```python
# -*- coding: utf-8 -*-
# File name: problem2.py
import math
from projectq import MainEngine
from projectq.ops import X, Y, Z, H,  S, T, CX, CZ, Rx, Ry, Rz, Measure, All
from projectq.meta import Loop, Compute, Uncompute, Control
from projectq.cengines import (MainEngine,
                               AutoReplacer,
                               LocalOptimizer,
                               TagRemover,
                               InstructionFilter,
                               DecompositionRuleSet)
import projectq.setups.decompositions
from hiq.projectq.backends import SimulatorMPI
from hiq.projectq.cengines import GreedyScheduler, HiQMainEngine

from mpi4py import MPI

def circuit_generator(eng, target_state, mapper):
    """The function you need to modify.
    Args:
        eng:
            The engine type should be defined in "__main__" function.
        target_state(list):
            A target quantum state vector like: [0.5. 0.5. 0.5. 0.5]. And this
            simple example can be prepared from 'H | qubit[0]; H | qubit[1]'
            where the 'qubit' are in a |0> state at very first.
        mapper(string):
            The mapper of a real quantum chip like: '3\n1,2\n2,3'. This simple
            example of mapper means that there are 3 qubits in this given chip
            and you can play 'CNOT' on 'qubit1' and 'qubit2'. You can also play
            'CNOT' on 'qubit2' and 'qubit3'. But you can not play 'CNOT' on
            'qubit1' and 'qubit3' because '1,3' is not in the given mapper.
    Returns:
        simulated_circuit(string):
            After your calculation you may get a final quantum circuit like:
            'H | qubit[0]; H | qubit[1]'. Each step seperated by a '; '. In
            our score we will search operations and qubit index in your result
            and run it on our backend. We will compare the final state your
            circuit produces with our target state by ourselves.
    """
    simulated_circuit = 'H | qubit[0]; H | qubit[1]'
    return simulated_circuit

if __name__ == "__main__":

    # use projectq simulator
    #eng = MainEngine()

    # use hiq simulator
    backend = SimulatorMPI(gate_fusion=True)

    cache_depth = 10
    rule_set = DecompositionRuleSet(modules=[projectq.setups.decompositions])
    engines = [TagRemover()
               , LocalOptimizer(cache_depth)
               , AutoReplacer(rule_set)
               , TagRemover()
               , LocalOptimizer(cache_depth)
               , GreedyScheduler()
               ]

    # make the compiler and run the circuit on the simulator backend
    eng = HiQMainEngine(backend, engines)

    qureg = eng.allocate_qureg(5)

    target_state = [0.5, 0.5, 0.5, 0.5]

    mapper = '3\n1,2\n2,3'

    circuit = circuit_generator(eng, target_state, mapper)

    All(Measure) | qureg

    print(circuit)
```