In [84]:
```
import matplotlib.pyplot as plt
import numpy as np
import rustworkx as rx
import statistics

from qiskit_ibm_runtime import QiskitRuntimeService
```

In [85]:
```
service = QiskitRuntimeService()
backend = service.backend("ibm_sherbrooke") # Eagle r3
```

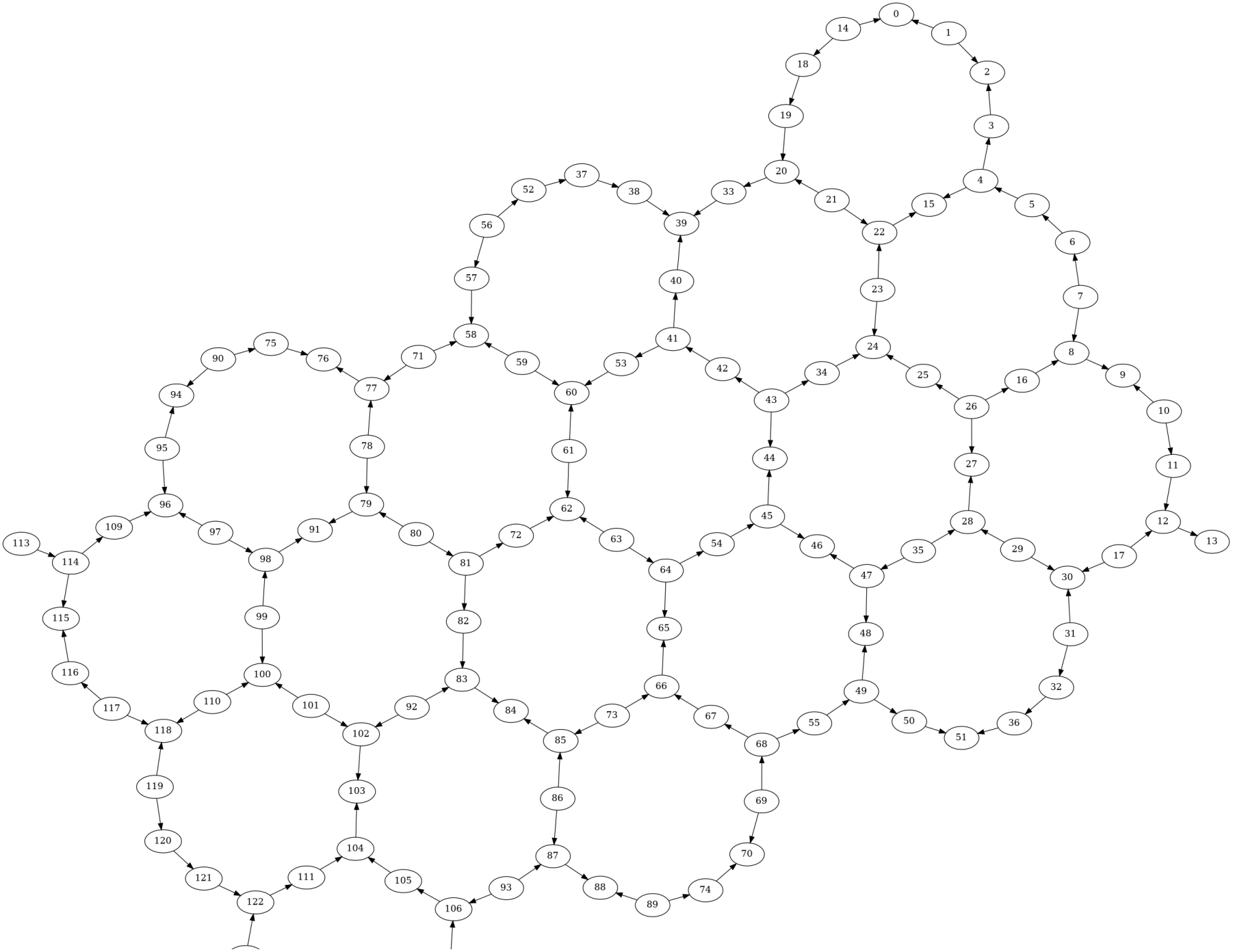
In [86]:
```
print(
    f"""
{backend.name}, {backend.num_qubits} qubits
processor type = {backend.processor_type}
basis gates = {backend.basis_gates}
""")
```

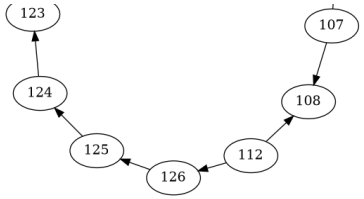
```
ibm_sherbrooke, 127 qubits
processor type = {'family': 'Eagle', 'revision': 3}
basis gates = ['ecr', 'id', 'rz', 'sx', 'x']
```

In [87]:

```
# This function requires that Graphviz is installed. If you need to install Graphviz you can refer to:
# https://graphviz.org/download/#executable-packages for instructions.
try:
    fig = backend.coupling_map.draw()
except RuntimeError as ex:
    print(ex)
fig
```

Out[87]:


0
14
1
18
2
19
3
20
4
37
52
38
33
21
15
5
56
39
22
6
57
40
23
7
58
41
24
8
90
75
76
71
59
53
42
34
25
16
9
94
77
60
43
26
10
95
78
61
44
27
11
96
79
62
45
28
12
109
97
91
80
72
63
54
46
35
29
17
113
114
98
81
64
47
30
115
99
82
65
48
31
116
100
83
66
49
32
117
110
101
92
84
73
67
55
50
36
118
102
85
68
51
119
103
86
69
120
104
87
70
121
111
105
93
88
74
122
106
89



In [88]:
```
for qn in range(backend.num_qubits):
    if (qn>=10):
        break
    print(f"{qn}: {backend.qubit_properties(qn)}")
```

```
0: QubitProperties(t1=0.0004293986238562639, t2=0.0003463274351571056, frequency=4635659360.509004)
1: QubitProperties(t1=0.0003063573838660812, t2=0.0004217947797350284 6, frequency=4736282041.074064)
2: QubitProperties(t1=0.0002603211878308971, t2=0.0002004966612998681 5, frequency=4819175412.239935)
3: QubitProperties(t1=0.0003588053042150239, t2=0.0002390807017304374, frequency=4747175699.898465)
4: QubitProperties(t1=0.0004031707607927116 6, t2=0.0004815865074737176 6, frequency=4787865308.857803)
5: QubitProperties(t1=0.0001138813306713741 2, t2=0.0003083799337469611 6, frequency=4850836738.234322)
6: QubitProperties(t1=0.0002374034965179541 8, t2=0.0001297866153741689 2, frequency=4899520617.220607)
7: QubitProperties(t1=0.0002777157460784294 4, t2=6.261095306795752e-05, frequency=4755794851.187578)
8: QubitProperties(t1=0.0003347068661430919 7, t2=0.0003316442175998825, frequency=4812593695.403755)
9: QubitProperties(t1=0.0002918465521830536 3, t2=6.439098542367134e-05, frequency=4637774824.0964)
```

In [89]:
```
t1s = [backend.qubit_properties(qq).t1 for qq in range(backend.num_qubits)]
f"Median T1: {(statistics.median(t1s)*10**6):.2f} \u03bcs"
```

Out[89]: 'Median T1: 282.51 µs'

In [90]:
```
# Your code here

t2s = [backend.qubit_properties(qq).t2 for qq in range(backend.num_qubits)]
f"Median T2: {(statistics.median(t2s)*10**6):.2f} \u03bcs"
```

Out[90]: 'Median T2: 219.75 µs'

In [91]:
```
target = backend.target
target.keys()
```

Out[91]:
```
dict_keys(['delay', 'rz', 'reset', 'switch_case', 'ecr', 'measure', 'x', 'if_else', 'id', 'for_loop', 's
x'])
```

In [92]:
```
for i, qq in enumerate(target['sx']):
    if (i>=10):
        break
    print(i, qq, target['sx'][qq])
```

```
0 (0,) InstructionProperties(duration=5.6888888888888887e-08, error=0.0005015628828134774)
1 (1,) InstructionProperties(duration=5.6888888888888887e-08, error=0.0005840615095545203)
2 (2,) InstructionProperties(duration=5.6888888888888887e-08, error=0.00017871478265339173)
3 (3,) InstructionProperties(duration=5.6888888888888887e-08, error=0.00017892290552088907)
4 (4,) InstructionProperties(duration=5.6888888888888887e-08, error=0.0001140242622522436)
5 (5,) InstructionProperties(duration=5.6888888888888887e-08, error=1.0)
6 (6,) InstructionProperties(duration=5.6888888888888887e-08, error=0.0003294099104723003)
7 (7,) InstructionProperties(duration=5.6888888888888887e-08, error=0.0004612054773508375 3)
8 (8,) InstructionProperties(duration=5.6888888888888887e-08, error=0.00012628326328157464)
9 (9,) InstructionProperties(duration=5.6888888888888887e-08, error=0.0011160292701674114)
```

In [70]:
```
for i, edge in enumerate(target['ecr']):
    if (i>=10):
        break
    print(i, edge, target['ecr'][edge])
```

```
0 (1, 0) InstructionProperties(duration=5.333333333333332e-07, error=0.019321026822751497)
1 (1, 2) InstructionProperties(duration=5.333333333333332e-07, error=0.0038381339160909467)
2 (3, 2) InstructionProperties(duration=5.333333333333332e-07, error=0.0044194877617828865)
3 (4, 3) InstructionProperties(duration=5.333333333333332e-07, error=0.0035250499862634066)
4 (4, 15) InstructionProperties(duration=5.333333333333332e-07, error=0.006129515156052617)
5 (5, 4) InstructionProperties(duration=5.333333333333332e-07, error=0.007540929673926111)
6 (6, 5) InstructionProperties(duration=5.333333333333332e-07, error=0.006480976536535871)
7 (7, 6) InstructionProperties(duration=5.333333333333332e-07, error=0.006502488783159177)
8 (7, 8) InstructionProperties(duration=5.333333333333332e-07, error=0.01071128576218977)
9 (8, 9) InstructionProperties(duration=5.333333333333332e-07, error=0.012091431861526447)
```

In [93]:
```
for i, qq in enumerate(target['measure']):
    if (i>=10):
        break
    print(i, qq, target['measure'][qq])
```

```
0 (0,) InstructionProperties(duration=1.216e-06, error=0.02294921875)
1 (1,) InstructionProperties(duration=1.216e-06, error=0.097412109375)
2 (2,) InstructionProperties(duration=1.216e-06, error=0.145263671875)
3 (3,) InstructionProperties(duration=1.216e-06, error=0.03369140625)
4 (4,) InstructionProperties(duration=1.216e-06, error=0.078857421875)
5 (5,) InstructionProperties(duration=1.216e-06, error=0.171630859375)
6 (6,) InstructionProperties(duration=1.216e-06, error=0.119873046875)
7 (7,) InstructionProperties(duration=1.216e-06, error=0.11962890625)
8 (8,) InstructionProperties(duration=1.216e-06, error=0.0146484375)
9 (9,) InstructionProperties(duration=1.216e-06, error=0.0849609375)
```

In [94]:
```
sx_errors = [inst_prop.error for inst_prop in target['sx'].values()]
f"Median SX error: {(statistics.median(sx_errors)):.3e}"
```

Out[94]: 'Median SX error: 2.015e-04'

In [73]:
```
ecr_errors = [inst_prop.error for inst_prop in target['ecr'].values()]
f"Median ECR error: {(statistics.median(ecr_errors)):.3e}"
```

Out[73]: 'Median ECR error: 5.684e-03'

In [95]:
```
# Your code here
meas_errors = [inst_prop.error for inst_prop in target['measure'].values()]
f"Median readout error: {(statistics.median(meas_errors)):.3e}"
```

Out[95]: 'Median readout error: 2.173e-02'

In [75]:
```
 1 print(backend.target["sx"][(0, )].calibration)
 2 backend.target["sx"][(0, )].calibration.draw()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[75], line 1
----> 1 print(backend.target["sx"][(0, )].calibration)
      2 backend.target["sx"][(0, )].calibration.draw()

AttributeError: 'InstructionProperties' object has no attribute 'calibration'
```

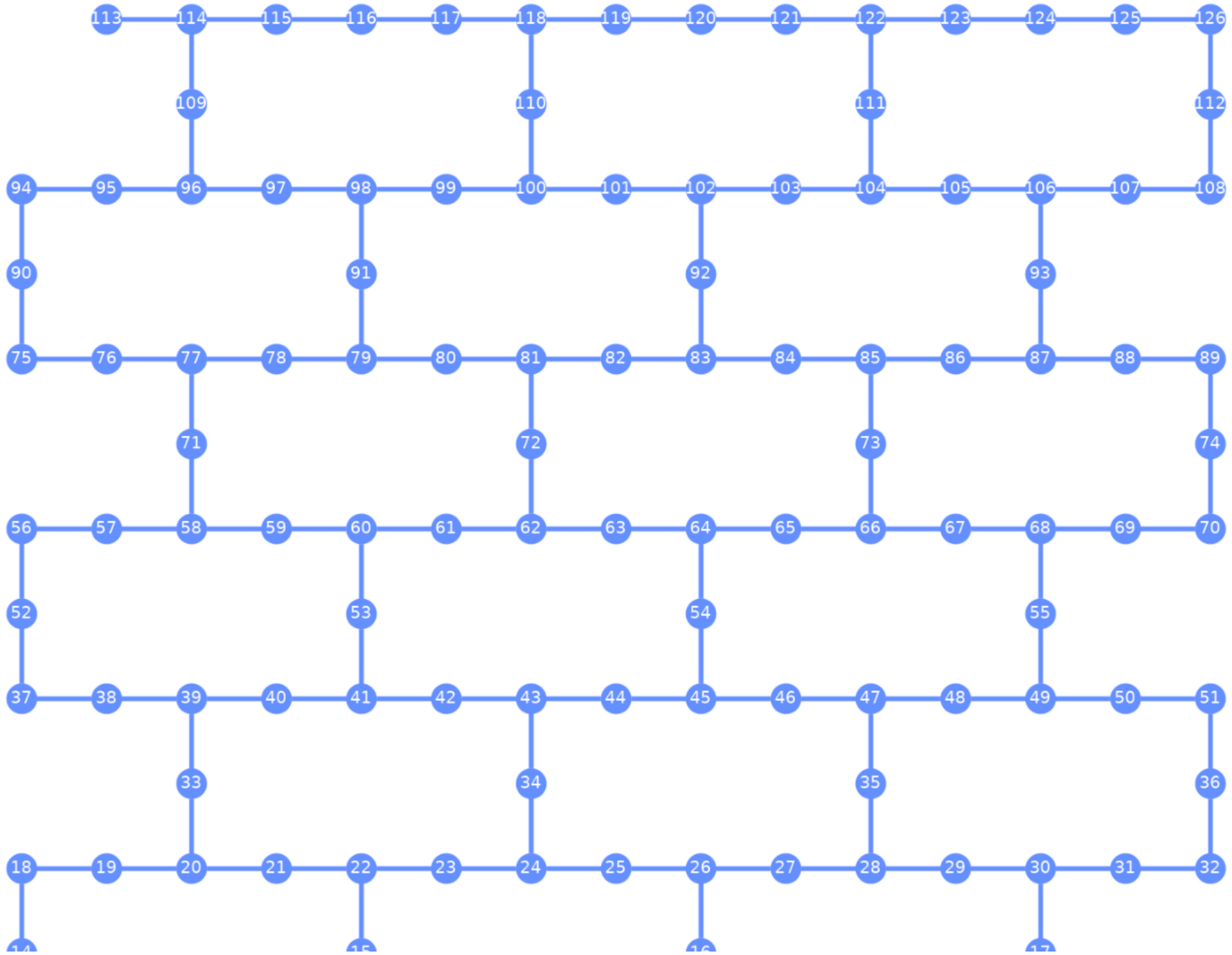
In [76]:
```
 1 # Your code here
 2
 3
 4
 5
 6
 7 # solution
 8 print(backend.target["ecr"][(122, 111)].calibration)
 9 backend.target["ecr"][(122, 111)].calibration.draw()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[76], line 8
      1 # Your code here
      2
      3
   (...)
      6
      7 # solution
----> 8 print(backend.target["ecr"][(122, 111)].calibration)
      9 backend.target["ecr"][(122, 111)].calibration.draw()

AttributeError: 'InstructionProperties' object has no attribute 'calibration'
```

In [96]:

```
from qiskit.visualization import plot_gate_map

plot_gate_map(backend, font_size=14)
```

Out[96]:

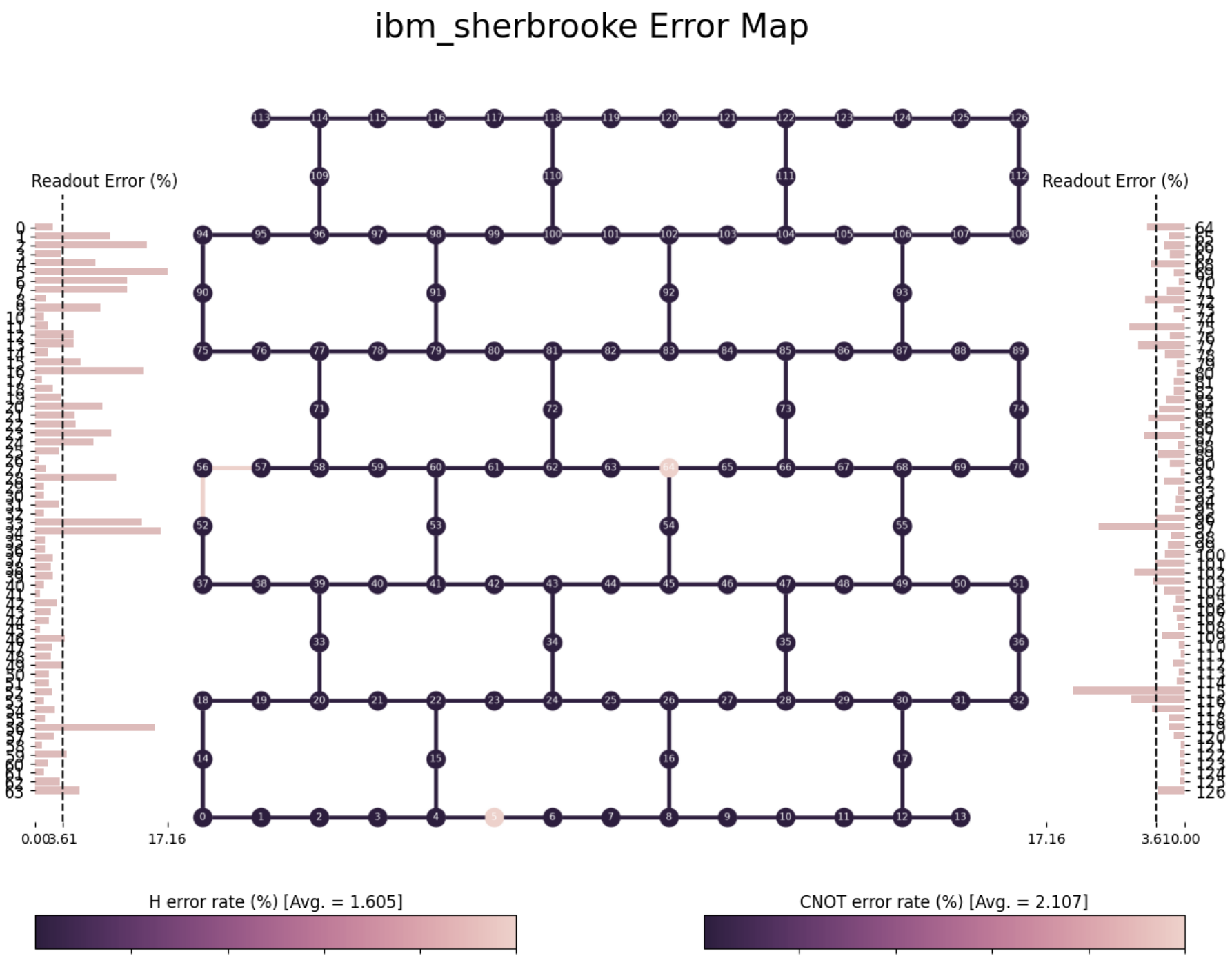

113 114 115 116 117 118 119 120 121 122 123 124 125 126
109 110 111 112
94 95 96 97 98 99 100 101 102 103 104 105 106 107 108
90 91 92 93
75 76 77 78 79 80 81 82 83 84 85 86 87 88 89
71 72 73 74
56 57 58 59 60 61 62 63 64 65 66 67 68 69 70
52 53 54 55
37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
33 34 35 36
18 19 20 21 22 23 24 25 26 27 28 29 30 31 32


14
15
16
17
0
1
2
3
4
5
6
7
8
9
10
11
12
13

In [97]:

```
from qiskit.visualization import plot_error_map

plot_error_map(backend)
```

Out[97]:


ibm_sherbrooke Error Map
Readout Error (%)
Readout Error (%)
0.00
3.61
17.16
17.16
3.61
10.00
H error rate (%) [Avg. = 1.605]
CNOT error rate (%) [Avg. = 2.107]

In [112]:
```
from qiskit.transpiler.preset_passmanagers import generate_preset_pass_manager
from qiskit_ibm_runtime import QiskitRuntimeService, SamplerV2
pm = generate_preset_pass_manager(backend=backend, optimization_level=3)
circuit_isa = pm.run(circuit_list)
```

In [117]:
```
from qiskit import QuantumCircuit
from qiskit_ibm_runtime import QiskitRuntimeService, SamplerV2 as Sampler

# Create empty circuit
circuit_isa = QuantumCircuit(2)
circuit_isa.measure_all()

# You'll need to specify the credentials when initializing QiskitRuntimeService, if they were not previ
# service = QiskitRuntimeService()
# backend = service.least_busy(operational=True, simulator=False)

sampler = Sampler(backend)
job = sampler.run([circuit_isa])
print(f"job id: {job.job_id()}")
result = job.result()
print(result)
```

```
job id: d0ljiuik4jhc73aes680
```

In [ ]:
```

```