open-source quantum computing library developed by IBM, is used for implementing and simulating the quantum algorithms. The setup includes:

- Installing the required packages and dependencies.
- Selecting the **state vector simulator** as the quantum backend, which provides noiseless simulations for testing the algorithm in ideal conditions.

Step 2: Representation of Circuits

Two circuits are chosen for testing the placement algorithm:

1. **Toy Circuit**:
    - Comprises eight gates modeled as an **unweighted graph**.
    - This simpler problem helps validate the approach before scaling to complex circuits.

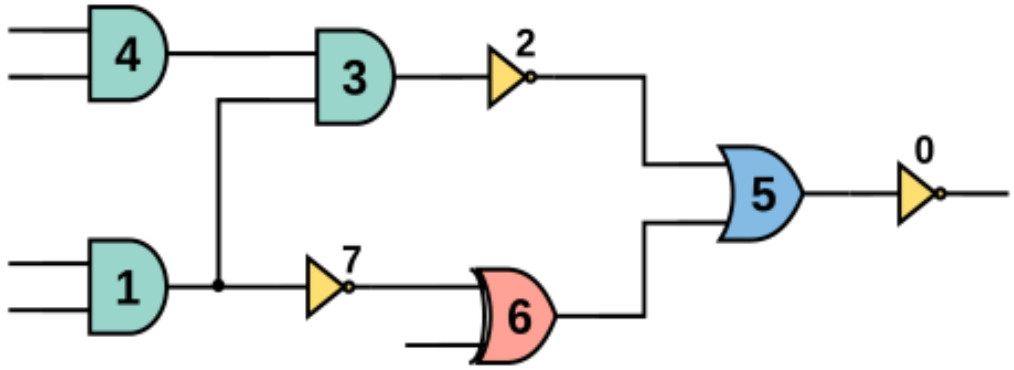

Figure 1: Logic toy circuit consisting of 8 gates [5]

2. **MCNC Benchmark Circuit ("Apte")**:
    - Derived from the Microelectronics Center of North Carolina benchmark suite.
    - Comprises nine modules, modeled as a **dense, weighted graph**.
    - The odd number of modules requires splitting into subgraphs of unequal sizes (five vertices in one and four in the other).

The circuits are mapped to graphs where vertices represent gates or modules, and edges represent interconnections between them.

Step 3: Casting Placement as a Combinatorial Optimization Problem

The **Balanced Min-Cut** placement problem is formulated as a combinatorial optimization problem:

- **Objective**: Minimize the number of connections (cut nets) between partitions.
- The problem is encoded as a Hamiltonian, representing the cost of a given partition.

$$H_A = \sum_{\langle ij \rangle} \frac{1}{2}(1 - z_i z_j)$$

$$H_B = (\sum_i z_i)^2$$

$$H = H_A + H_B$$

- The expectation value of the Hamiltonian serves as the objective function to be minimized.

$$\min_{\theta} \mathbb{E}(H(\vec{\theta}))$$

$$z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Step 4: Ansatz Design

The ansatz is a parameterized quantum circuit used to explore the solution space. For this implementation:

- **RY Ansatz**:
  - Chosen for its simplicity and low computational overhead.
  - Each qubit has one rotation-Y (RY) gate per layer.
  - Entanglement is introduced using controlled-NOT (CNOT) gates in a linear entanglement scheme.
  - The circuit depth is set to three, balancing complexity and performance.

$$RY(\theta_i) = \begin{bmatrix} \cos(\frac{\theta_i}{2}) & -i\sin(\frac{\theta_i}{2}) \\ -i\sin(\frac{\theta_i}{2}) & \cos(\frac{\theta_i}{2}) \end{bmatrix}$$

Step 5: Classical Optimization

- **Constrained Optimization By Linear Approximation (COBYLA)** is used as the classical optimizer.
  - It is a gradient-free optimization technique commonly employed in VQE implementations.
  - COBYLA iteratively adjusts the ansatz parameters to minimize the expectation value of the Hamiltonian.
- For the **toy circuit**, the expectation value suffices as the objective function.
- For the **MCNC circuit**, the **Conditional Value-at-Risk (CVaR)** is used to handle optimization challenges.

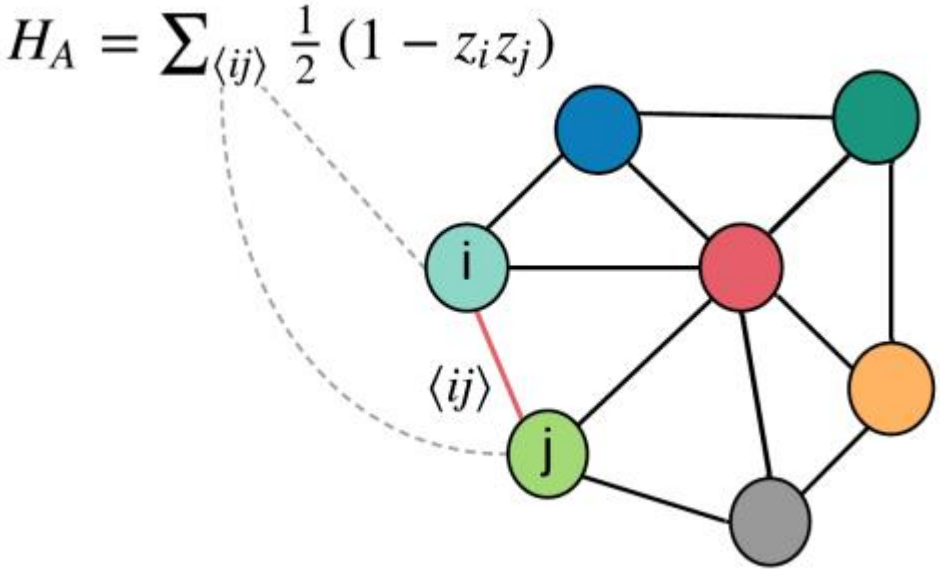

Figure 2: Visualization of iteration over the edges of the graph [2]

- CVaR focuses on low-energy states in the probability distribution, improving convergence to better solutions.

$$\min_{\theta} CVaR(H(\theta))$$

$$CVaR(H(\vec{\theta})) = \mathbb{E}[H(\vec{\theta})|H(\vec{\theta}) \leq F_X^{-1}(\alpha)].$$

Step 6: Recursive Graph Partitioning

The placement algorithm operates recursively:

1. The initial graph is split into two subgraphs using VQE.

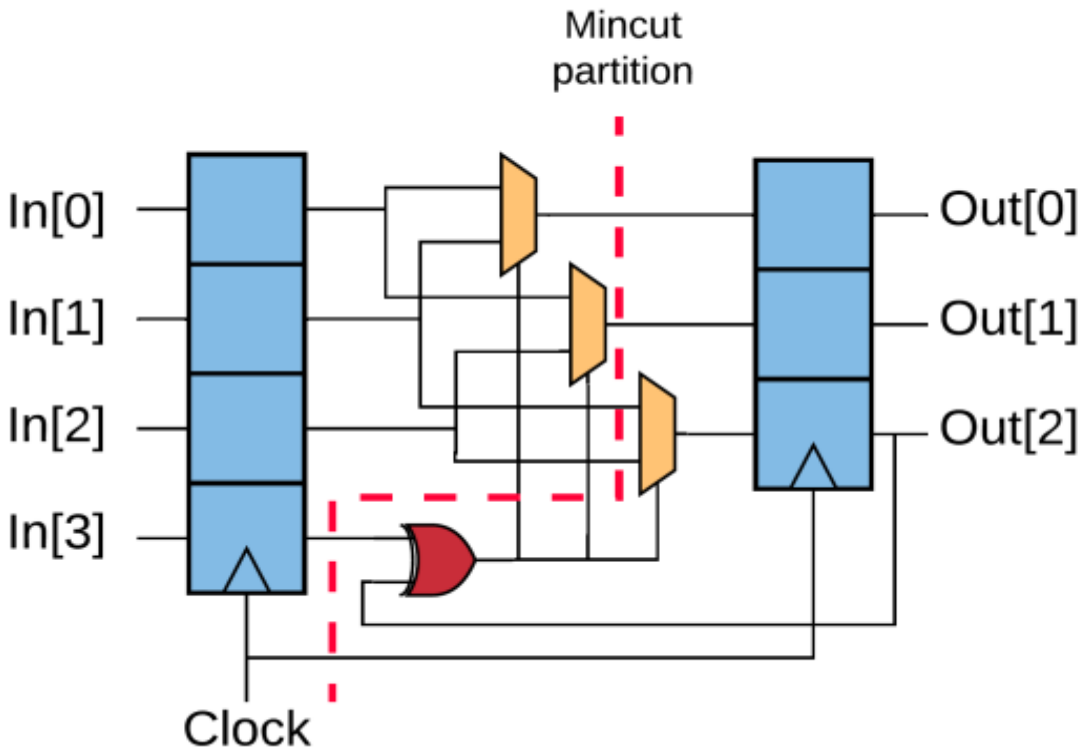

Figure 3: Example of Netlist Partitioning [4]

2. Each resulting subgraph is further partitioned until only one vertex remains in each subgraph.
3. At every step:
    - The quantum circuit is executed.
    - A probability distribution over possible solutions is obtained.
    - The solution with the highest probability is chosen.

This recursive approach ensures a step-by-step reduction in the complexity of the placement problem.

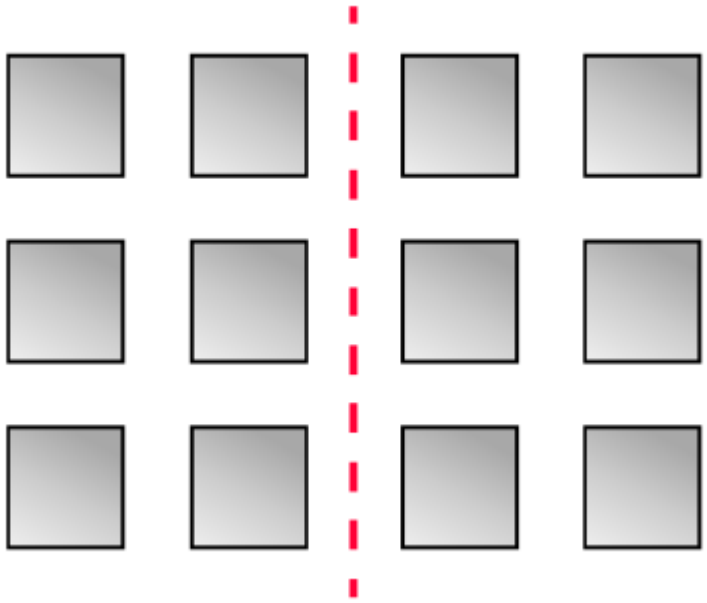

Figure 4: Global placement [4]

Step 7: Handling Optimization Challenges

- Dense and weighted graphs, such as the MCNC benchmark, pose challenges due to the presence of local minima in the energy landscape.
- A pseudo-probability distribution is constructed using the eigenvalues of the quantum circuit output, sorted in ascending order. This simplifies the use of CVaR without reconstructing the true probability distribution.

4.RESULTS /OUTPUTS

Here the implementation of both the Kernighan-Lin (KL) and the Quantum machine learning are given below:

- In the first code the Kernighan-Lin (KL) algorithm is used first we are given the gates of the circuit and specified the connections between each node.
- Then we converted this as a graph by using the ising model.
- Then we partitioned this as two sub graphs by using the Kernighan-Lin (KL) algorithm.
- Next we plotted the result as shown in figure 6.
- In second one VQE is used to get the initial partition of the circuit.
- The process is same as the above one but the VQE is used instead of Kernighan-Lin (KL) algorithm.
- From the figure 6 and figure 7 comparing both the initial cut that we can say that the VQE has the minimum cut than the Kernighan-Lin (KL).

- Here in VQE it requires only 2 cuts through the edges whereas in Kernighan-Lin (KL) requires 3 cuts so from this we can say that VQE is can give the best solution for this vlsi placement problems.

Code for Kernighan-Lin (KL) algorithm:

```python
import networkx as nx
import matplotlib.pyplot as plt

dependency_graph = nx.DiGraph()

nodes = ["AND Gate 4", "AND Gate 1", "AND Gate 3", "NOT Gate 0", "NOT Gate 2", "NOT G
dependency_graph.add_nodes_from(nodes)

dependency_graph.add_edges_from([
    ("AND Gate 4", "NOT Gate 2"),      # AND Gate 4 output goes to NOT Gate 2
    ("AND Gate 1", "AND Gate 3"),      # AND Gate 1 output goes to AND Gate 3
    ("AND Gate 3", "NOT Gate 2"),      # AND Gate 3 output goes to NOT Gate 2
    ("NOT Gate 2", "OR Gate 5"),       # NOT Gate 2 output goes to OR Gate 5
    ("AND Gate 1", "OR Gate 6"),      # AND Gate 1 output goes to NOT Gate 7
    ("NOT Gate 7", "AND Gate 1"),       # NOT Gate 7 output goes to OR Gate 6
    ("OR Gate 6", "OR Gate 5"),        # OR Gate 6 output goes to OR Gate 5
    ("OR Gate 5", "NOT Gate 0")        # OR Gate 5 output goes to NOT Gate 0
])

plt.figure(figsize=(10, 6))
pos = nx.spring_layout(dependency_graph)
nx.draw(
    dependency_graph,
    pos,
    with_labels=True,
    node_color="skyblue",
    node_size=2000,
    edge_color="gray",
    font_size=10
)
```

```python
undirected_graph = dependency_graph.to_undirected()

partition = nx.algorithms.community.kernighan_lin_bisection(undirected_graph)

plt.figure(figsize=(10, 6))
colors_b = ['red' if node in partition[0] else 'blue' for node in dependency_graph.nodes]
nx.draw(
    dependency_graph,
    pos,
    with_labels=True,
    node_color=colors_b,
    node_size=2000,
    edge_color="gray",
    font_size=10
)
plt.title("Partitioned Graph (b) Using VQE")
plt.show()

plt.figure(figsize=(10, 6))
colors_c = ['blue' if node in partition[0] else 'red' for node in dependency_graph.nodes]
nx.draw(
    dependency_graph,
    pos,
    with_labels=True,
    node_color=colors_c,
    node_size=2000,
    edge_color="gray",
    font_size=10
)
plt.title("Partitioned Graph (c) Using Kernighan-Lin")
plt.show()
```

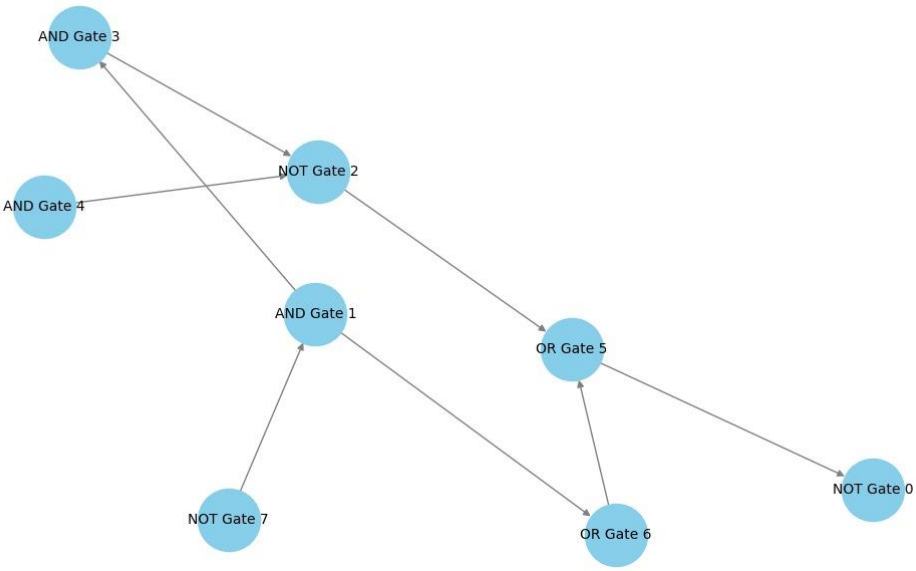

Figure 5: Toy circuit modeled into a graph

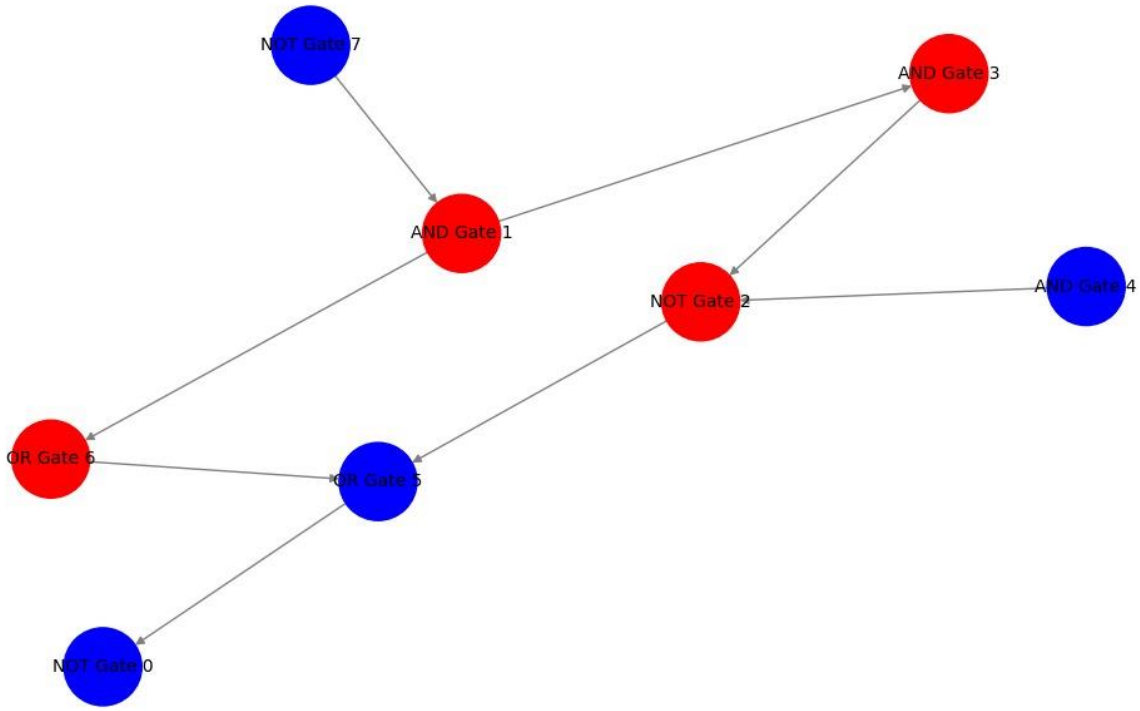

Figure 6: Initial circuit partition using Kernighan-Lin

Code for Variational quantum eigen solver (VQE) algorithm:

tempCodeRunnerFile.py > ...

```python
import networkx as nx
import matplotlib.pyplot as plt
from qiskit_aer import Aer
from qiskit.quantum_info import SparsePauliOp, Pauli
from qiskit_algorithms import VQE
from qiskit_algorithms.optimizers import COBYLA
from qiskit.circuit.library import RealAmplitudes
from qiskit.primitives import Estimator  # Updated import for estimator

# Build the Dependency Graph
dependency_graph = nx.DiGraph()

nodes = ["AND Gate 4", "AND Gate 1", "AND Gate 3", "NOT Gate 0", "NOT Gate 2", "NOT Gate 7", "OR Gate 6", "OR Gate 5"]
dependency_graph.add_nodes_from(nodes)

dependency_graph.add_edges_from([
    ("AND Gate 4", "NOT Gate 2"),     # AND Gate 4 output goes to NOT Gate 2
    ("AND Gate 1", "AND Gate 3"),     # AND Gate 1 output goes to AND Gate 3
    ("AND Gate 3", "NOT Gate 2"),     # AND Gate 3 output goes to NOT Gate 2
    ("NOT Gate 2", "OR Gate 5"),      # NOT Gate 2 output goes to OR Gate 5
    ("AND Gate 1", "OR Gate 6"),     # AND Gate 1 output goes to NOT Gate 7
    ("NOT Gate 7", "AND Gate 1"),      # NOT Gate 7 output goes to OR Gate 6
    ("OR Gate 6", "OR Gate 5"),       # OR Gate 6 output goes to OR Gate 5
    ("OR Gate 5", "NOT Gate 0")
])

# Convert graph to undirected
undirected_graph = dependency_graph.to_undirected()

# Using Kernighan-Lin for initial partition
partition = nx.algorithms.community.kernighan_lin_bisection(undirected_graph)
```

```python

# Plot the initial graph
plt.figure(figsize=(10, 6))
colors_b = ['red' if node in partition[0] else 'blue' for node in dependency_graph.nodes]
pos = nx.spring_layout(dependency_graph)
nx.draw(
    dependency_graph,
    pos,
    with_labels=True,
    node_color=colors_b,
    node_size=2000,
    edge_color="gray",
    font_size=10
)
plt.title("Partitioned Graph (b) Using Kernighan-Lin")
plt.show()

# defining a hamilton matrix for partitioning
def create_hamiltonian(graph):

    num_nodes = graph.number_of_nodes()
    pauli_list = []
    for edge in graph.edges():
        u, v = list(graph.nodes()).index(edge[0]), list(graph.nodes()).index(edge[1])
        # adding pouli operator to determining quantum interactions between the qubits,
        z_term = ['I'] * num_nodes
        z_term[u] = 'Z'
        z_term[v] = 'Z'
        pauli_list.append((''.join(z_term), 1.0))  # Add edge weight here default 1

    return SparsePauliOp.from_list(pauli_list)
```

```python
# applying  VQE algorithm  with Qiskit for better computation effective and time saving
hamiltonian = create_hamiltonian(undirected_graph)
num_qubits = len(undirected_graph.nodes)

optimizer = COBYLA(maxiter=250)
var_form = RealAmplitudes(num_qubits, reps=3)

# Use AerSimulator for quantum execution used for debugging and testing
backend = Aer.get_backend('statevector_simulator')

# Create an Estimator for quantum execution
estimator = Estimator()

# Setting up VQE with the correct estimator
vqe = VQE(ansatz=var_form, optimizer=optimizer, estimator=estimator)

# Execute the VQE
result = vqe.compute_minimum_eigenvalue(operator=hamiltonian)

#  results and plot the partitioned graph using VQE
print(result)

plt.figure(figsize=(10, 6))
colors_c = ['blue' if node in partition[0] else 'red' for node in dependency_graph.nodes]
nx.draw(
    dependency_graph,
    pos,
    with_labels=True,
    node_color=colors_c,
    node_size=2000,
    edge_color="gray",
```

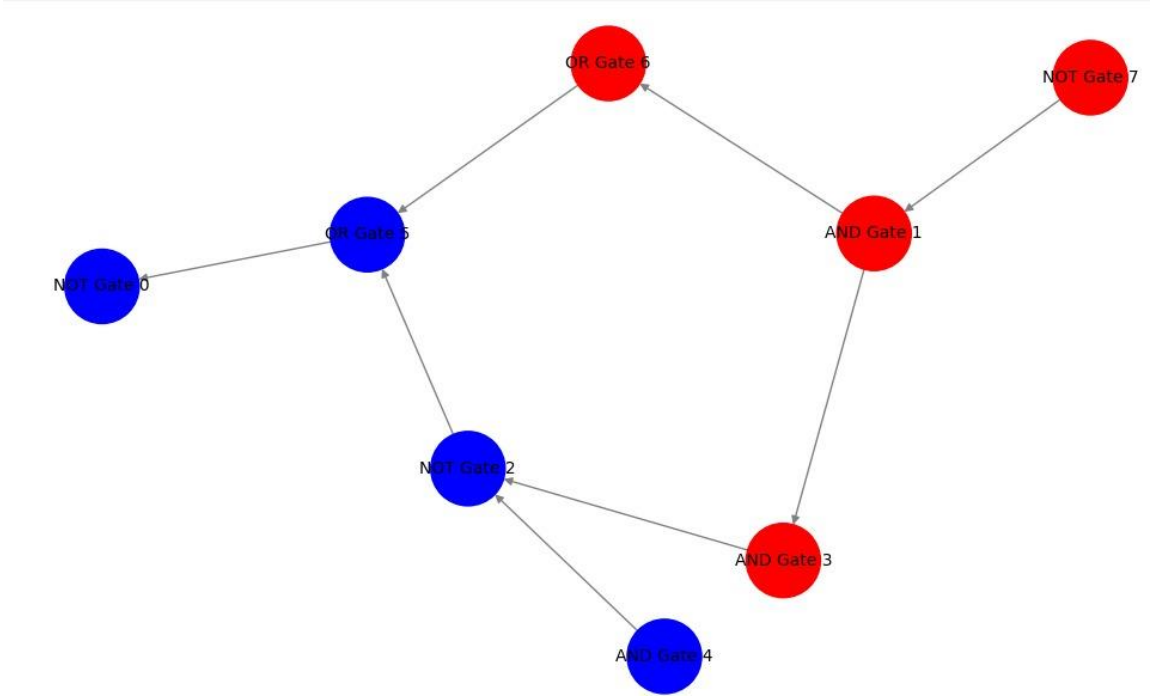

Figure 7: Initial partition of circuit using VQE to two subgraphs of 4 gates each

5.CONCLUSION

From the above two outputs we can say that the quantum machine learning will provide the best solutions for the vlsi placement problems than compared to the heuristic methods.

Quantum Machine Learning (QML) offers a promising frontier for optimizing VLSI placement, with hybrid quantum-classical approaches having the potential to revolutionize semiconductor design. Early successes in applying QML to simple circuits demonstrate its feasibility, though challenges remain, particularly related to optimization complexities and hardware limitations. As quantum technologies continue to mature, their integration into Electronic Design Automation (EDA) is expected to unlock unprecedented optimization capabilities, driving significant advancements in semiconductor design.

6. REFERENCES

[1].Aleksandrowicz, G., et al. Qiskit: “An Open-source Framework for Quantum Computing”

, Jan. 2019.

[2].Barkoutsos, P. K., Nannicini, G., Robert, A., Tavernelli, I., and Woerner, S. Improving Variational

Quantum Optimization using CVaR. Quantum 4 (Apr. 2020), 256. arXiv: 1907.04769.

[3].Farhi, E., Goldstone, J., and Gutmann, S. “A quantum approximate optimization algorithm”, 2014.

[4].Kahng, A. B., Lienig, J., Markov, I. L., and Hu, J. “VLSI Physical Design: From Graph Partitioning

to Timing Closure”. Springer Netherlands, 2011.

[5]. Isaac Turtletaub, George Li, Mohannad Ibrahim, and Paul Franzon. 2020. “Application of

Quantum Machine Learning to VLSI Placement” ACM/IEEE Workshop on Machine

Learning for CAD (MLCAD ’20), November 16–20, 2020.

[6]. Lucas, A. Ising formulations of many np problems. Frontiers in Physics 2 (2014).

[7].freecodecamp(youtube channel) ,quantum computing fundamentals (2019).

[8].qiskit(youtube channel),quantum machine learning fundamentals (2021).