QOSF task

Dhawal Verma

October 2024

1 Noise Model

Given a quantum circuit, this piece of code add noise gates, which are Pauli Operators depending on the wire-number of gate. The probablity for a random Pauli operator acting on the qubit after a one-qubit gate is α and probability of having a random Pauli operator acting on the qubit after a two-qubit gate is β.

error gate = [XGate(), YGate(), ZGate()]

```
# Determine if noise should be added
if gate.num_qubits == 1:
    # Add noise after a one-qubit gate with probability p1
    if np.random.rand() < p1:
        pauli_gate = np.random.choice([0,1,2])
        noisy_circuit.append(error_gate[pauli_gate],[qubits[0]])

elif gate.num_qubits == 2:
    # Add noise after a two-qubit gate with probability p2
    if np.random.rand() < p2:
        # Apply a random Pauli gate to each qubit in the two-qubit gate
        for qubit in qubits:
            pauli_gate = np.random.choice([0,1,2])
            noisy_circuit.append(error_gate[pauli_gate], [qubit])
```

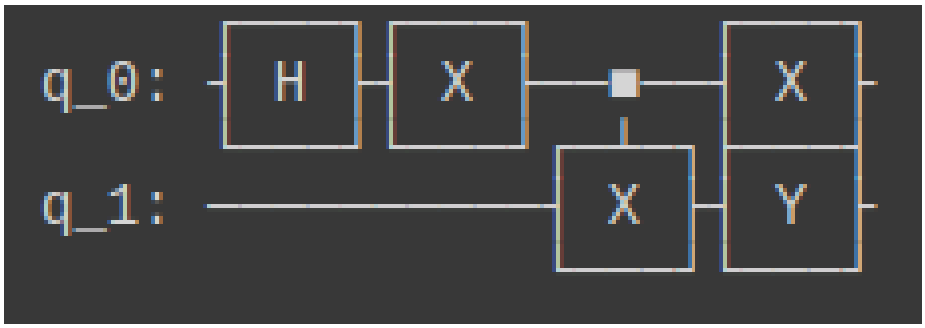
Figure 1: Enter Caption

A bell circuit composed of H and CNOT, will result in

The function add_noise takes three params as quantum circuit, alpha and beta, to return a noisy circuit. This model of noise deals with the bitflip and phaseflip, which models the noise in the channel.

2 Transpilation

Here, a circuit which can be composed of different exotic gates need to be decompose or represent in the combination of basis gates. Basis gates (or native

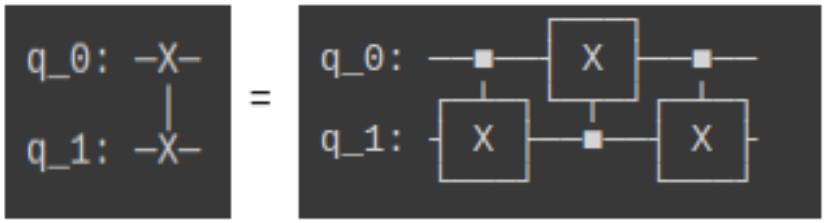


gates) are the fundamental operations that a specific quantum hardware can execute directly. Each quantum processor has a set of basis gates defined by the physical constraints of its hardware architecture. More complex quantum gates are then constructed from these basis gates using a process called transpilation or decomposition. The function custom_transpile will take a circuit and transpile the gates into following basis gates; CX,ID,RZ,SX,X

In Qiskit framework, the same basis gate set is used for the transpilation, but CX is replaced by ECR. ECR gate are equivalent of CX(CNOT) gate, with a facility to cancel out unwanted errors in IBM Quantum processor. The transpilation perform in Qskit by PassManager class keep care of the mapping of logical qubits into physical qubits, conversion of gates into basis gates and the optimization of the number of gates used.

The function custom_transpile that I created will take all the gates that is used in the quantum_sum, and transpile them into basis gates. The gates used in quantum_sum are RX,RY,CP,SWAP,H.

The transpilation works by first decomposing the gates into basis gates, and then adding the bunch of basis gates inplace in the circuit instead of the former exotic gate. For example, SWAP gate will be replaced by three CNOT gates as shown in figure.



SWAP Three CNOTs

3 Adding two numbers

The Draper adder employs a combination of the Quantum Fourier Transform (QFT) and controlled-phase operations to manipulate qubits that represent the binary forms of the integers being added. The QFT facilitates the conversion of the numbers into a superposition of states, allowing for the simultaneous processing of multiple combinations. After performing the necessary phase rotations to add the values, the algorithm applies the inverse QFT to retrieve the result in a superposition state, effectively encoding the sum in the quantum state of the qubits.

QFT circuit are used to convert the computational basis into frequency domains where operations can be performed over the phase rather than over amplitude. And the inverse QFT is used to bring back the circuit to computational basis. The actual operation that mimics the addition of b onto a is the controlled phase rotations, performed by CP gate. The number of qubits require is $2 * n$ where n is the maximum of the number of binary digits of both a and b.

On a noise-free simulator, the measurement of Draper Sum Algorithm circuit returns the count of the result highest or equal to the num_shots. In Qiskit, the simulator is obsolete, so I am presenting the result of adding two numbers a = 1 and b = 1, using the Draper Algorithm, done on Utility_Scale Quantum Computer.

The measurement are done using observables, which are expectation values for the given Pauli operators provided by SparsePauliOp. For quantum_sum(1,1), the observables were "ZIII","IZII","IIZI","IIIZ","XIII","IXII","IIXI","IIIX","ZZII"...

The job_result gave the expectation value as [1.00824742, -1.02444208, -0.49897331, -0.59521332, -0.00618557, 0.03613177, -0.10061602, -0.07284079, -1.03928171, 0.50719823, 0.56612022, -0.59185919, -0.02244669, -0.00442968, 0.01311475, 0.04840484]. To infer the meaning of these expectation values for the given Pauli operators, here's a step-by-step breakdown:

1. Matching Expectation Values with Observables: Each value in the Expectation values corresponds to the observable at the same index in observables_labels. For instance:

1.00824742 corresponds to "ZIII". -1.02444208 corresponds to "IZII", and so on.

2. Positive and Negative Signs:

Positive Values: A positive expectation value suggests alignment of the qubit(s) with the +1 eigenstate of the Pauli observable (e.g., —0⟩ for Pauli-Z). Negative Values: A negative expectation value implies alignment with the -1 eigenstate of the observable (e.g., —1⟩ for Pauli-Z). Strength of Alignment:

3. Values Near ±1 indicate strong alignment, meaning that the measured qubit state likely matches the associated eigenstate of the Pauli operator. Values Near 0 indicate less alignment or a superposition/mixed state along that axis. For example, 0.03613177 for "IXII" suggests low alignment with this observable. Interpretation by Observable Type:

4. Single-Qubit Operators ("ZIII", "IZII", "IIZI", "IIIZ"): These measure individual qubits in the Z basis. For example, the value -1.02444208 for "IZII"

implies that the second qubit tends toward the —1⟩ state when measured along Z.

5. X-Basis Operators ("XIII", "IXII", "IIXI", "IIIX"): These check alignment in the X basis, with positive or negative values indicating alignment or anti-alignment in the $|+>$ or $|->$ states, respectively. The value 0.04840484 for "XIIX" implies only slight alignment with the correlation of the first and last qubits in the X basis.

6. ZZ Correlations ("ZZII", "IZZI", "IIZZ", "ZIIZ"): These measure the alignment of pairs of qubits in the Z basis. Positive values suggest qubits are more often in the same state ($|00>$ or $|11>$), while negative values suggest anti-alignment (—01⟩ or —10⟩).

7. The value -1.03928171 for "ZZII" shows strong anti-correlation in the Z basis for the first two qubits, meaning they are often in opposite states (one is $|0>$, the other is $|1>$). XX Correlations ("XXII", "IXXI", "IIXX", "XIIX"): These measure pairwise alignment in the X basis.

The value -0.59185919 for "ZIIZ" suggests anti-alignment between the second and third qubits in the Z basis Putting It Together:

Qubit 1 likely has high alignment with the $|0>$ state in the Z basis (as seen with "ZIII" and value 1.00824742) and Qubit 2 has high alignment with the $|1>$ state. Thus the final state is measure as $|0100>$. Following the left hand rule, and considering the left digits to be least significant, this state is binary form of 2, which is 1+1. The pairwise anti-correlation values in "ZZII" suggest that Qubits 1 and 2 are often in opposite states in the Z basis. Still, we are not sure about the last two qubits whether they are in 0 or 1 state. this is because the error in the real QPUs that has been used for the calculation already takes over.

4 Effects of noise on quantum addition

Now we employs all of the function that we created in a common circuit. This includes the quantum_sum_circuit, the transpilation function, and the basic model of noise. The QPU here is already noisy, and Qiskit lack the simulator thus the true effect of noise that we model using Pauli Operator can not be studied with QPU.

The flow of the code is:

Draper_Adder → Custom Transpilation → Adding Noise → Qiskit Transpilation → Simulation result.

We will take three circuits that adds number 1 and 2, with set of *alpha* and *beta* equals to [0.0,0.0],[0.1,0.1],[0.2,0.2],[0.3,0.3]. Further in the PassManager, we will keep optimizationlevel = 3, so that we get smaller depth circuit.

As we are employing two transpilation with the same basis gate set, it does not effect the result substantially, and can be used ofr the study of the noise. The second transpilation will not change any gate except CNOT to ECR, which are equivalent in number of qubits.

1. How does the noise affect the results? In the context of algorithms like the Draper adder or QFT, noise can alter the phase and amplitude of states, leading to inaccurate results.

2. Is there a way to decrease the effect of noise?

Error Mitigation Techniques such as ZNE, which run the circuit at different noise levels and extrapolate to estimate the zero-noise result. Error Correction encodes logical qubits into multiple physical qubits to detect and correct errors. Reducing the number of gates by decomposing the circuit into a smaller set of essential operations or transpiling for optimal gate use can decrease error rates. Regular calibration of qubits and using high-fidelity gates and qubits with longer coherence times reduce the impact of noise.

3. How does the number of gates used affect the results?

Each gate operation adds a small probability of error; thus, the more gates used, the higher the cumulative error. Complex gates take longer to execute, and as gate operations accumulate, qubits may decohere before the circuit completes.