QGEN: An Improved Approach to VLSI Placement Optimization

Team: Q Generation (Q 世代)

Team member:

Kai(王宣凱), Peter (徐培哲), Aaron (謝晉維), Tim (李泰岳), Owen (陳昀燦)

Outline

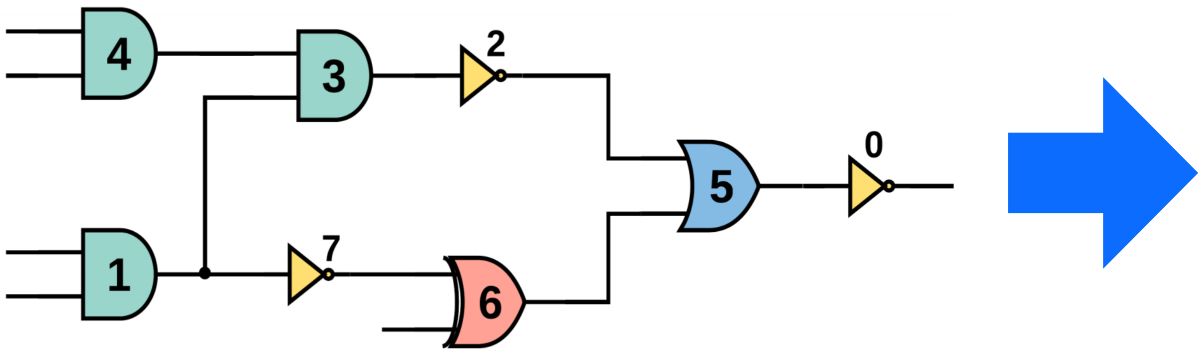
- Introduction of VLSI placement
- Existing Issues and Challenges
- Method design
- Workflow
- Results
- Applications
- Conclusion

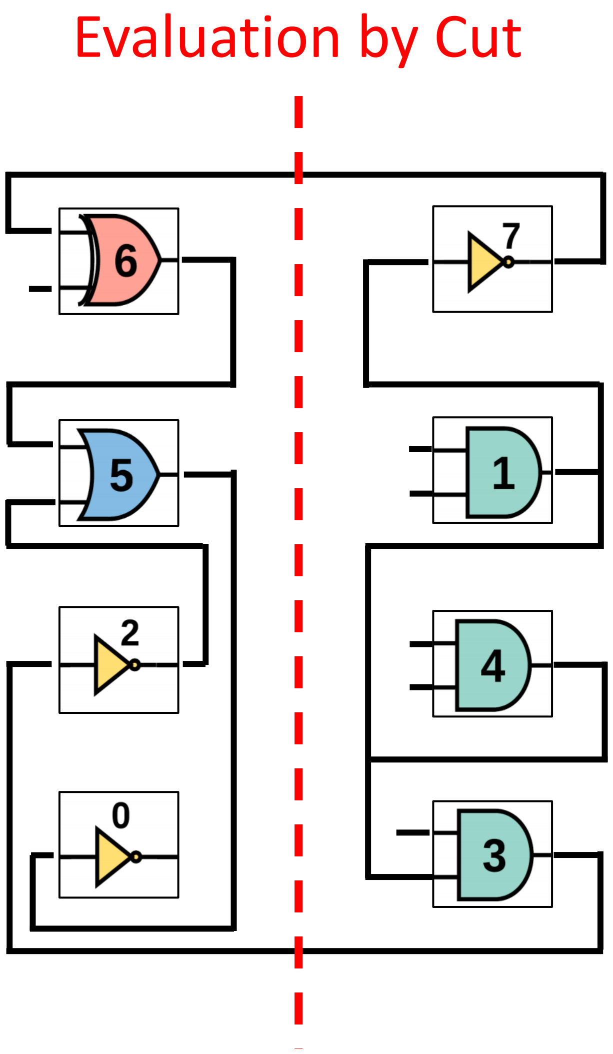
Outline

- ???????????????????????????????
- Existing Issues and Challenges
- Method design
- Workflow
- Results
- Applications
- Conclusion

VLSI (Very Large-Scale Integration) placement

Commonly encountered problem in Electronic Design Automation (EDA)





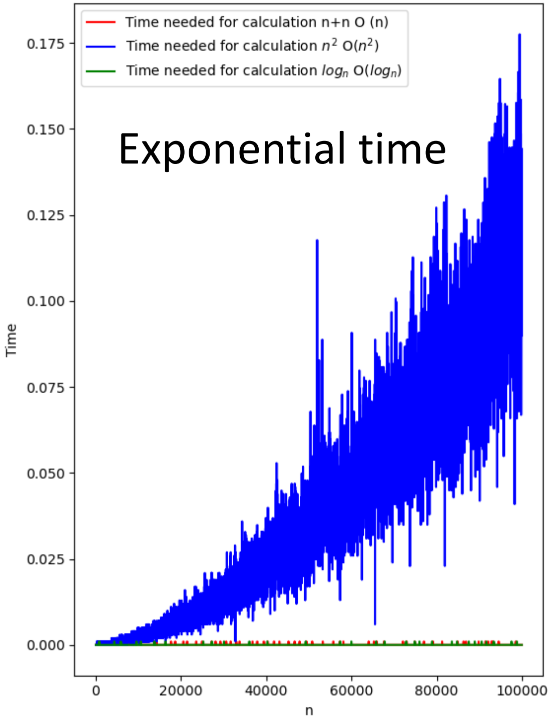
Create an integrated circuit using ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? to connect and ?

Outline

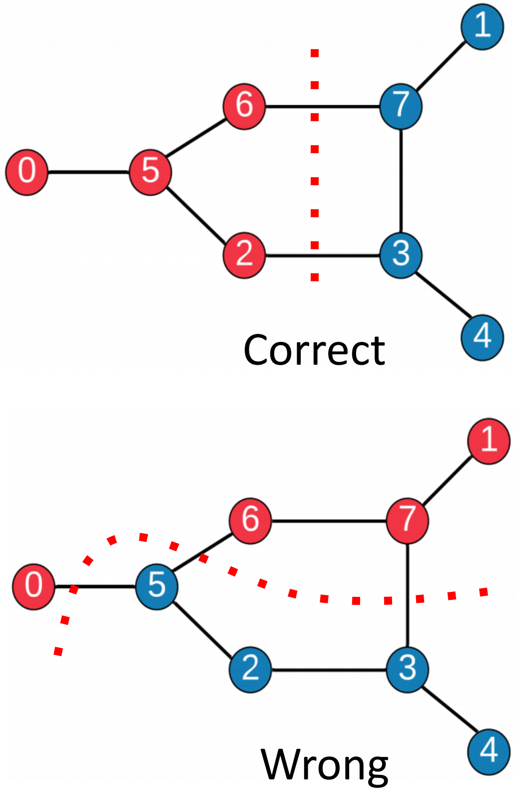
- Introduction of VLSI placement
- ????????????????????
  ???????????
- Method design
- Workflow
- Results
- Applications
- Conclusion

Existing Issues and Challenges

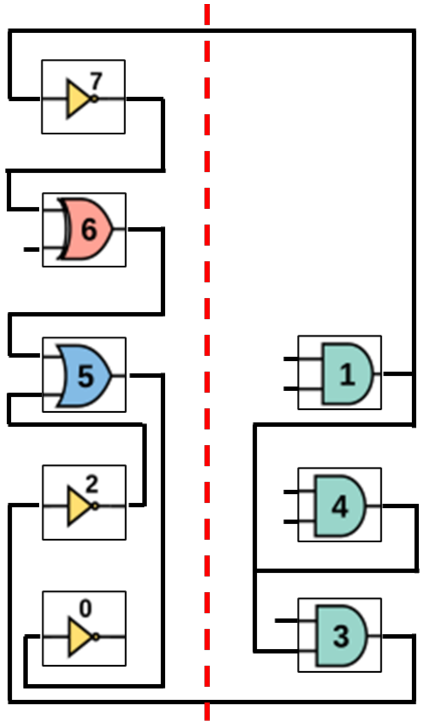
????????
???????



??????
??????

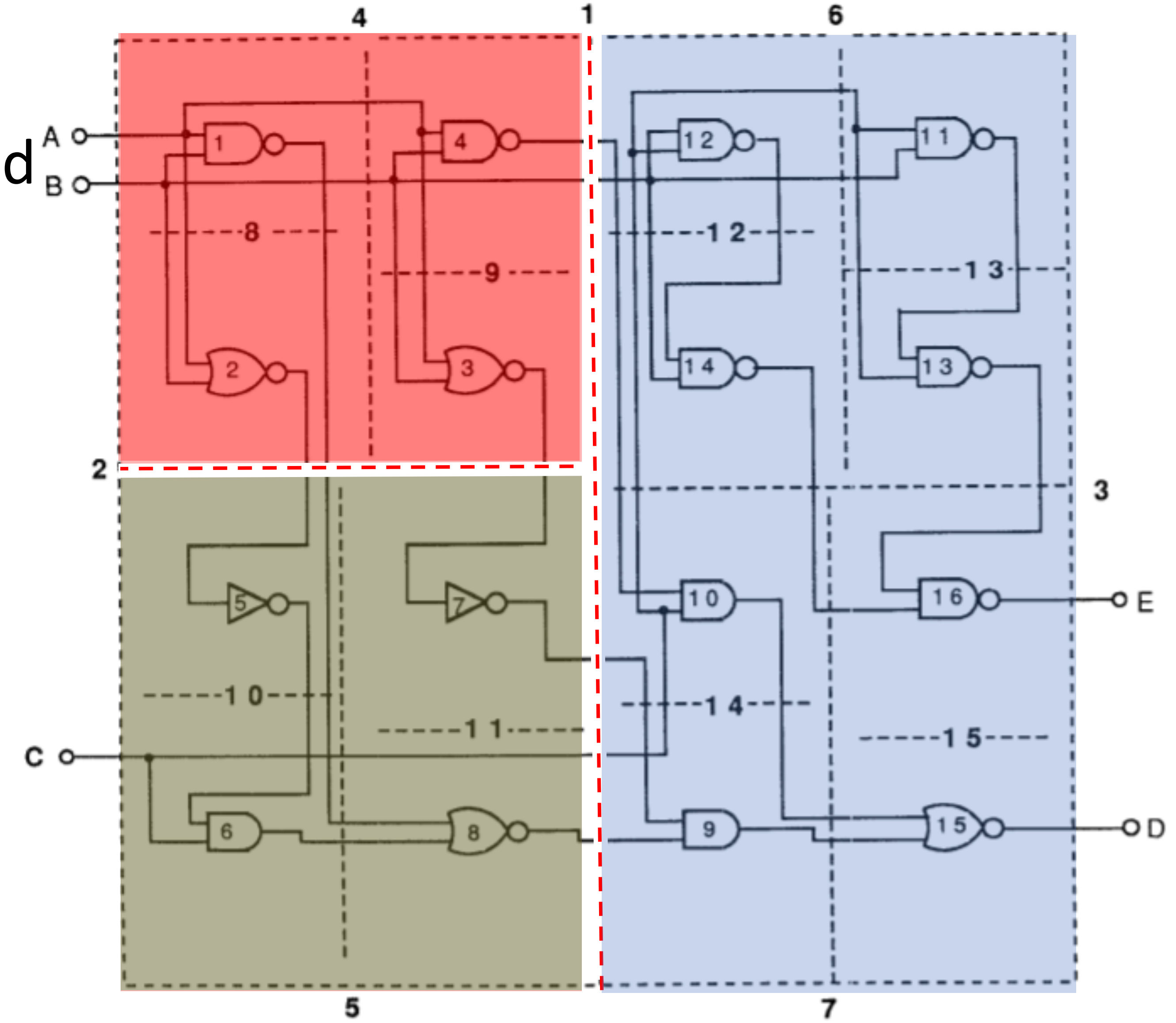


???
????????



Balanced Min-Cut is a good strategy

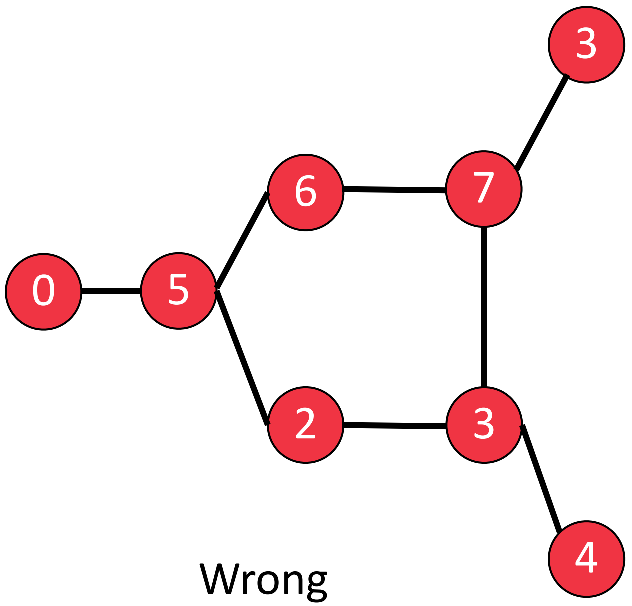
- Faster
- Can be implemented on NISQ device
- Scalability (Divide and conquer)



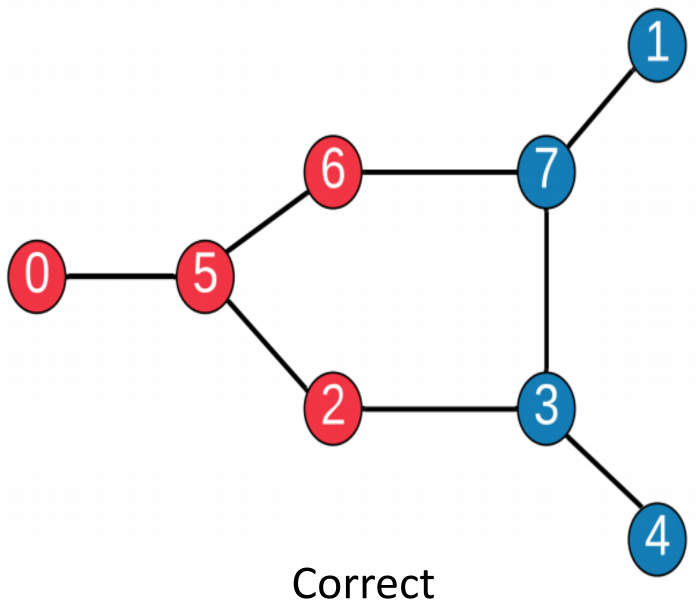
No cuts commonly occurred in state-of-the art[1]

- A Hamiltonian defined on Qiskit platform[2] for Balanced Min-Cut.

Stuck in local minima --> **No cuts**



Wrong



Correct

1. Turtletaub, Isaac, et al. "Application of Quantum Machine Learning to VLSI Placement.", doi:10.1145/3380446.3430644.
2. Qiskit. Quantum algorithms & applications in python, May 2020

Outline

- Introduction of VLSI placement
- Existing Issues and Challenges
- ???????????????
- Workflow
- Results
- Applications
- Conclusion

Method design

An Improved Min-Cut Algorithm by “Q generation”

$$H = H_A + H_B$$

- Original hamiltonian

$$H = \sum_{<ij>} \frac{1}{2}(1 - z_i z_j) + \sum_i (z_i)^2$$

- ???????????
  ????????????
  ??????

$$\mathrm{H} = \frac{1}{m}\sum_{i,j}^{m} \frac{1}{2}(1 - z_i z_j) + (\sum_{i=0}^{n-1} z_i)^2 \frac{a}{n^2}$$

Outline

- Introduction of VLSI placement
- Existing Issues and Challenges
- Method design
- ?????????
- Results
- Applications
- Conclusion

Workflow

1.Define the problem as a hamiltonian (Ising model)

$$H = \sum_{<ij>} \frac{1}{2}(1 - z_i z_j) + \sum_i (z_i)^2$$

$$H = \frac{1}{m}\sum_{i,j}^{m} \frac{1}{2}(1 - z_i z_j) + \left(\sum_{i=0}^{n-1} z_i\right)^2 \frac{a}{n^2}$$

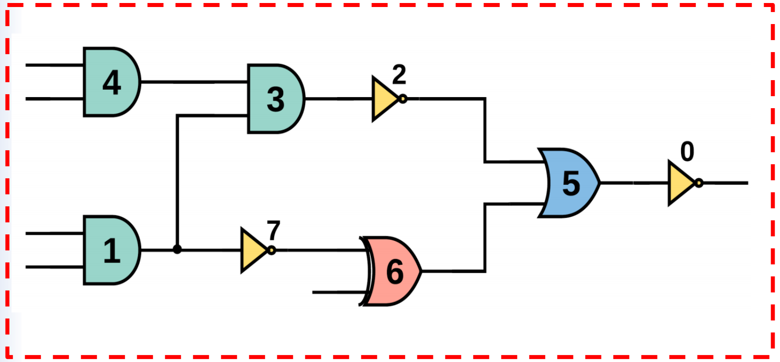
```
def normalized_h(adj_mtx,ratio):
    qubit_num = len(adj_mtx)
    iden = I
    for i in range(1,qubit_num):
        iden = iden^I
    op = iden-iden
    first_term = iden-iden
    m=0
    # for zizj term
    second_term = iden - iden
    for i in range(qubit_num):
        for j in range(qubit_num):
            if i>j:
                # 2ZiZj
                temp = np.ones(qubit_num)*I
                temp[i] = Z
                temp[j] = Z
                op_0 = temp[0]
                for k in range(1,qubit_num):
                    op_0 = op_0^temp[k]
                #print(op_0)
                second_term = second_term + 2*op_0
                # 0.5*I - 0.5*ZiZj
                if adj_mtx[i][j] == 1:
                    first_term = first_term + 0.5*iden - 0.5 * op_0
                    m+=1
    weighted_h = op + first_term/m + ratio/(qubit_num**2)*(second_term + qubit_num * iden)

    return weighted_h
```

Workflow

2.Transform the function into quadratic programming

8 qubits




```python
adj_mtx = [[0, 0, 0, 0, 0, 1, 0, 0]
          ,[0, 0, 0, 0, 0, 0, 0, 1]
          ,[0, 0, 0, 1, 0, 1, 0, 0]
          ,[0, 0, 1, 0, 1, 0, 0, 1]
          ,[0, 0, 0, 1, 0, 0, 0, 0]
          ,[1, 0, 1, 0, 0, 0, 1, 0]
          ,[0, 0, 0, 0, 0, 1, 0, 1]
          ,[0, 1, 0, 1, 0, 0, 1, 0]]
op_w= normalized_h(adj_mtx,1)
qp_w = QuadraticProgram()
qp_w.from_ising(op_w)
```

Workflow

3a. Classical method: NumPy Eingensolver

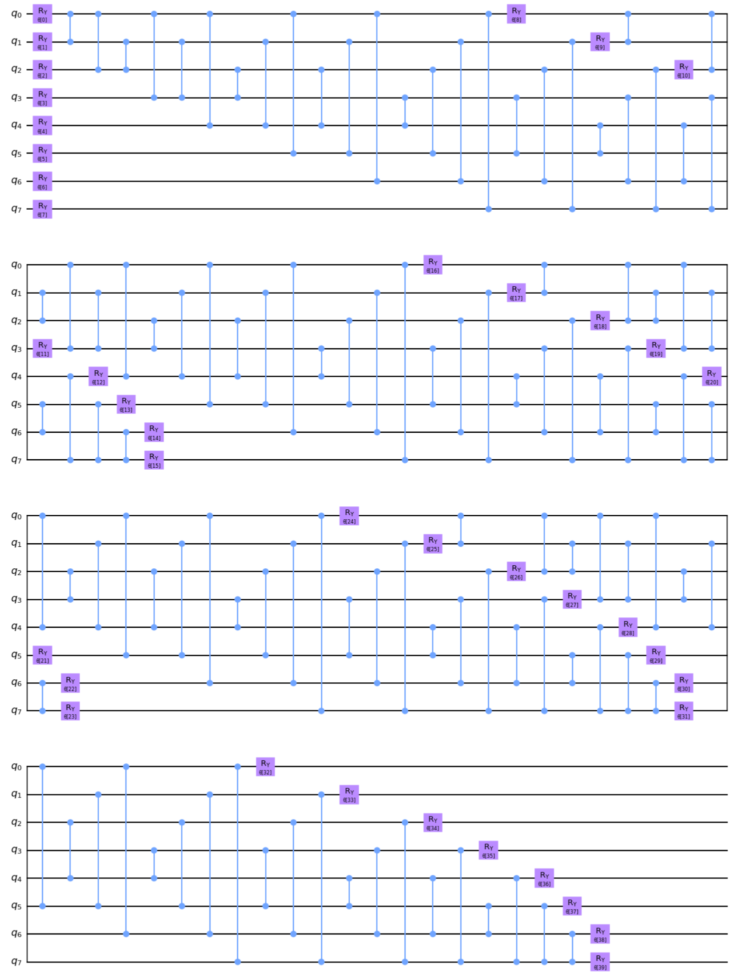
```
exact = MinimumEigenOptimizer(NumPyMinimumEigensolver())
result = exact.solve(qp)
print(result.prettyprint())
```

3b. Using???????????????
???????????to find the solution：

- VQE(Variational Quantum Eigensolver)
- QAOA(Quantum Approximate Optimization Algorithm)
- GAS(Grover Adaptive Search)

Workflow

4a.VQE & QAOA (ansatz:TwoLocoal)



Change the ‘reps’ layers (1~10)

Change optimizer (SLSQP 、COBYLA)

VQE

```
optimizer = SLSQP(maxiter=1000)
algorithm_globals.random_seed = 1234
seed = 132
num = 8
backend = Aer.get_backend('statevector_simulator')

ry = TwoLocal(num, 'ry', 'cz', reps=5, entanglement='full')
quantum_instance = QuantumInstance(backend=backend, seed_simulator=seed, seed_transpiler=seed)
vqe = VQE(ry, optimizer=optimizer, quantum_instance=quantum_instance)

vqe_meo = MinimumEigenOptimizer(vqe)

result = vqe_meo.solve(qp)

print(result.samples)
```

QAOA

```
adj_mtx = [[0, 0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 0, 0, 1], [0, 0, 0, 1, 0, 1, 0, 0], [0, 0, 1, 0, 1, 0, 0, 1], …
```

輸出已摺疊 …

```
optimizer = SLSQP(maxiter=1000)
algorithm_globals.random_seed = 1234
backend = Aer.get_backend('statevector_simulator')

quantum_instance = QuantumInstance(backend=backend, seed_simulator=seed, seed_transpiler=seed)
qaoa = QAOA(optimizer=optimizer, reps=3, quantum_instance=quantum_instance)
meo = MinimumEigenOptimizer(qaoa)

qaoa_meo = MinimumEigenOptimizer(qaoa) #please do not change this code

result = qaoa_meo.solve(qp)

print(result.samples) #please do not change this code
```

113]

Workflow

4b.GAS: Change iteration (1~8)

GROVER

```python
iter_arr = np.linspace(1,8,8)
ratio_arr = np.linspace(0.2,2,5)

for ratio_idx in range(len(ratio_arr)):
    for iter_idx in range(len(iter_arr)):

        ratio = ratio_arr[ratio_idx]
        iter_num = iter_arr[iter_idx]
        op_w = normalized_h(adj_mtx,ratio)
        qp = QuadraticProgram()
        qp.from_ising(op_w)

        grover_optimizer = GroverOptimizer(8, num_iterations=iter_num, quantum_instance=backend)
        results = grover_optimizer.solve(qp)
        print(results.prettyprint())

        prob = {}
        fval = {}
```

```python
        for i in range(len(result.samples)):
            tmp = ""
            count = 0
            for num in range(8):
                tmp += str(int(result.samples[i].x[num]))
                count += int(result.samples[i].x[num])

            prob[tmp] = result.samples[i].probability
            fval[tmp] = result.samples[i].fval
```

5.Compare the result with ???????????? and ?????????? (with ratio a)

Outline

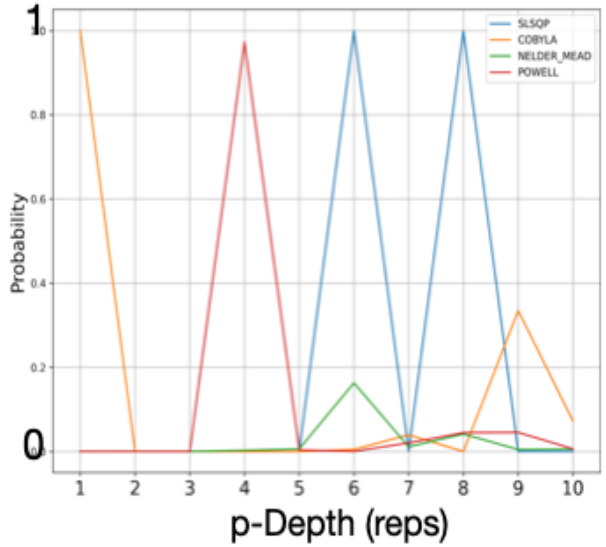
- Introduction of VLSI placement
- Existing Issues and Challenges
- Method design
- Workflow?
- ??????? ?
- Applications
- Conclusion

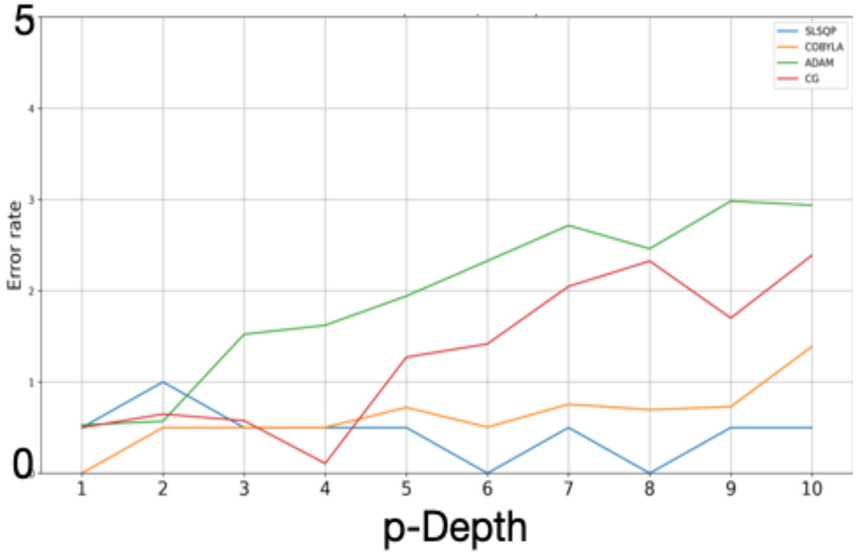
Results—Unnormalized Hamiltonian

Probability of Finding Right Answer

Error rate $= \frac{expectation - classical\ expectation}{|classical\ expectation|}$

VQE

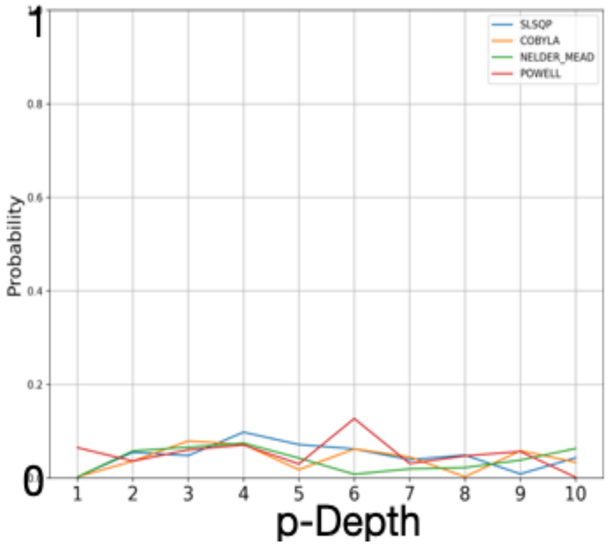


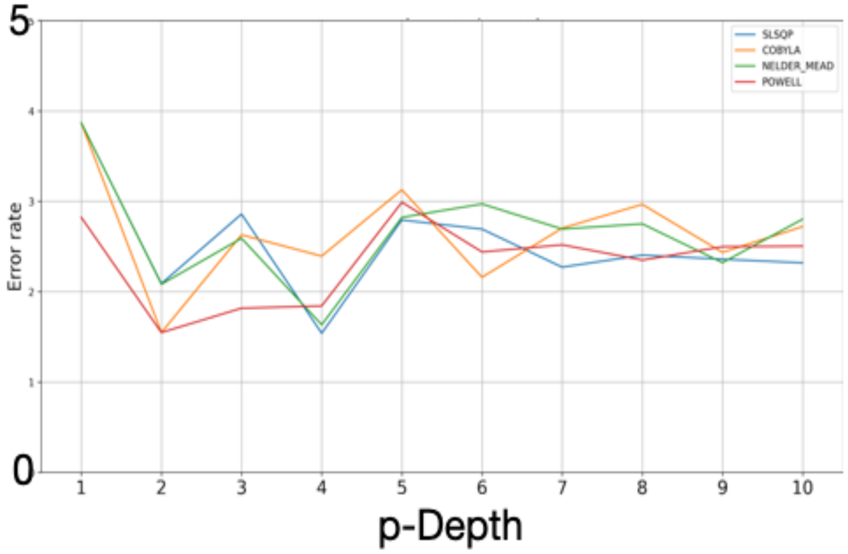


- Due to the number of parameter, VQE is better than QAOA

- The closer to 0 the error rate is, the better

QAOA





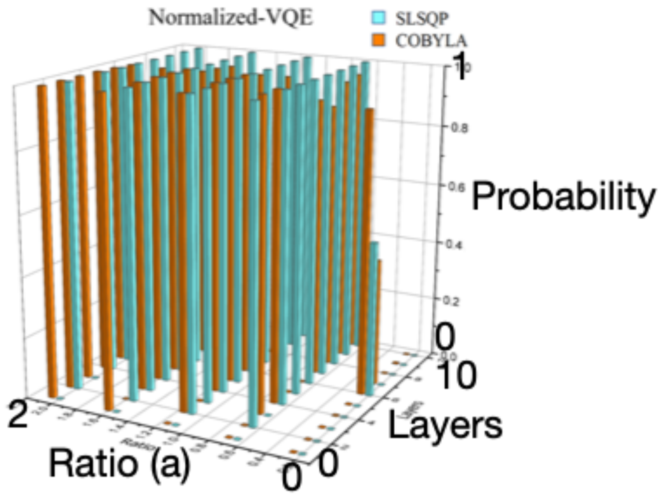
- SLSQP and COBYLA our-perform the others, so we choose it to compare with normalized Hamiltonian (QGEN)

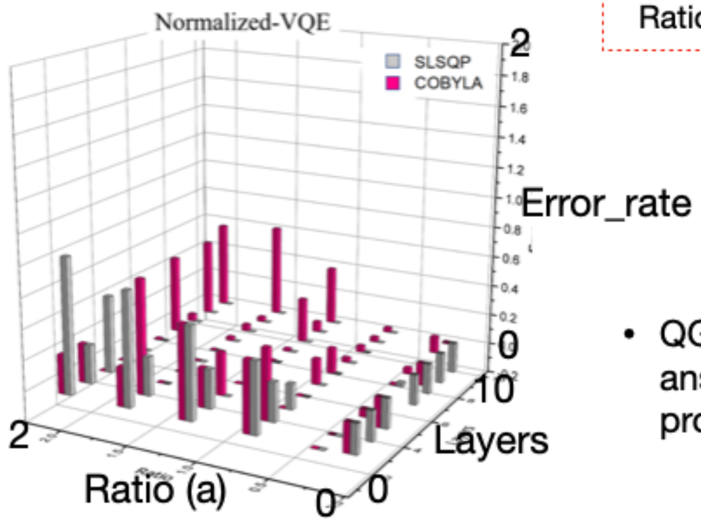
Results—Normalized Hamiltonian

Probability of Finding Right Answer

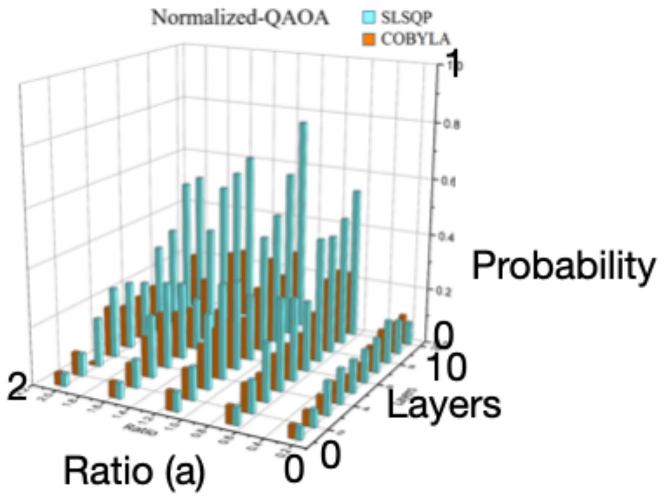
Error rate

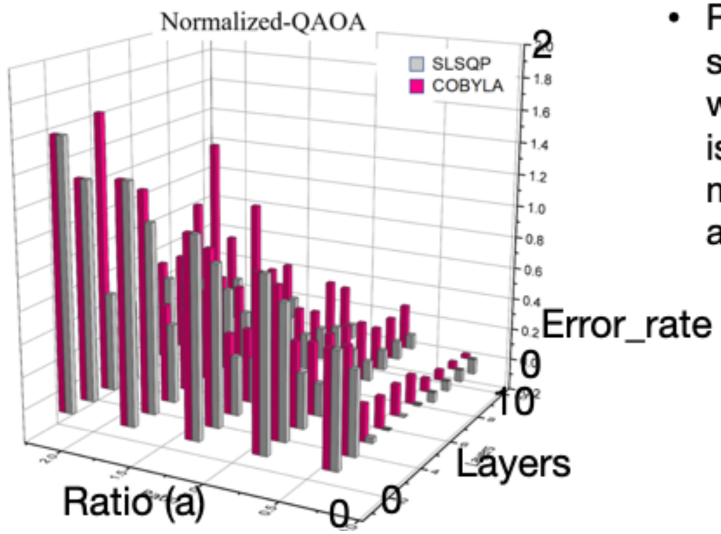
VQE





QAOA



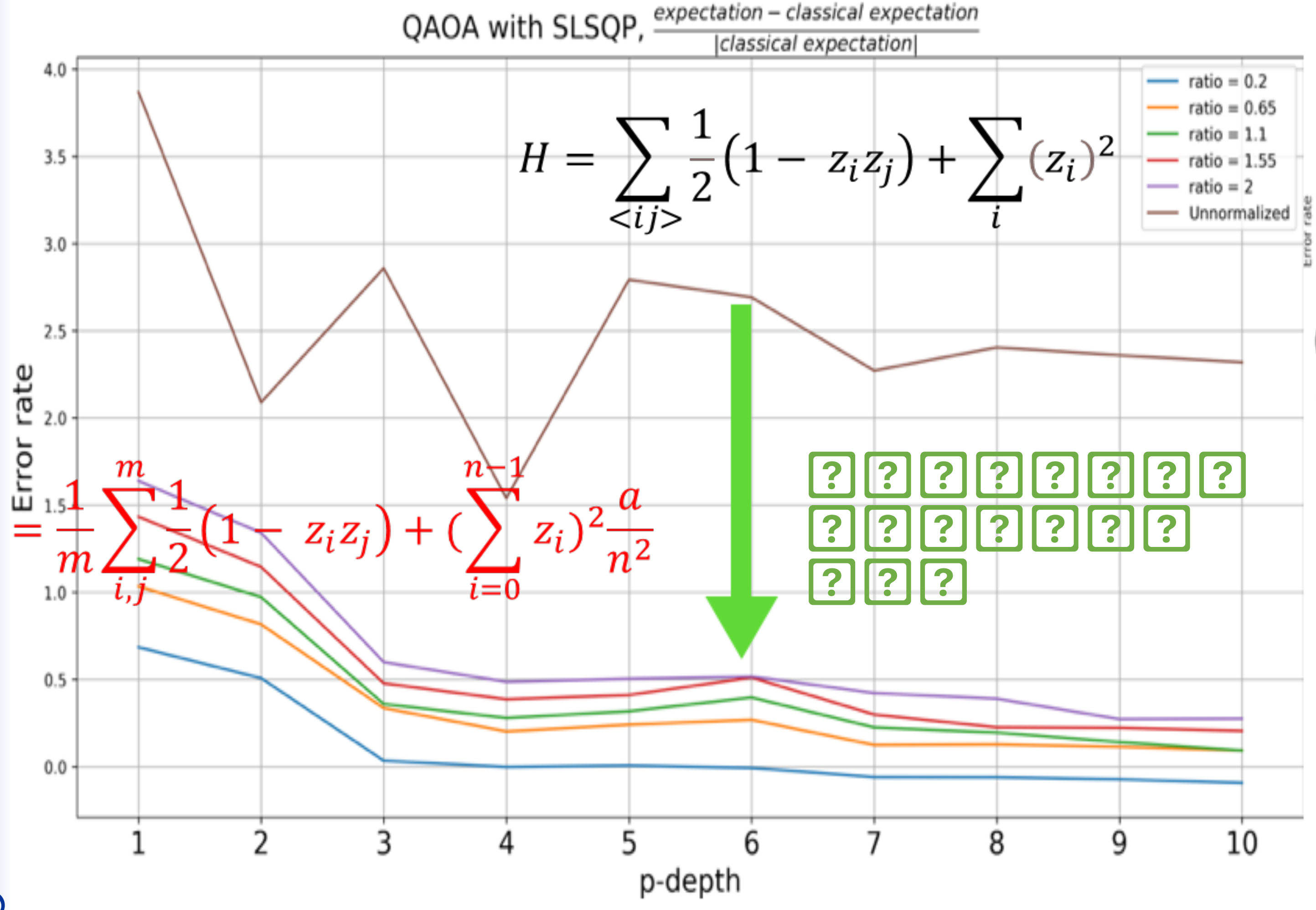


$$\mathrm{H} = \frac{1}{m}\sum_{i,j}^{m}\frac{1}{2}(1 - z_i z_j) + \Big(\sum_{i=0}^{n-1} z_i\Big)^2 \frac{a}{n^2}$$

Ratio = a = [0.2, 0.65, 1.1, 1.55, 2]

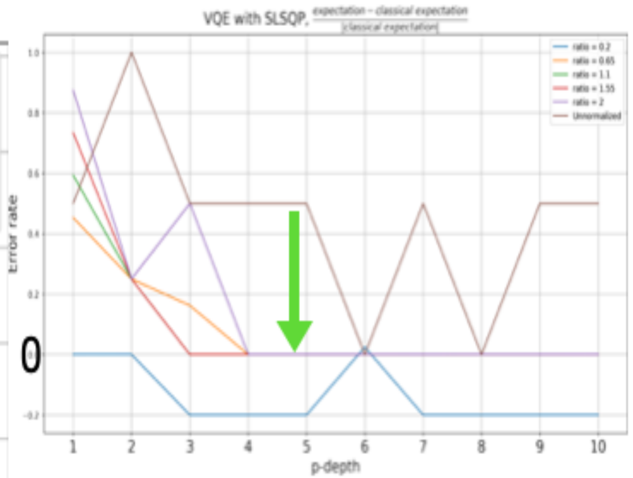
- QGEN can find the correct answer with high probability usually by VQE
- Ratio = 1.1 will be the best solution to the problem which means normalization is effective which means minimum cut and balanced are both important

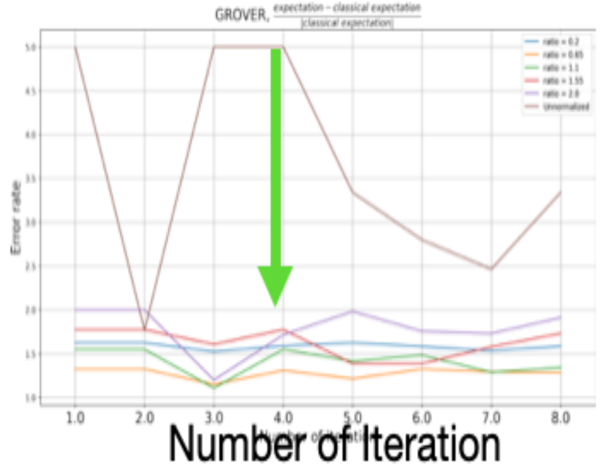
Results Comparison—QAOA/VQE/GROVER



$$H = \sum_{<ij>} \frac{1}{2}(1 - z_i z_j) + \sum_i (z_i)^2$$

$$\mathrm{H} = \frac{1}{m}\sum_{i,j}^{m} \frac{1}{2}(1 - z_i z_j) + (\sum_{i=0}^{n-1} z_i)^2 \frac{a}{n^2}$$

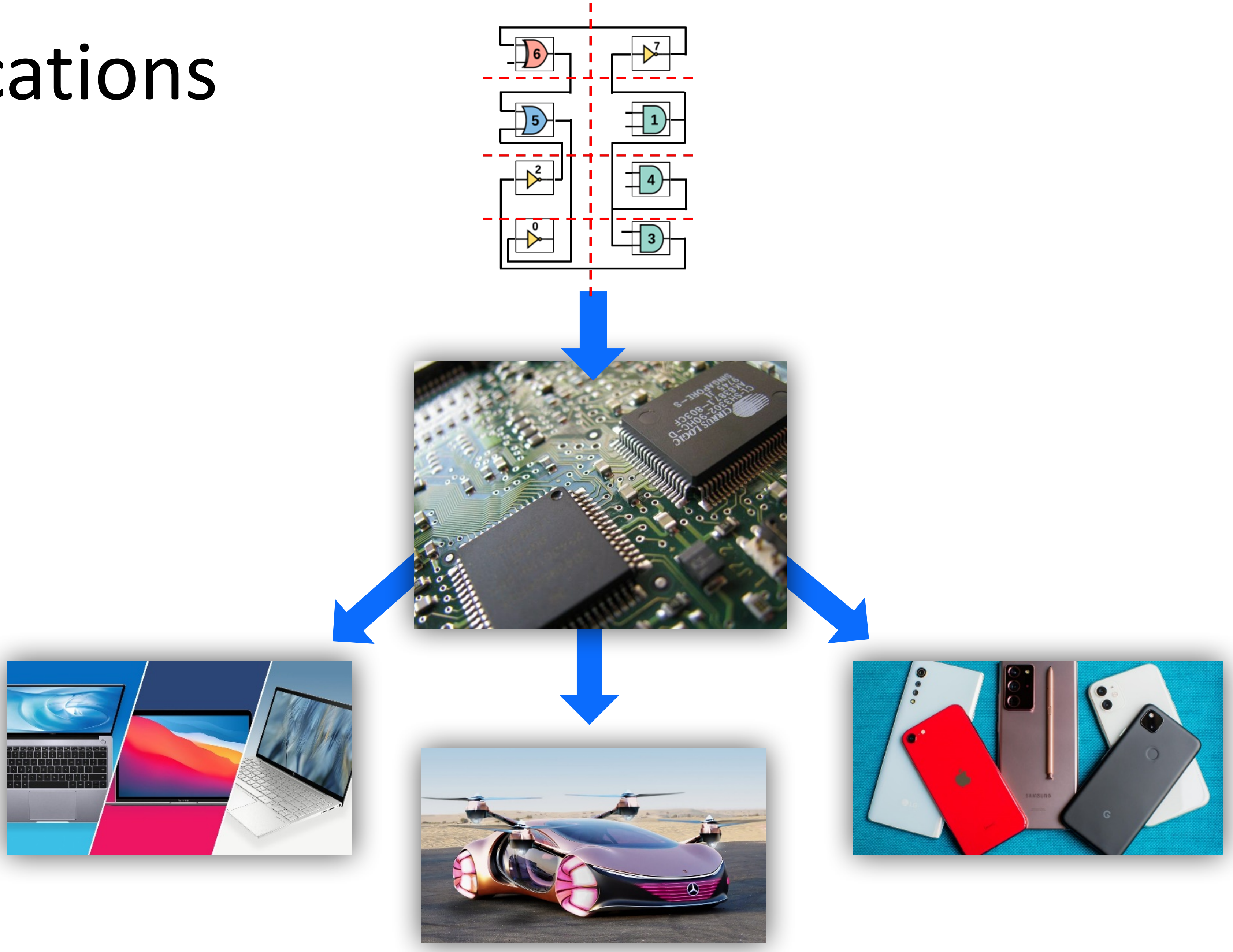




Outline

- Introduction of VLSI placement
- Existing Issues and Challenges
- Method design
- Workflow
- Results
- ?????????????
- Conclusion

Applications



Outline

- Introduction of VLSI placement
- Existing Issues and Challenges
- Method design
- Workflow
- Results
- Applications
- ???????????

Conclusion

- Breakthrough
  - Lower error rates
  - Alternative solutions
  - More balance
  - Robustness/stability

- Easy to use
  - Publicly Available code on GitHub
  - Visualizing results
  - Readable code and well-documented

Reference

- Qiskit. Quantum algorithms & applications in python, May 2020. https://github.com/Qiskit/qiskit-aqua Accessed 8-8-18.
- Barahona, Francisco, et al. “An Application of Combinatorial Optimization to Statistical Physics and Circuit Layout Design.” Operations Research, vol. 36, no. 3, 1988, pp. 493–513., doi:10.1287/opre.36.3.493.
- Turtletaub, Isaac, et al. “Application of Quantum Machine Learning to VLSI Placement.” Proceedings of the 2020 ACM/IEEE Workshop on Machine Learning for CAD, 2020, doi:10.1145/3380446.3430644.
- Author Cadence PCB Solutions, et al. “VLSI Technology: Its History and Uses in Modern Technology.” VLSI Technology: Its History and Uses in Modern Technology, 17 Mar. 2022, resources.pcb.cadence.com/blog/2020-vlsi-technology-its-history-and-uses-in-modern-technology#:~:text=VLSI%20refers%20to%20an%20integrated,gates%20or%20transistors%20per%20IC.