

```

1  # -*- coding: utf-8 -*-
2  # File name: problem2.py
3  import math
4  from projectq import MainEngine
5  from projectq.ops import X, Y, Z, H, S, T, CX, CZ, Rx, Ry, Rz, Measure, All
6  from projectq.meta import Loop, Compute, Uncompute, Control
7  from projectq.engines import (MainEngine,
8                                AutoReplacer,
9                                LocalOptimizer,
10                               TagRemover,
11                               InstructionFilter,
12                               DecompositionRuleSet)
13  import projectq.setups.decompositions
14  from hiq.projectq.backends import SimulatorMPI
15  from hiq.projectq.engines import GreedyScheduler, HiQMainEngine
16
17  from mpi4py import MPI
18
19  def circuit_generator(eng, target_state, mapper):
20      """The function you need to modify.
21      Args:
22          eng:
23              The engine type should be defined in "__main__" function.
24          target_state(list):
25              A target quantum state vector like: [0.5. 0.5. 0.5. 0.5]. And this
26              simple example can be prepared from 'H | qubit[0]; H | qubit[1]'
27              where the 'qubit' are in a |0> state at very first.
28          mapper(string):
29              The mapper of a real quantum chip like: '3\n1,2\n2,3'. This simple
30              example of mapper means that there are 3 qubits in this given chip
31              and you can play 'CNOT' on 'qubit1' and 'qubit2'. You can also play
32              'CNOT' on 'qubit2' and 'qubit3'. But you can not play 'CNOT' on
33              'qubit1' and 'qubit3' because '1,3' is not in the given mapper.
34      Returns:
35          simulated_circuit(string):
36              After your calculation you may get a final quantum circuit like:
37              'H | qubit[0]; H | qubit[1]'. Each step seperated by a '; '. In
38              our score we will search operations and qubit index in your result
39              and run it on our backend. We will compare the final state your
40              circuit produces with our target state by ourselves.
41      """
42      simulated_circuit = 'H | qubit[0]; H | qubit[1]'
43      return simulated_circuit
44
45  if __name__ == "__main__":
46
47      # use projectq simulator
48      #eng = MainEngine()
49
50      # use hiq simulator

```

```
51 backend = SimulatorMPI(gate_fusion=True)
52
53 cache_depth = 10
54 rule_set = DecompositionRuleSet(modules=[projectq.setups.decompositions])
55 engines = [TagRemover()
56            , LocalOptimizer(cache_depth)
57            , AutoReplacer(rule_set)
58            , TagRemover()
59            , LocalOptimizer(cache_depth)
60            , GreedyScheduler()
61            ]
62
63 # make the compiler and run the circuit on the simulator backend
64 eng = HiQMainEngine(backend, engines)
65
66 qureg = eng.allocate_qureg(5)
67
68 target_state = [0.5, 0.5, 0.5, 0.5]
69
70 mapper = '3\n1,2\n2,3'
71
72 circuit = circuit_generator(eng, target_state, mapper)
73
74 All(Measure) | qureg
75
76 print(circuit)
```