|Y⟩Quantum

BlueQubit Challenge

Team: QuantumETS GOAT

Pacôme Gasnier
Guy-Philippe Nadon
Axel Oneglia

Contents

|Y⟩Quantum

Overview

01 Circuits 1, 2 & 3

Circuit Approach

BlueQubit librairy-based solution

Peak Probability

X & Y measurements

Grover-Inspired Amplification

Peak Amplification

02 Circuit 4, 5 & 6

Differences (1000+ two-qubit gates)

Comparison of Quantum Backends

Pattern Identification

Circuit cutting

MPS / MPO Simulation

How to create a Peak

State Reconstruction

03 Creativc Extensions

Amplitude Amplification or Grover's Trick

Phase Kickback

Multi-basis State Tomography

ML Model Approach

Future Directions

Circuit 1, 2 & 3

Backends

```
dev = qml.device("default.qubit", wires=number_of_qubits,shots=1000)
```

```
dev = qml.device("bluequbit.cpu", wires=number_of_qubits, token=token, shots=shots)
```

```
Job zC9lRvWxRGXe37zm finished with status: FAILED_VALIDATION. Circuit contains more than 33 qubits, which is not supported for CPU backend with Pennylane.
```

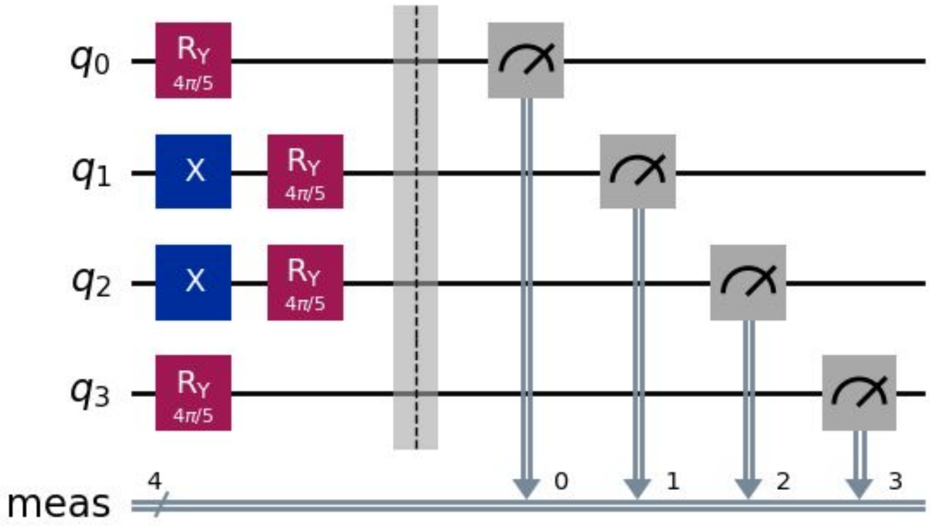
```
bq = bluequbit.init(token)

circuit = qasm2.load(qasm_path)
if not circuit.cregs:
    circuit.measure_all()

results = bq.run(
    circuits=circuit,
    device=device,
    shots=shots,
    asynchronous=False,
    job_name="sharp-peak-qasm"
    )
```

PENNYLANE

BlueQubit

Qiskit

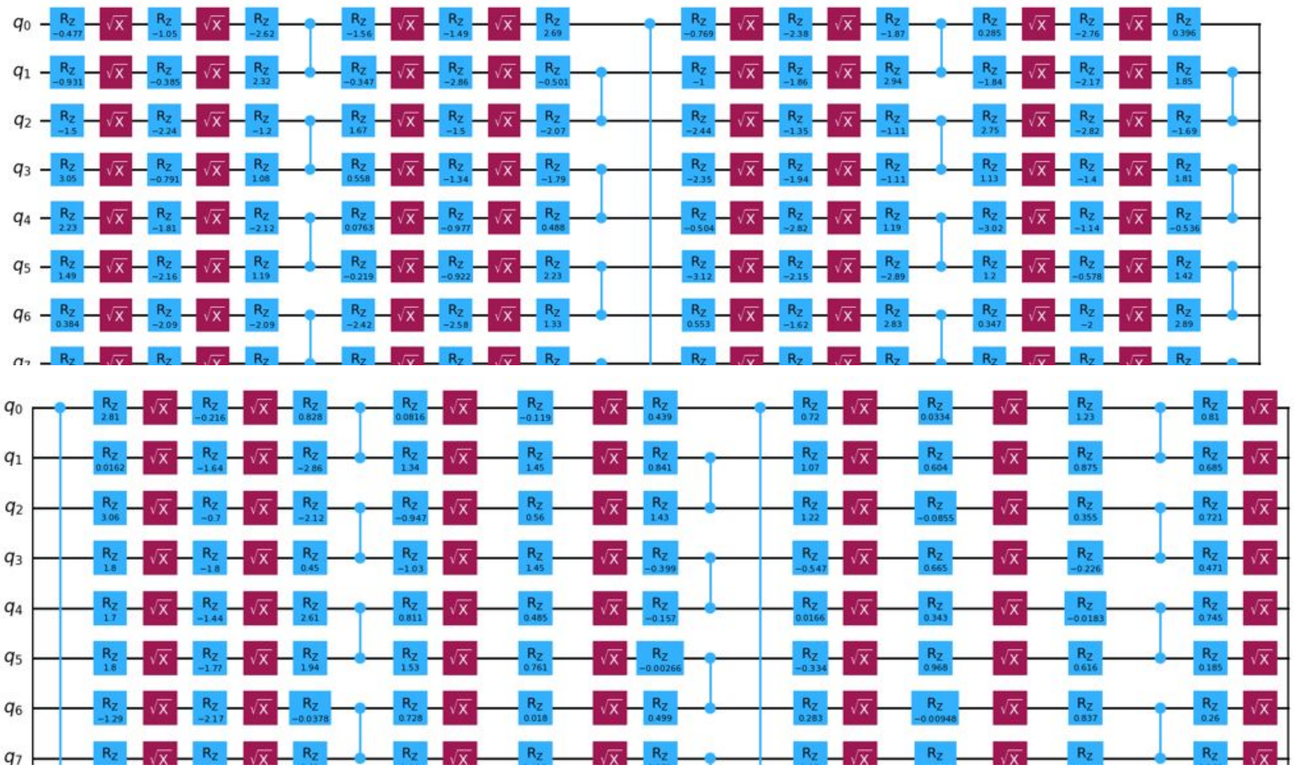
Circuit Approach



Circuit 1

- N qubits : 4
- Depth : 3
- Gates :
    - X : 2
    - Ry : 4

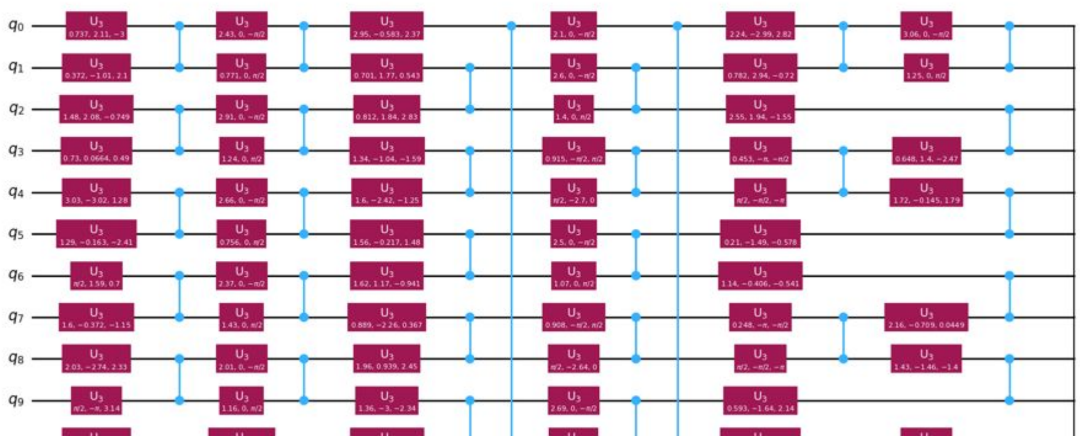
Circuit Approach



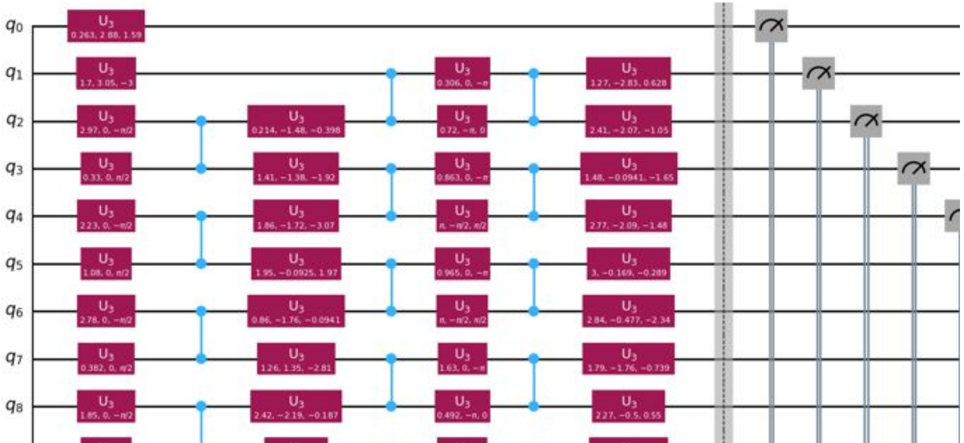
Part of circuit 2

- N qubits : 28
- Depth : 91
- Gates :
  - Rz : 1260
  - Sx : 840
  - Cz : 210

Circuit Approach



Part of circuit 3



- N qubits : 44
- Depth : 20
- Gates :
    - U3 : 399
    - Cz : 178

|Y⟩Quantum

Peak Probability

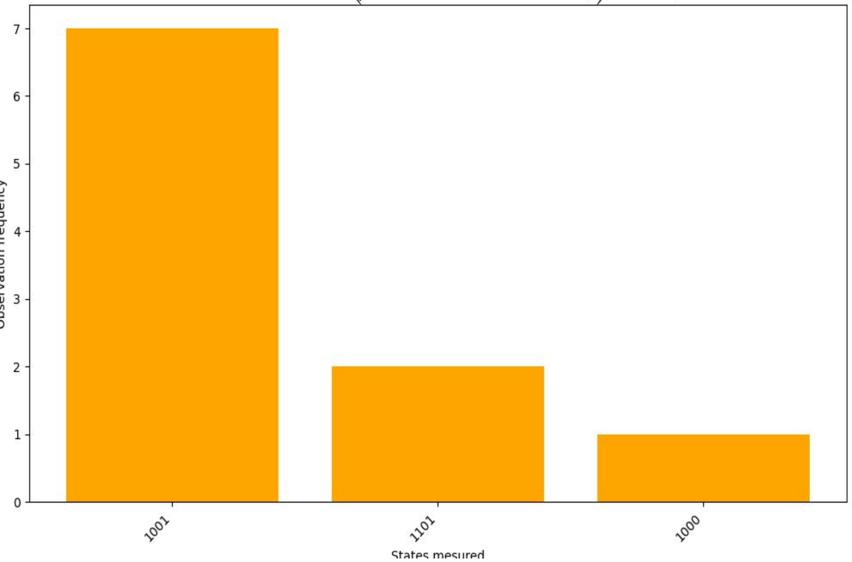
```python
import bluequbit
from qiskit import QuantumCircuit

DEVICE = 'mps.cpu'
SHOTS = 1000

qc_qiskit = QuantumCircuit.from_qasm_file(QASM_FILENAME)
qc_qiskit.measure_all()

print(qc_qiskit.draw(output='text'))

bq = bluequbit.init(API_KEY)

result = bq.run(qc_qiskit, device=DEVICE, shots=SHOTS)

counts = result.get_counts()
print(sorted(counts.items(), key=lambda item: item[1], reverse=True))
```

|Y⟩Quantum

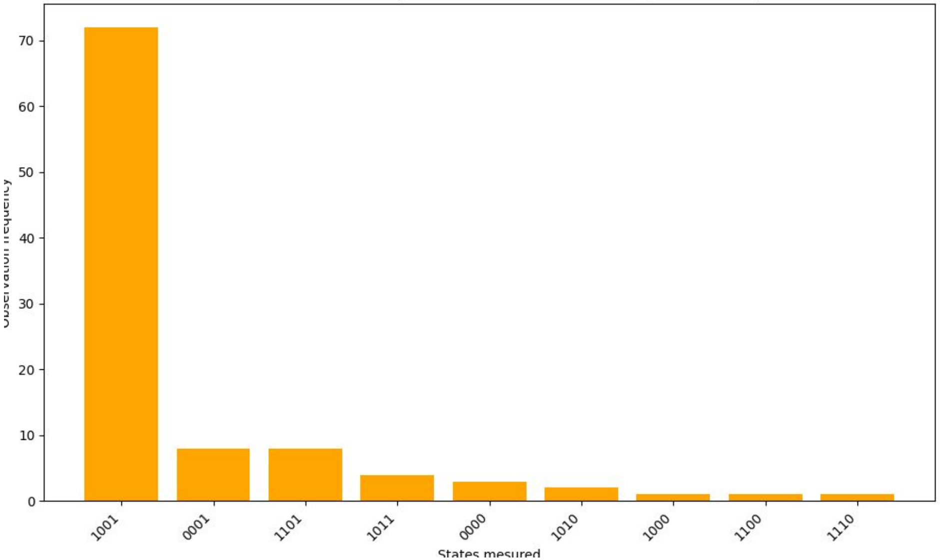
Peak Probability

Circuit 1

Distribution of the P1 quantum circuit measurements (for 10 shots)



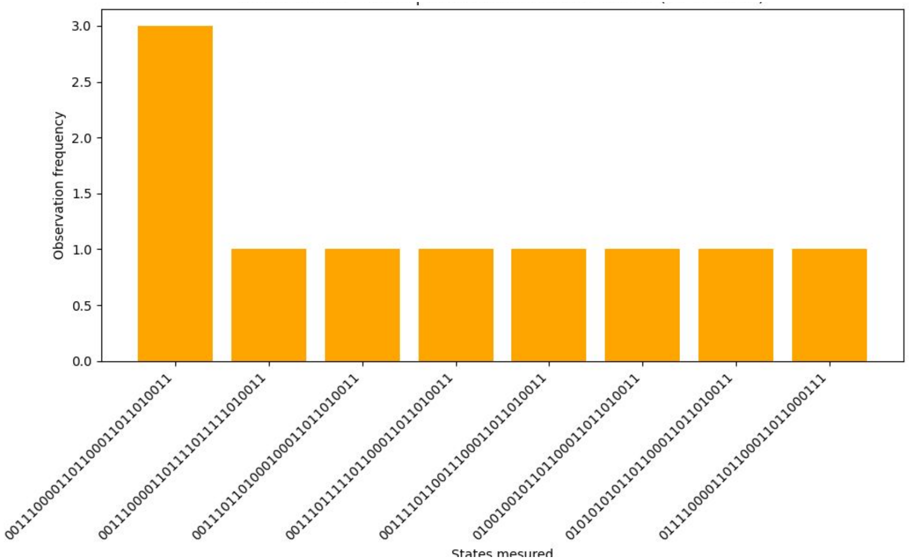
Distribution of the P1 quantum circuit measurements (for 100 shots)



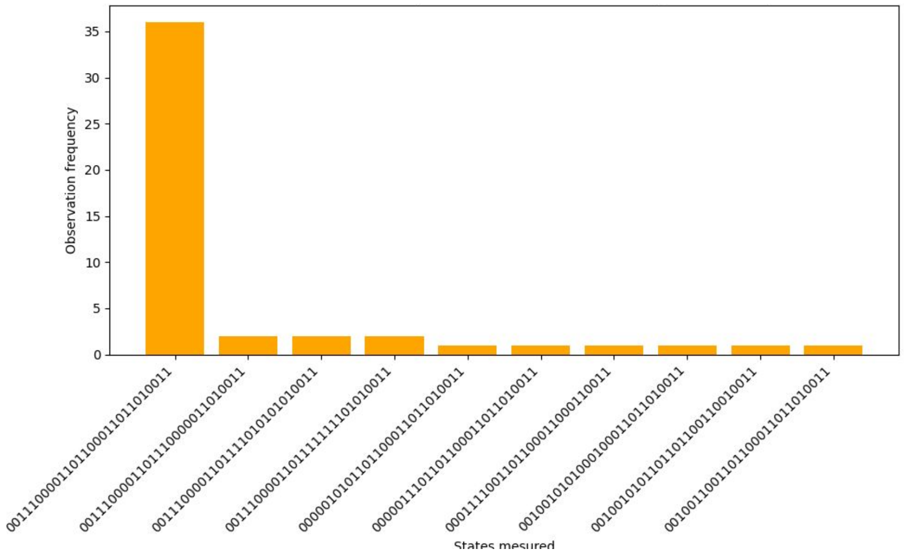
Peak Probability

Circuit 2

Distribution of the P2 quantum circuit measurements (for 10 shots)



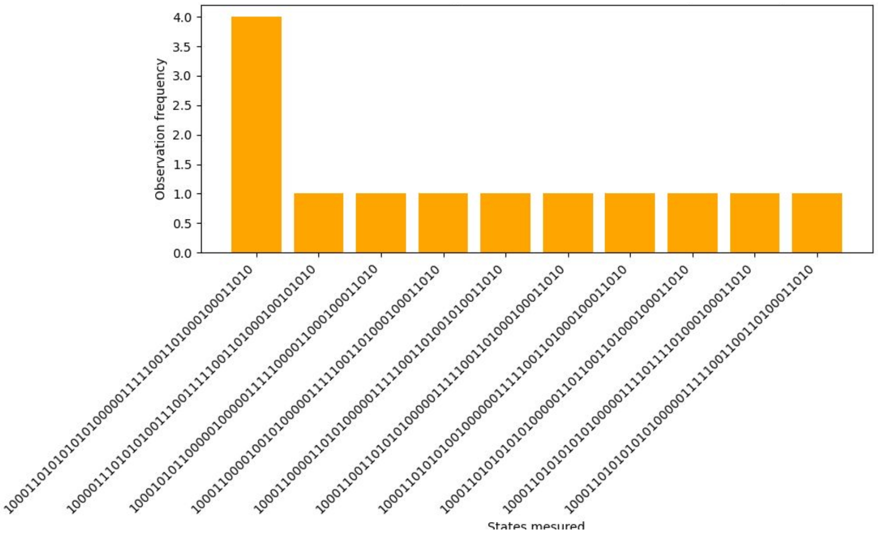
Distribution of the P2 quantum circuit measurements (for 100 shots)



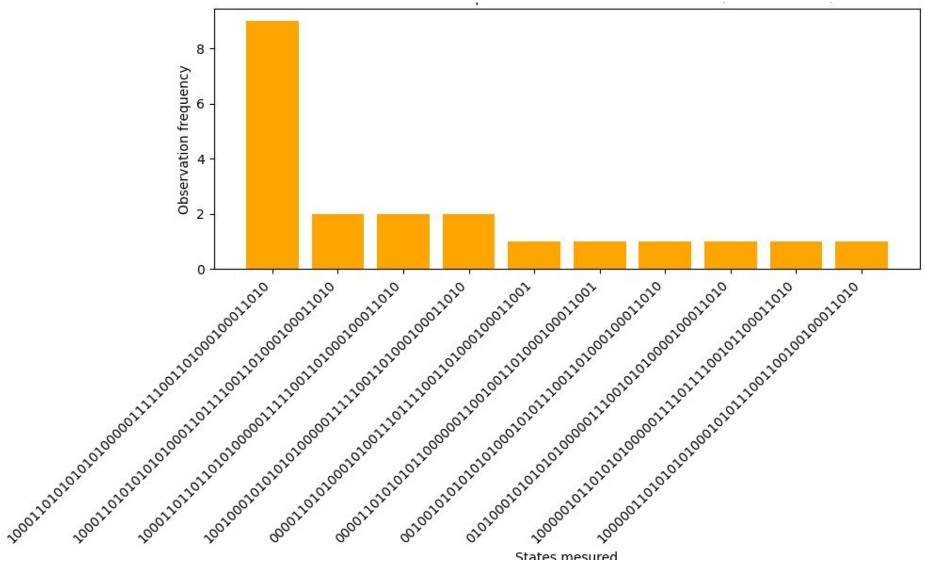
Peak Probability

Circuit 3

Distribution of the P3 quantum circuit measurements (for 25 shots)

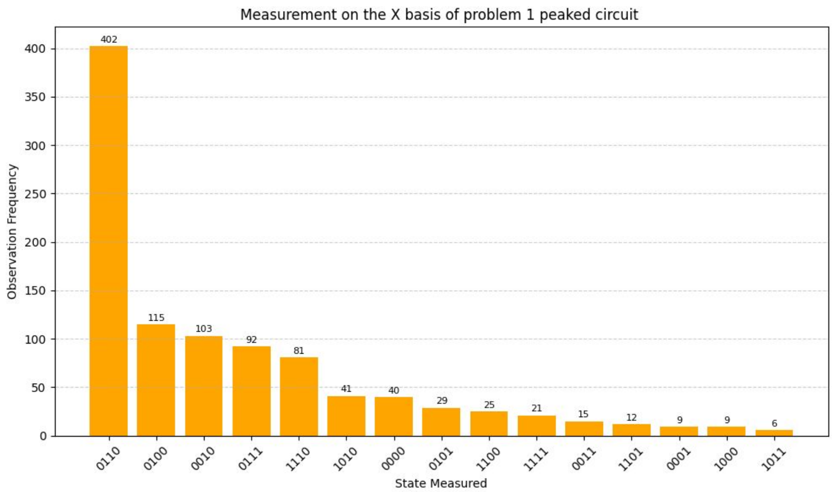


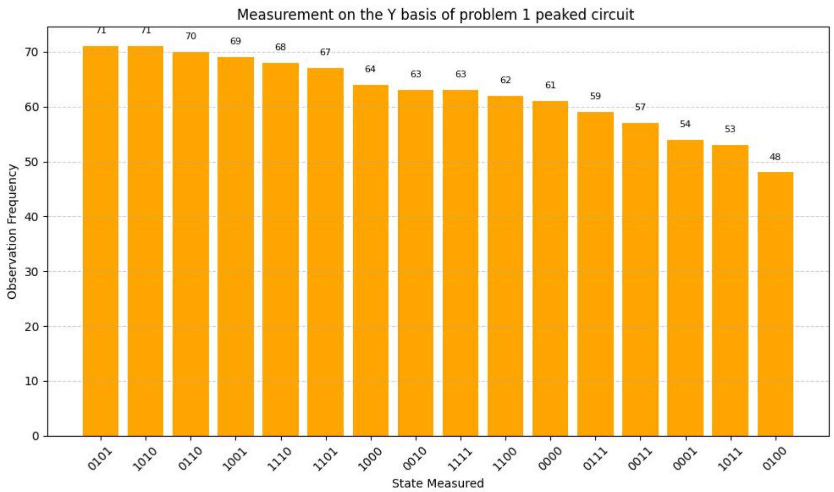
Distribution of the P3 quantum circuit measurements (for 100 shots)



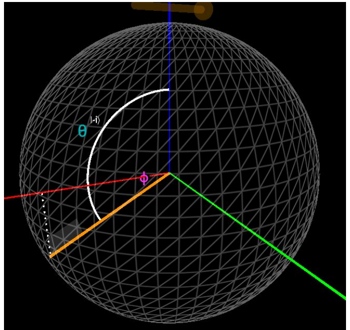
X & Y measurements

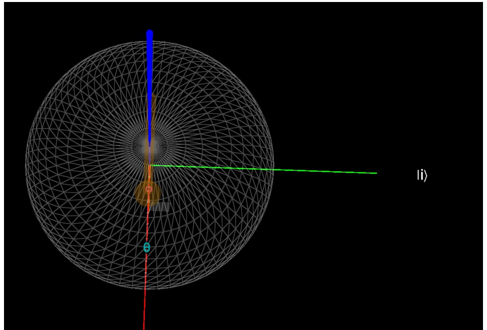
Circuit 1



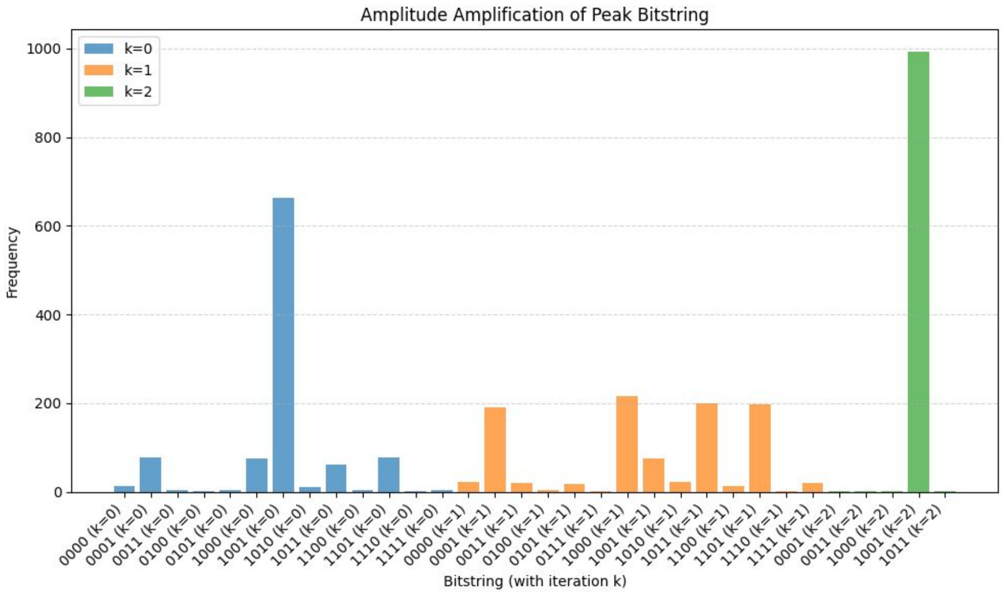


When measuring on the y axis, it endup being more similar to a hadamard/half chance of being i or -i





Grover-Inspired Amplification




```
@qml.qnode(dev)
def amplified_circuit(k):
    # Prepare the initial state
    state_prep(wires=range(n_system))
    # Apply k iterations of Q = -A S_0 A^\dagger O
    for _ in range(k):
        oracle()
        diffusion()
    return qml.counts()
```

Took inspiration from grover's algorithm to try to amplify the peak

```
def oracle():
    dim = 2 ** n_system
    diag = np.ones(dim, dtype=complex)
    target_index = int(peak_bitstring, 2)
    diag[target_index] = -1  # Phase flip for peak
    qml.DiagonalQubitUnitary(diag, wires=range(n_system))
```

```
def diffusion():
    qml.adjoint(state_prep)(wires=range(n_system))  # A^\dagger
    # S_0 = 2|0...0><0...0| - I
    dim = 2 ** n_system
    diag_S0 = -np.ones(dim, dtype=complex)
    diag_S0[0] = 1  # 1 for |0...0>, -1 elsewhere
    qml.DiagonalQubitUnitary(diag_S0, wires=range(n_system))
    state_prep(wires=range(n_system))  # A
```

```
Results for k=0 :
0000 → 13 times
0001 → 77 times
0011 → 4 times
0100 → 2 times
0101 → 4 times
1000 → 76 times
1001 → 664 times
1010 → 10 times
1011 → 61 times
1100 → 5 times
1101 → 77 times
1110 → 2 times
1111 → 5 times
```

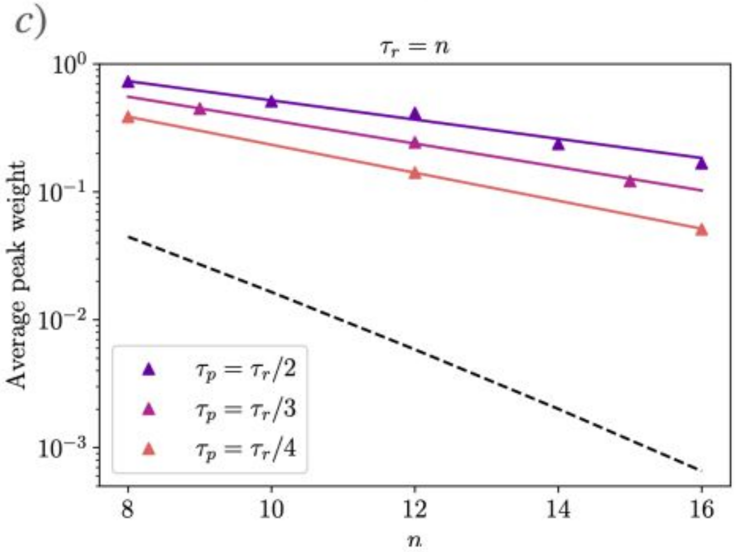
```
Results for k=1 :
0000 → 23 times
0001 → 191 times
0011 → 19 times
0100 → 3 times
0101 → 17 times
0111 → 1 times
1000 → 216 times
```

```
Results for k=2 :
0001 → 2 times
0011 → 1 times
1000 → 2 times
1001 → 993 times
1011 → 2 times
```

Peak Amplification

Conjecture 3.1 (Upper bound on average peak weight). *At* $\tau_r = \text{poly}(n)$ *and* $\tau_p = k\tau_r$, *where* $0 < k < 1$, *we have*

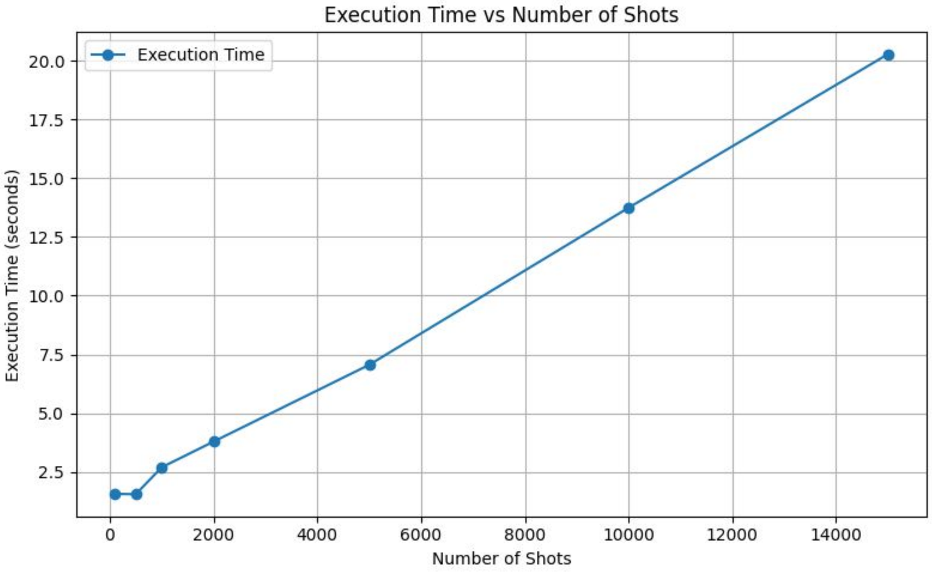
$$\textit{average peak weight} = O(\exp(-n)).$$



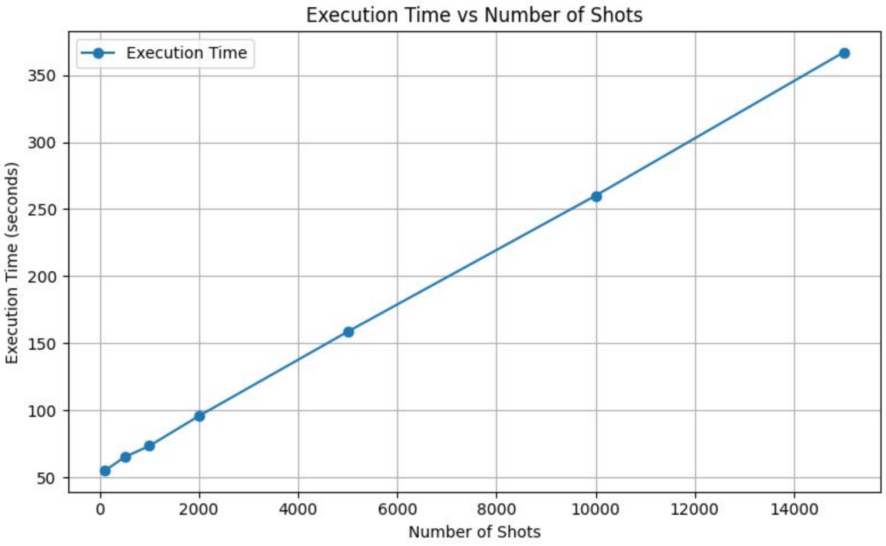
Can alter the non peak qubits' distribution as we can calculate an upper bound on average peak weights with the number of qubit and the proportion between the number of layer in the random quantum circuit compared to the parametrized quantum circuit

n = number of qubits
tr = number of random gates
tp = number of parametrized gates

Computation Time



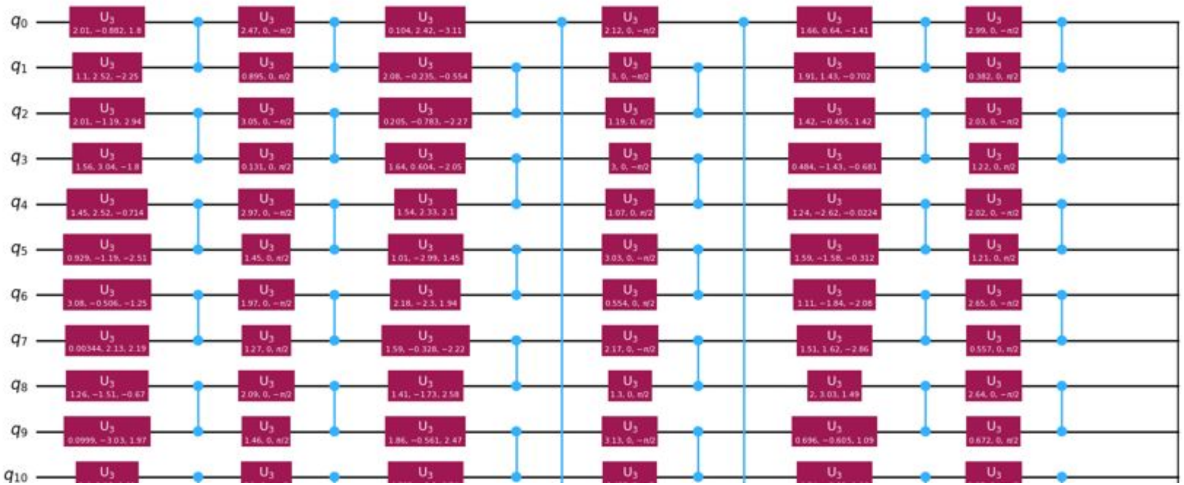
Circuit 1



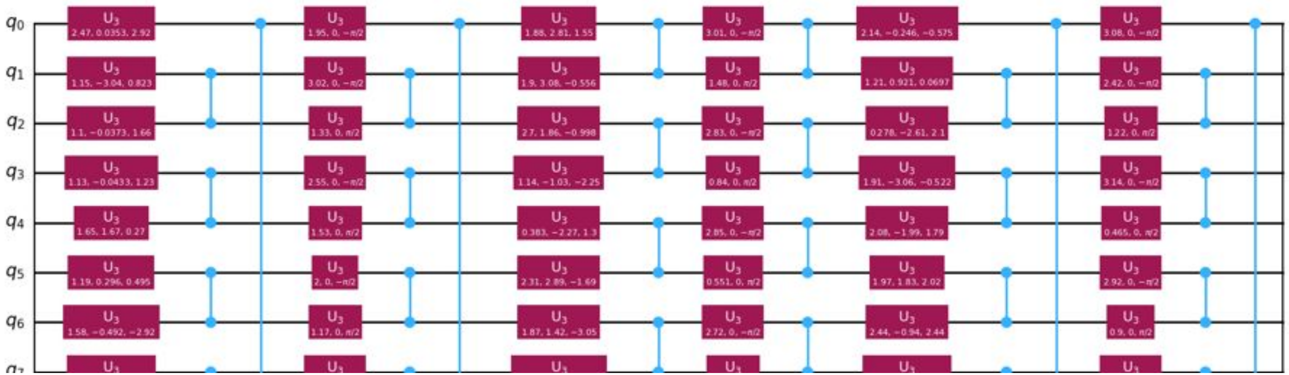
Circuit 2

Circuit 4, 5 & 6

Differences (1000+ two-qubit gates)



Part of circuit 4

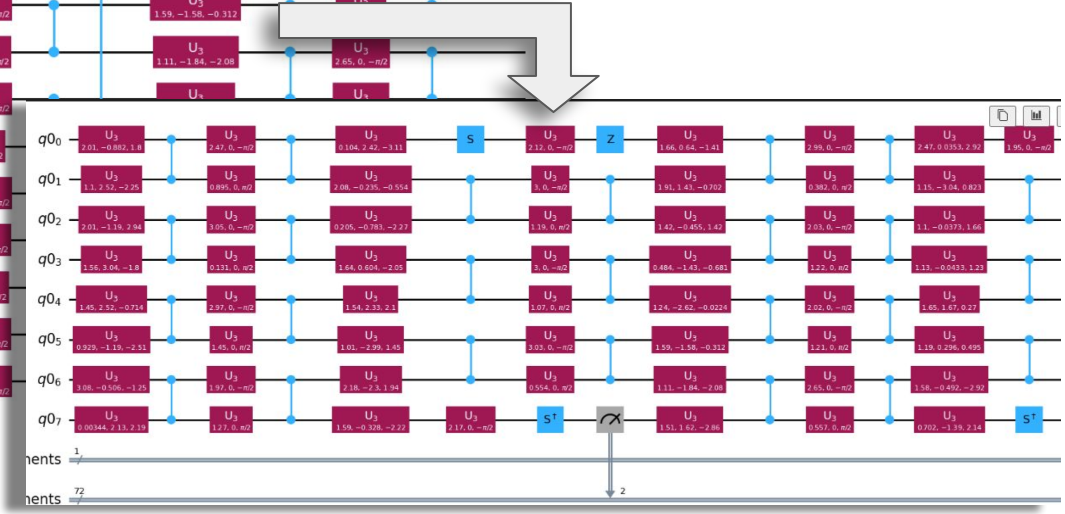


- N qubits : 48
- Depth : 438
- Gates :
  - U3 : 10240
  - Cz : **5096**

Circuit Cutting

```
BQJobNotCompleteError: Job l4gwmtD8lSMSU7yT finished with status: FAILED_VALIDATION. The number of two-qubit gates is too big for mps.cpu (maximum 1000).
```

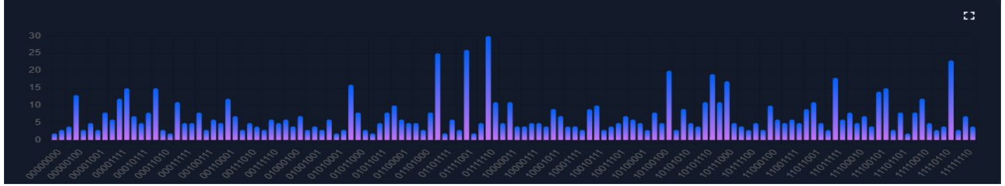
48 qubits is hard to simulate! We need to somehow be able to run it…



Circuit Cutting

```
Reconstructed expectation value: (-4.860285038568546e+299+0j)
```

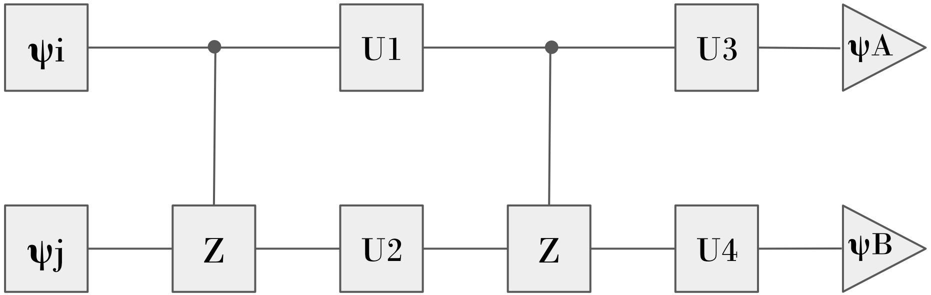
cut-subcircuit-group-999	oA3uc8NkDgtiGfxs	Completed	MPS.CPU	$0	4/13/2025, 12:35:52 AM	⋮
cut-subcircuit-group-998	IvwSDCVS62AyzB7N	Completed	MPS.CPU	$0	4/13/2025, 12:35:51 AM	⋮
cut-subcircuit-group-997	NO9VI2I5XgcOyvvm	Completed	MPS.CPU	$0	4/13/2025, 12:35:50 AM	⋮
cut-subcircuit-group-996	mAxjHGzw7pHSZepa	Completed	MPS.CPU	$0	4/13/2025, 12:35:49 AM	⋮
cut-subcircuit-group-995	QCmOYwFZUWQvMzaR	Completed	MPS.CPU	$0	4/13/2025, 12:35:48 AM	⋮
cut-subcircuit-group-994	E7dOOOAV3J4wFn2c	Completed	MPS.CPU	$0	4/13/2025, 12:35:47 AM	⋮
cut-subcircuit-group-993	9Vg3UrWNC95cILER	Completed	MPS.CPU	$0	4/13/2025, 12:35:46 AM	⋮
cut-subcircuit-group-992	ZD7UVdI5SosTgaqX	Completed	MPS.CPU	$0	4/13/2025, 12:35:45 AM	⋮
cut-subcircuit-group-991	6e6qbznUv6783Cys	Completed	MPS.CPU	$0	4/13/2025, 12:35:44 AM	⋮
cut-subcircuit-group-990	u2NPzUD7MuoPWcnj	Completed	MPS.CPU	$0	4/13/2025, 12:35:43 AM	⋮
cut-subcircuit-group-989	xg41WHVg9msBGzC8	Completed	MPS.CPU	$0	4/13/2025, 12:35:42 AM	⋮
cut-subcircuit-group-988	Ji1lFc5plwOKd2bK	Completed	MPS.CPU	$0	4/13/2025, 12:35:42 AM	⋮
cut-subcircuit-group-987	CkGOL8JuTvbq6jOF	Completed	MPS.CPU	$0	4/13/2025, 12:35:41 AM	⋮




```
Subcircuit 0 #0: counts = {'00000000': 10}
Added stitched: 00000000________________________________________ → 10
Subcircuit 0 #1: counts = {'00000000': 10}
Added stitched: 00000000________________________________________ → 10
Subcircuit 0 #2: counts = {'00000000': 10}
Added stitched: 00000000________________________________________ → 10
Subcircuit 0 #3: counts = {'00000000': 10}
Added stitched: 00000000________________________________________ → 10
Subcircuit 0 #4: counts = {'00000000': 10}
Added stitched: 00000000________________________________________ → 10
Subcircuit 0 #5: counts = {'00000000': 10}
Added stitched: 00000000________________________________________ → 10
Subcircuit 0 #6: counts = {'00000000': 10}
Added stitched: 00000000________________________________________ → 10
Subcircuit 0 #7: counts = {'00000000': 10}
Added stitched: 00000000________________________________________ → 10
Subcircuit 0 #8: counts = {'00000000': 10}
Added stitched: 00000000________________________________________ → 10
Subcircuit 0 #9: counts = {'00000000': 10}
Added stitched: 00000000________________________________________ → 10
Subcircuit 1 #0: counts = {'00000000': 10}
Added stitched: ________00000000________________________________ → 10
Subcircuit 1 #1: counts = {'00000000': 10}
Added stitched: ________00000000________________________________ → 10
Subcircuit 1 #2: counts = {'00000000': 10}

Subcircuit 5 #9: counts = {'00000000': 10}
Added stitched: ________________________________________00000000 → 10
```

Pattern Identification

To reduce the depths of the circuit we searched for patterns that we could simplify

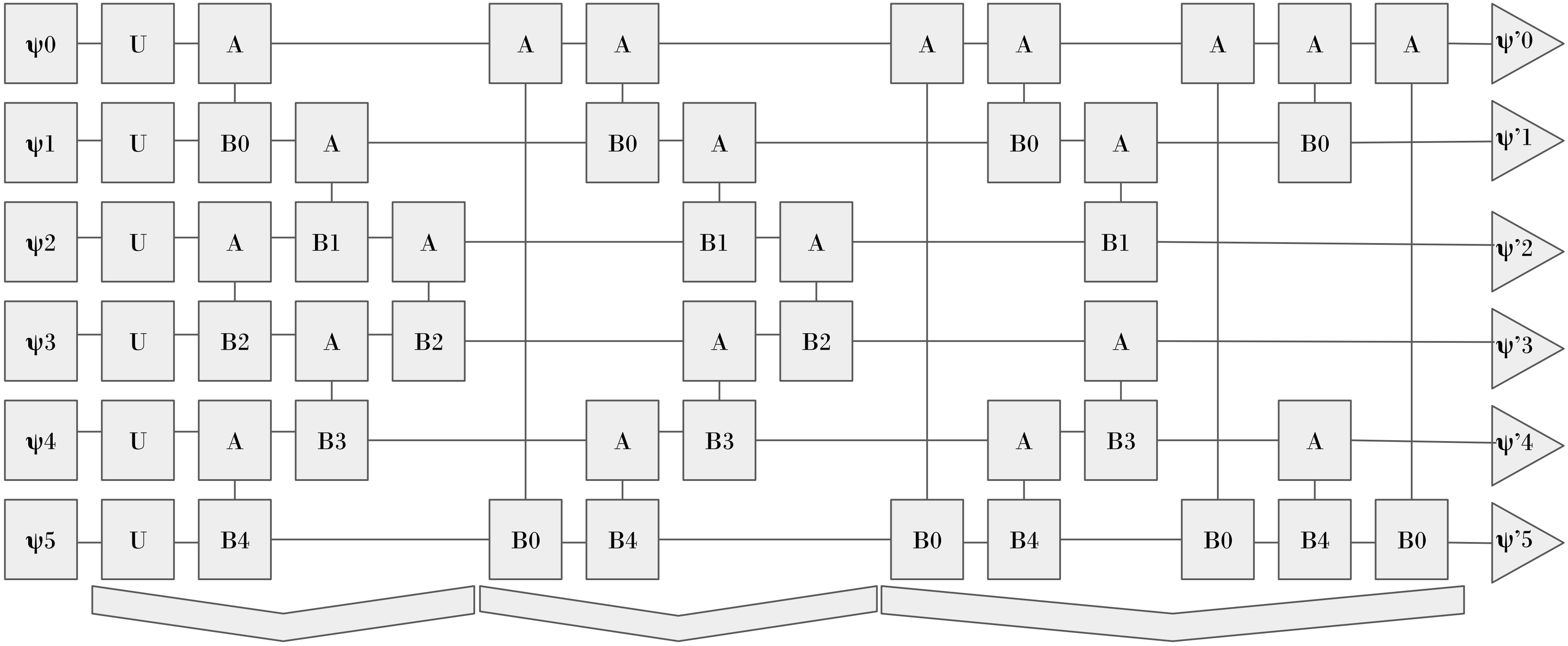


Circuit 4 : Pattern

$$|\psi_A\rangle = |\psi_1\rangle U_1 U_3 \longrightarrow \boxed{A}$$

$$|\psi_B\rangle = |\psi_2\rangle CZ_{|\psi_1\rangle} U_2 CZ_{|\psi_1\rangle U_1} U_4 \longrightarrow \boxed{Bi}$$

Pattern Identification : circuit 4



RQC

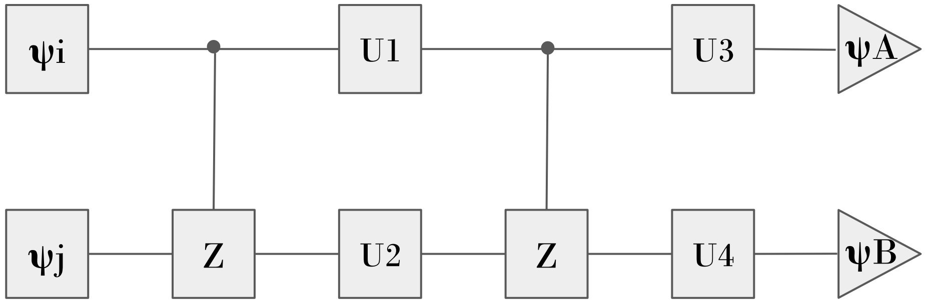
PQC Pattern repeated approx. 40 times

PQC Pattern until the end of the circuit

Pattern Identification : circuit 5

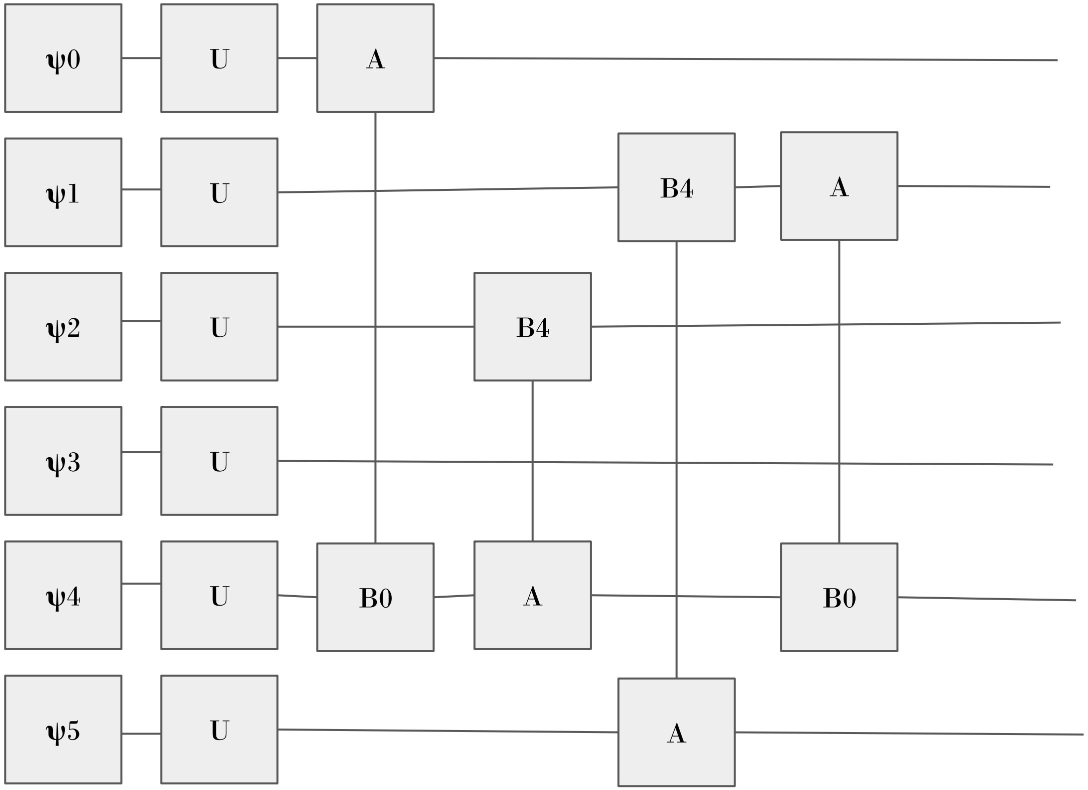
$$|\psi_A\rangle = |\psi_1\rangle U_1 U_3 \longrightarrow \boxed{A}$$

$$|\psi_B\rangle = |\psi_2\rangle CZ_{|\psi_1\rangle} U_2 CZ_{|\psi_1\rangle U_1} U_4 \longrightarrow \boxed{Bi}$$



This time ψj and ψi can be two qubits nonadjacent

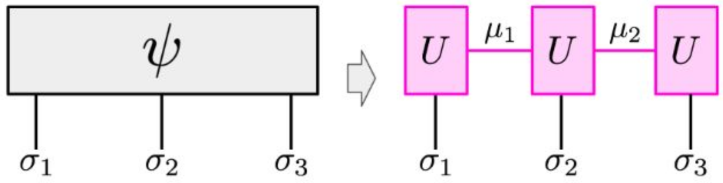
Pattern Identification : circuit 5

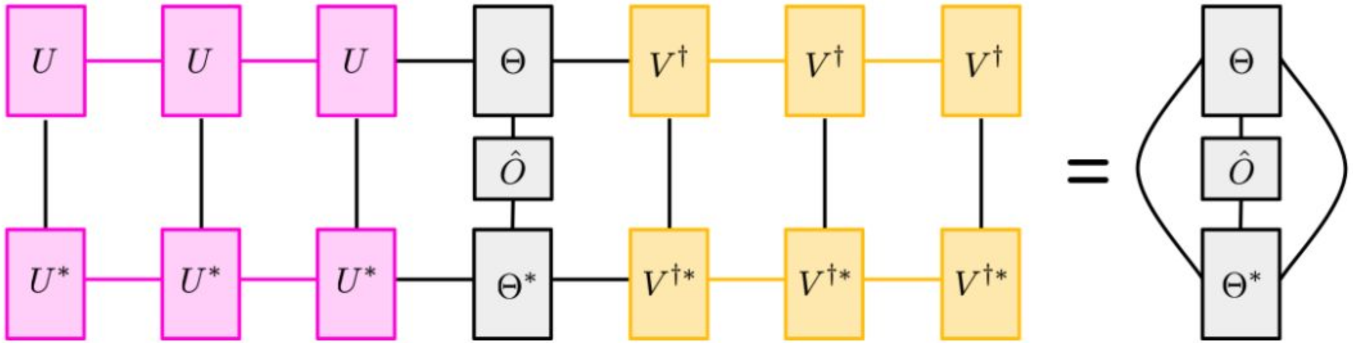


MPS / MPO Simulation

We use MPS to fasten our calculations using its canonical properties to compute our circuit more efficiently

$$|\psi\rangle = \sum_{\sigma_1,..,\sigma_n} U^{\sigma_1}..U^{\sigma_n}|\sigma_1..\sigma_n\rangle,$$





How to create a Peak

- Generate a random quantum circuit with a suitable gate pattern.
- Apply an optimization to a parameterized quantum circuit (PQC) to maximize the probability of a target output.
- Use an ansatz like RealAmplitudes or EfficientSU2 for the PQC.
- The peak quantum circuit is the concatenation of the random circuit and the optimized PQC.
- The goal is to amplify the probability of a specific bitstring.

But we were not able to identify the link between the optimization and which qubit will peak.
Therefore we couldn't master the peak quantum circuit production process.

Objective function:

$$\max_{\boldsymbol{\theta}} \left| \langle 0^n | U_{rqc} U_{pqc}(\boldsymbol{\theta}) | 0^n \rangle \right|^2.$$

State Reconstruction

```python
backend = AerSimulator(method='matrix_product_state')

from qiskit.transpiler import generate_preset_pass_manager

# Transpile the subexperiments to ISA circuits
pass_manager = generate_preset_pass_manager(optimization_level=1, backend=backend)
isa_subexperiments = {
    label: pass_manager.run(partition_subexpts)
    for label, partition_subexpts in subexperiments.items()
}
from qiskit import ClassicalRegister

# Add classical registers + measurements to each subcircuit in-place
for label, circuit_group in isa_subexperiments.items():
    for i, subcircuit in enumerate(circuit_group):
        num_qubits = subcircuit.num_qubits
        creg = ClassicalRegister(num_qubits, name="cr")
        subcircuit.add_register(creg)
        subcircuit.measure(range(num_qubits), range(num_qubits))

with Batch(backend=backend) as batch:
    sampler = SamplerV2(mode=batch)
    jobs = {
        label: sampler.run(subsystem_subexpts, shots=shots)
        for label, subsystem_subexpts in isa_subexperiments.items()
    }
results = {label: job.result() for label, job in jobs.items()}
```

```python
# Reconstructing the expectation values of our 48 qubits circuit
reconstructed_expval_terms = reconstruct_expectation_values(
    results,
    coefficients,
    subobservables,
)

# Reconstruct final expectation value
reconstructed_expval = np.dot(reconstructed_expval_terms, observable.coeffs)
print(f"Reconstructed expectation value: {reconstructed_expval}")
```


```
'00001100': 1, '00110000': 1, '01001011': 1, '01010110': 1, '01011010': 1, '01110000': 1, '10001000': 1, '10011101': 1,
'00010011': 1, '01010011': 1, '10011100': 1, '10111100': 1, '11001000': 1, '11001101': 1, '11011000': 2, '11100000': 1,
'00001110': 1, '00101111': 1, '00110111': 1, '00111010': 1, '00111100': 1, '10000011': 1, '10010001': 1, '10011011': 1,
'00010100': 2, '00111011': 1, '01000100': 1, '01010111': 1, '01101000': 1, '01111101': 1, '11000100': 1, '11100101': 1,
'00000111': 1, '00001101': 1, '00011010': 1, '00101001': 2, '00101010': 1, '01100001': 1, '01100111': 1, '10011100': 1,
'00001100': 1, '00011010': 1, '00111000': 1, '01000101': 1, '01100111': 1, '01110000': 1, '10001010': 1, '10011000': 1,
'00011110': 1, '00100110': 1, '00100111': 1, '00110100': 1, '01111001': 1, '10010101': 2, '10011000': 1, '10011110': 1,
'00000111': 1, '00011111': 1, '00110100': 1, '01000001': 2, '01001110': 1, '01010100': 1, '01101000': 1, '10000000': 1,
'00001010': 1, '00010010': 1, '00010100': 1, '00100100': 1, '01001000': 1, '01010000': 1, '10011000': 1, '10111011': 1,
'00010001': 1, '00100111': 1, '01000000': 1, '01100101': 1, '01110001': 1, '10011110': 1, '10111111': 1, '11010010': 1,
'00001110': 1, '00010101': 1, '00011101': 1, '00100101': 1, '00101111': 1, '00110101': 1, '00111011': 1, '10000010': 1,
'00100011': 1, '01010101': 1, '01101001': 1, '10000001': 1, '10100010': 1, '10100011': 1, '10100110': 1, '10110110': 1,
'00000011': 1, '00011001': 1, '01000110': 1, '01101011': 1, '01101101': 1, '01111110': 1, '10111011': 1, '10111101': 1,
'00000111': 1, '00101101': 1, '01100001': 1, '10001001': 1, '10100000': 2, '10111000': 1, '11000010': 2, '11011100': 1}
'00011010': 1, '00100011': 1, '01000001': 1, '01011111': 1, '01101001': 1, '10000000': 1, '10101001': 1, '11101100': 1,
'00100011': 1, '00110110': 1, '01000010': 1, '01000110': 1, '01001000': 1, '01001001': 1, '01101010': 1, '11001000': 1,
'00001010': 1, '00011111': 1, '00100001': 1, '00100111': 1, '00101000': 1, '01100001': 1, '01100101': 1, '10000101': 1,
'00000101': 1, '01000110': 1, '01011101': 1, '01110100': 1, '10001001': 1, '10010001': 1, '10010111': 1, '10100110': 1,
'00001101': 1, '00100101': 1, '00111011': 1, '01000000': 1, '01000101': 1, '10001100': 1, '11011100': 1, '11100000': 1,
```

Creative extensions

ML Model Approach

We thought about training a Machine Learning Model on peak quantum circuit that we would create artificially.

The objective was to train a model able to identify the peaks in these circuits.

But, as we were not able to fully understand the correlation between the optimisation and the peak during the creation of peak quantum circuits, we decided to drop this idea.

But it could be a good direction to dig deeper with a better comprehension.

Multi-basis State Tomography

Use classical shadows to reduce the cost of multi-basis state tomography

- Measure qubits randomly in different bases
- Estimate arbitrary observables
- Train another circuit to imitate its behavior

Future Directions

- Deepen the intuitive link with Grover's algorithm
- Combine peak quantum circuit with quantum optimization
- Map features for quantum classifier
- Be able to control the production of multiple peaks quantum circuit to communicate more information

|Y⟩Quantum

References

[1] Aaronson, S., & Zhang, Y. (2022). On verifiable quantum advantage with peaked circuit sampling. University of Texas at Austin; University of Toronto; Vector Institute for Artificial Intelligence. Retrieved from https://arxiv.org/abs/2112.08496

[2] Qiskit Community. (2024). Qiskit Addon Cutting Tutorials. IBM Quantum. https://qiskit.github.io/qiskit-addon-cutting/tutorials/

[3] BlueQubit Docs, https://app.bluequbit.io/docs

[4] Qiskit Docs, https://docs.quantum.ibm.com/api/qiskit/qiskit.synthesis.SuzukiTrotter

[5] IBM Quantum. (2024). Specify observables using Pauli operators. Retrieved from https://docs.quantum.ibm.com/guides/specify-observables-pauli

References

[6] Bravyi, S., Gosset, D., & Liu, Y. (2023). Classical simulation of peaked shallow quantum circuits. https://arxiv.org/abs/2309.08405

[7] Zlokapa, A., Villalonga, B., Boixo, S., & Lidar, D. A. (2023). Boundaries of quantum supremacy via random circuit sampling. npj Quantum Information, 9, 36. https://arxiv.org/abs/2005.02464

[8] PennyLane Team. (2024). Matrix Product State (MPS) simulator tutorial. Xanadu. Retrieved from https://pennylane.ai/qml/demos/tutorial_mps

[9] Zhang, Y. (2024). Peaked circuits. https://github.com/yuxuanzhang1995/Peaked-circuits