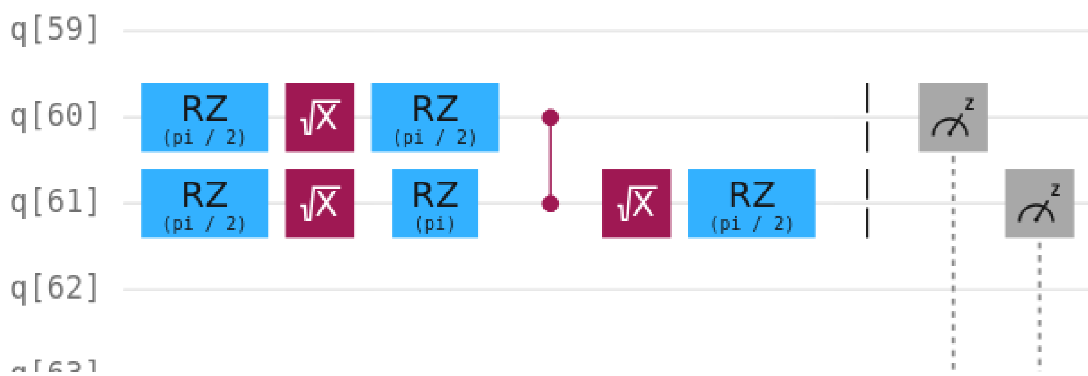

Hardware-Level Explanation of the Transpiled Bell Circuit (IBM Torino)

This document summarizes, in detailed but structured form, the meaning of the hardware-native circuit you observed after executing your Bell-state experiment on the IBM **ibm_torino** backend. This is the machine-language version of your simple logical circuit:

H(0)
CX(0, 1)
measure all

Because superconducting quantum processors do **not** support Hadamard or textbook CNOT gates natively, Qiskit must translate your logical circuit into the device's **basis gate set**, which is limited to:

- **RZ(θ)** : virtual Z rotation (no physical pulse)
- **SX** : square-root X rotation (native pulse)
- **X** : full π rotation around X
- **CX** : cross-resonance-based entangling gate

Everything else — including H, S, T, and even some Y rotations — must be synthesized from these.

Below is a clean, modular explanation of **each component** in your transpiled circuit.

---

1. Physical Qubit Mapping (q[60], q[61])

Your logical qubits were automatically mapped to **physical qubits 60 and 61**. This is chosen by Qiskit based on:

- device connectivity (must be neighbors for CX),
- gate fidelity,
- error rates,
- routing efficiency.

You don't choose these unless you *force* a mapping.

---

2. The Hadamard Decomposition (q[60])

A true Hadamard gate does not exist on IBM hardware. Instead, Qiskit synthesizes it as:

H ≡ RZ(π/2) → SX → RZ(π/2)

This is exactly what you see at the left of q[60]:

- **RZ(π/2)** : virtual phase shift
- **SX** : physical microwave pulse
- **RZ(π/2)** : another virtual phase shift

Together, these pulses implement the same unitary as the Hadamard.

---

3. Why q[61] Gets Extra Gates Before the CX

Even though you didn't apply any operation to qubit 1 before the CX, Qiskit inserts matching **RZ–SX–RZ** blocks on q[61]. These serve multiple hardware-level purposes:

A. Frame alignment

Superconducting qubits use virtual-Z frame tracking. The transpiler adds RZ gates to synchronize qubit phases prior to entanglement.

B. Compensation for parasitic interactions

The cross-resonance (CX) pulse unintentionally introduces stray interaction terms (IX, ZI, ZZ, XI). Pre-corrections reduce these.

C. Calibration optimization side effects

Machines maintain calibrated frames for individual qubits; these pre-gates bring q[61] into the same dynamical frame as q[60] before entangling.

So these gates appear not because your algorithm requires them, but because **the hardware does**.

---

4. The Entangling Gate (CX) Implementation

Your logical CNOT becomes a machine-level **cross-resonance pulse sequence**, represented visually as:

```
q[60] •────────
q[61] └──X─────
```

But physically, a CNOT is *not* a single clean pulse. It is:

- a microwave drive applied to q[60] at q[61]'s resonant frequency,
- producing an effective Hamiltonian ~ ZX,
- with multiple cancellation and echo pulses.

This is why there are **additional SX and RZ gates on q[61] after the CNOT**: these are part of cross-resonance echo sequences required to clean up undesired evolution.

---

5. Barrier Separators (the “||” vertical markers)

The vertical bars mark **compiler barriers**. They signal the transpiler:

- not to reorder gates across this point,
- to finalize all unitary evolution before measurement,
- to keep measurement operations isolated.

These do not affect the physics but structure the schedule.

---

6. Measurement Pulses (Gray Z boxes)

The gray boxes represent the final **Z-basis measurement pulses**, which:

- energize the qubit's readout resonator,
- demodulate the returning microwave signal,
- collapse the qubit into |0⟩ or |1⟩,
- feed the result into classical registers.

The dashed lines connect these measurements to the output classical bits.

---

7. Putting It All Together

Your simple logical Bell circuit:

H(0)
CX(0, 1)
measure

Becomes this hardware-expanded form:

q[60]: RZ → SX → RZ ————●—————————— Measure
q[61]: RZ → SX → RZ ————X → SX → RZ —— Measure

Where each additional pulse corrects, aligns, or compensates for physical realities of superconducting qubit operation.

8. Why All This Matters

This circuit reveals the entire **physical implementation layer** of your algorithm. It shows:

- how abstract gates become microwave pulses,
- how qubit connectivity forces routing choices,
- how hardware compensates for unwanted couplings,
- how real machines differ profoundly from textbook circuits.

Studying these decompositions is essential for:

- understanding quantum noise,
- interpreting experimental results,
- improving algorithm design for real hardware,
- working effectively with quantum transpilation.

This summary serves as a reference for your future quantum computing explorations.