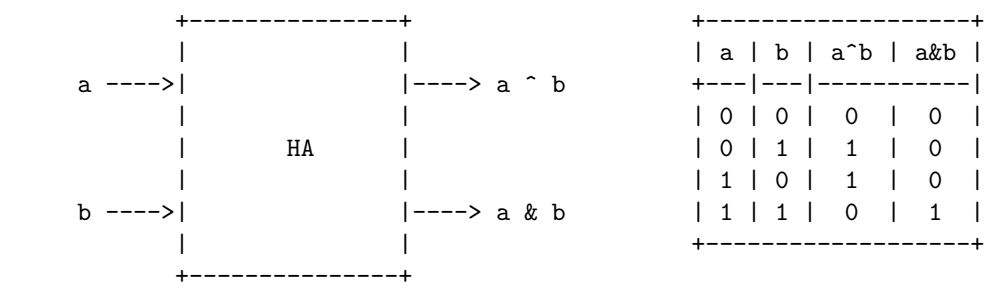
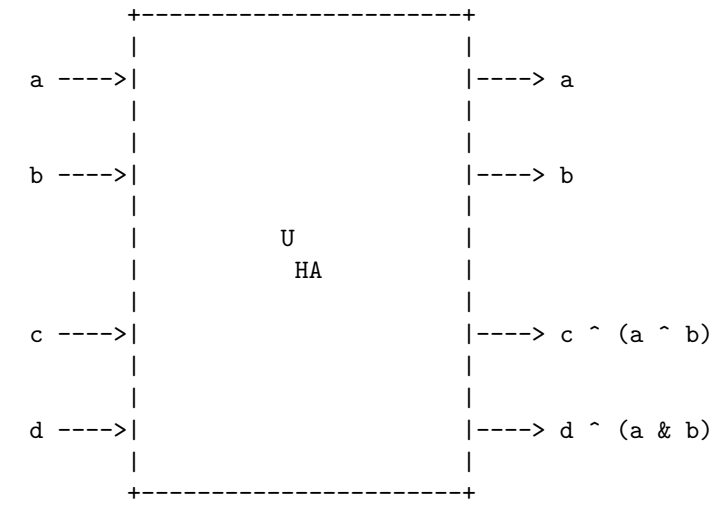
1 Reversible Half Adder

Half-Adder:



a	b	a^b	a&b
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

reversible:



a	b	c	d	a	b	c^(a^b)	d^(a&b)
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	1
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
1	0	0	0	1	0	1	0
1	0	0	1	1	0	1	1
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	1
1	1	0	0	1	1	0	1
1	1	0	1	1	1	0	0
1	1	1	0	1	1	1	1
1	1	1	1	1	1	1	0

one-to-one

```
quantum-circuit:
----------------

  a -----#--------------------------------#------
         |                                |
         |                                |
  b ------------------#------------------#------
         |           |                  |
         |           |                  |
  c ----|+|----------|+|-----------------------
           c ^ a        (c ^ a)^ b       |
                                         |
  d --------------------------------------|+|-----
                                            d ^ (a & b)
```

2 Hamming Code

https://github.com/bchatterjee99/quantum/blob/master/hamming.c

3 qtrng

https://github.com/bchatterjee99/quantum/blob/master/qtrng.py

4 Entanglement

```
       +---------+
       |         |
|0> --|    H    |-------------------------#------
       |         |   |0>+|1>              |
       +---------+  ---------             |
                     sqrt(2)               |   |01> + |10>
                                           |   -----------
                                           |     sqrt(2)
       +---------+                         |
       |         |                         |
|0> --|    X    |-------------------------|+|-----
       |         |     |1>
       +---------+               |
                                 |
                                 v

                           |01> + |11>
                           -----------
                             sqrt(2)
```

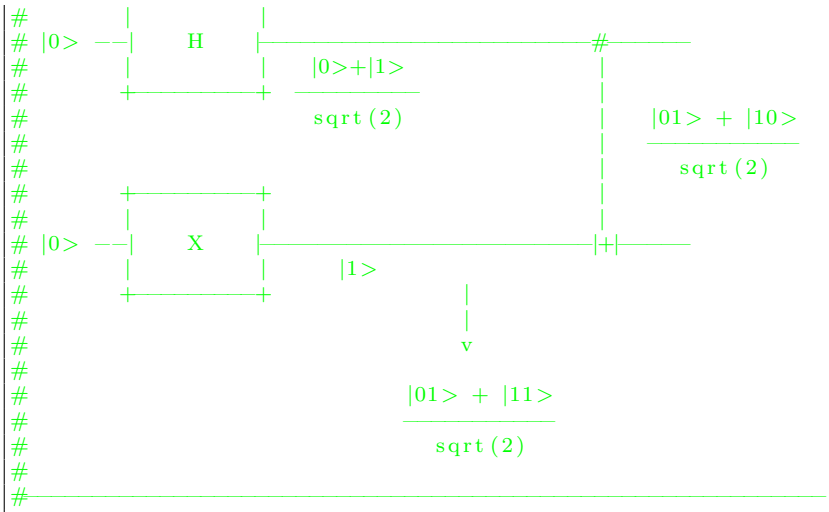
https://github.com/bchatterjee99/quantum/blob/master/entangled.py

```
from qiskit import QuantumRegister, ClassicalRegister
from qiskit import QuantumCircuit, Aer, execute

q1 = QuantumRegister(2);
c1 = ClassicalRegister(2);

qc = QuantumCircuit(q1, c1);
backend = Aer.get_backend("qasm_simulator");


#-------------------------------------------------------
#
#       +---------+
```



```
qc.h(q1[0]);
qc.x(q1[1]);
qc.cx(q1[0], q1[1])
qc.measure(q1, c1);

qjob = execute(qc, backend, shots=100);
counts = qjob.result().get_counts();

print(counts)
```