# 组合门性质与验证

## 矩阵乘可交换性

由题目猜测可得：



首先可知，若：



则这两矩阵乘可交换，于是编写如下matlab代码，验证三个Z阵，两两乘可交换：

%% 验证合并验证算子的乘可交换性

Z=[1 0;0 -1];

Z0=kron(kron(eye(2),eye(2)),Z);

Z1=kron(kron(eye(2),Z),eye(2));

Z2=kron(kron(Z,eye(2)),eye(2));

Z0Z1=Z0\*Z1;

Z1Z2=Z1\*Z2;

Z0Z2=Z0\*Z2;

Z0Z1 == Z1\*Z0

ans = 8×8 logical array

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

Z0Z2 == Z2\*Z0

ans = 8×8 logical array

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

Z1Z2 == Z2\*Z1

ans = 8×8 logical array

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

于是得证它们的乘可交换性，并猜测也具有乘可交换性。

## double-qubits单元线路的位可交换性

在提示中，我们有如下线路：

|  |
| --- |
|  |

我们猜想，如下线路的等效算子和上面的相同：

|  |
| --- |
|  |

于是编写matlab代码验证可得：

%% 验证2qubits单元线路的位可交换性

Z0\_2=kron(eye(2),Z);

Z1\_2=kron(Z,eye(2));

CNOT1=[eye(2) zeros(2);zeros(2) flip(eye(2),2)];% q1为控制位

CNOT0=[1 0 0 0;

0 0 0 1;

0 0 1 0;

0 1 0 0];% q0为控制位

unit=CNOT0\*expm(-1j\*pi/8\*Z1\_2)\*CNOT0;

unit==CNOT1\*expm(-1j\*pi/8\*Z0\_2)\*CNOT1

ans = 4×4 logical array

1 1 1 1

1 1 1 1

1 1 1 1

1 1 1 1

于是可得这两矩阵相等，也就是我们可以有两种单元线路，用它们来构建算子。

## 单元线路之间的可交换性

现在我们已知



不禁猜测，是否有



也就是单元线路之间是否是可交换次序的，于是我们依然使用matlab编程，可得：

e0=expm(-1j\*pi/8\*Z0Z1);

e1=expm(-1j\*pi/8\*Z1Z2);

e2=expm(-1j\*pi/8\*Z0Z2);

e0e1=e0\*e1;

e0e2=e0\*e2;

e1e2=e1\*e2;

e0e1==e1\*e0

ans = 8×8 logical array

1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1

e0e2==e2\*e0

ans = 8×8 logical array

1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1

e1e2==e1\*e2

ans = 8×8 logical array

1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1

也就是说，单位线路的添加顺序也可交换。

由于我们已知



如果我们假设有一个式子为：



那么我们不禁猜测是否有办法处理中间间隔的门，一种想法是交换之，而另一种想法则是约并之，我们不妨来看看三个qubit情形下的这样两个门：



其中方括号内的表示在二维希尔伯特空间下的经典受控非门，现在我们对它作积：



线路图为：

|  |
| --- |
|  |

如果约化，显然它的结果会是一个三维算子，在matlab里尝试这件事情可得

kron(CNOT0,eye(2))\*kron(eye(2),CNOT0)

ans = 8×8

1 0 0 0 0 0 0 0  
 0 0 0 1 0 0 0 0  
 0 0 0 0 0 0 1 0  
 0 0 0 0 0 1 0 0  
 0 0 0 0 1 0 0 0  
 0 0 0 0 0 0 0 1  
 0 0 1 0 0 0 0 0  
 0 1 0 0 0 0 0 0

这显然不利于我们解决问题，于是考虑第一种思路，即交换性，这种思路下我们想把旋转门换到一起，我们可以让第三位量子比特与第零位经过同样的制备，这样可以实现间邻的受控非门，下面是尝试过程：

CNOT12=kron(CNOT0,eye(2));

CNOT01=kron(eye(2),CNOT0);

CNOT21=kron(CNOT1,eye(2));

CNOT10=kron(eye(2),CNOT1);

RZ1=expm(-1j\*pi/8\*Z1);

RZ2=expm(-1j\*pi/8\*Z2);

Z3\_3=kron(Z,kron(eye(2),kron(eye(2),eye(2))));

CNOT12\_3=kron(eye(2),CNOT12);

CNOT01\_3=kron(eye(2),CNOT01);

CNOT23\_3=kron(CNOT0,kron(eye(2),eye(2)));

RZ1\_3=kron(eye(2),RZ1);

RZ2\_3=kron(eye(2),RZ2);

RZ3\_3=expm(-1j\*pi/8\*Z3\_3);

RCCRCCRC=RZ1\_3\*CNOT01\_3\*CNOT12\_3\*RZ2\_3\*CNOT12\_3\*CNOT23\_3\*RZ3\_3\*CNOT23\_3;

prodR=RZ1\_3\*RZ2\_3\*RZ3\_3;

K=inv(prodR)\*RCCRCCRC

K = 16×16 complex

Rows 7:16 | Columns 6:15

0.0000 + 0.0000i 1.0000 - 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.7071 + 0.7071i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 1.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.7071 + 0.7071i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.7071 + 0.7071i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 1.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.7071 - 0.7071i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i -0.0000 - 1.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i -0.0000 - 1.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.7071 - 0.7071i 0.0000 + 0.0000i 0.0000 + 0.0000i

RC1=RZ1\*CNOT01

RC1 = 8×8 complex

0.9239 - 0.3827i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.9239 - 0.3827i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.9239 + 0.3827i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.9239 + 0.3827i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.9239 - 0.3827i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.9239 - 0.3827i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.9239 + 0.3827i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.9239 + 0.3827i 0.0000 + 0.0000i 0.0000 + 0.0000i

RC1==CNOT01\*RZ1

ans = 8×8 logical array

1 1 1 1 1 1 1 1  
 1 1 1 0 1 1 1 1  
 1 1 1 1 1 1 1 1  
 1 0 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 0  
 1 1 1 1 1 1 1 1  
 1 1 1 1 1 0 1 1

RC1\*inv(RZ1)

ans = 8×8 complex

1.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.7071 - 0.7071i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 1.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.7071 + 0.7071i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 1.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.7071 - 0.7071i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 1.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.7071 + 0.7071i 0.0000 + 0.0000i 0.0000 + 0.0000i

RC1\*inv(RZ1)\*inv(CNOT01)

ans = 8×8 complex

1.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.7071 - 0.7071i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 1.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.7071 + 0.7071i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 1.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.7071 - 0.7071i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 1.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.7071 + 0.7071i

RC1\*CNOT12\*inv(RZ1)\*inv(CNOT01)

ans = 8×8 complex

1.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.7071 - 0.7071i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 1.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.7071 + 0.7071i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 1.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.7071 - 0.7071i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 1.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.7071 + 0.7071i

CR2=CNOT12\*RZ2

CR2 = 8×8 complex

0.9239 - 0.3827i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.9239 - 0.3827i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.9239 + 0.3827i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.9239 + 0.3827i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.9239 + 0.3827i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.9239 + 0.3827i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.9239 - 0.3827i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.9239 - 0.3827i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i

RC2=RZ2\*CNOT12

RC2 = 8×8 complex

0.9239 - 0.3827i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.9239 - 0.3827i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.9239 - 0.3827i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.9239 - 0.3827i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.9239 + 0.3827i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.9239 + 0.3827i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.9239 + 0.3827i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.9239 + 0.3827i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i

RC1\*inv(RZ1) == RC2\*inv(RZ2)

ans = 8×8 logical array

1 1 1 1 1 1 1 1  
 1 0 1 0 1 1 1 1  
 1 1 0 1 1 1 0 1  
 1 0 1 1 1 1 1 0  
 1 1 1 1 1 1 1 1  
 1 1 1 1 1 0 1 0  
 1 1 0 1 1 1 0 1  
 1 1 1 0 1 0 1 1

RC1\*CNOT12\*inv(RC1)

ans = 8×8

1 0 0 0 0 0 0 0  
 0 0 0 0 0 1 0 0  
 0 0 0 0 0 0 1 0  
 0 0 0 1 0 0 0 0  
 0 0 0 0 1 0 0 0  
 0 1 0 0 0 0 0 0  
 0 0 1 0 0 0 0 0  
 0 0 0 0 0 0 0 1

其中，



这里是因为，所以省略求逆符号不写，是一个很有趣的矩阵，它的元素组成是：



它总在对称位上有元素，并且有如下关系：

|  |  |
| --- | --- |
|  |  |
| 2 | 1 |
| 6 | e+,e+,1 |
| 10 | e+ |
| 14 | 1,1,e+ |
| 18 | 1 |
| 22 | e+,1,e+ |
| 26 | e- |
| 30 | -i,-i,e- |

即，行列和为奇数次二倍奇数位上总有1个元素，反之则有3个元素，且分布在紧邻着对角线的位置，这一种特殊的矩阵，如果能将它分解成若干基础阵在这情形下的kron积或者矩阵积，或者两者的混合，将会使得CNOT门的数量减少至多一半，我们将该矩阵分块可得：



其中，



可以看出这些阵具有良好的性质，然而第四个子阵的存在是我们难以将其写进递推里，对前三个，我们有：



而，对第四个：



但是好在，利用这些规律我们可以将表出



下面探究和的关系：

K1=[1 0 0 0;

0 0 0 exp(1i\*pi/4);

0 0 exp(1i\*pi/4) 0;

0 1 0 0];

CNOT0\*K1

ans = 4×4 complex

1.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 1.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.7071 + 0.7071i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.7071 + 0.7071i

K1\*CNOT0

ans = 4×4 complex

1.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.7071 + 0.7071i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.7071 + 0.7071i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 1.0000 + 0.0000i

CNOT1\*K1

ans = 4×4 complex

1.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.7071 + 0.7071i  
 0.0000 + 0.0000i 1.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.7071 + 0.7071i 0.0000 + 0.0000i

K1\*CNOT1

ans = 4×4 complex

1.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.7071 + 0.7071i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.7071 + 0.7071i  
 0.0000 + 0.0000i 1.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i

可得



同理可得



这样，我们可以将式子写成



这样，如果我们可以使用8个量子比特，前四个构建第一个对应的量子线路，后四个构建后一项的量子线路，分别经过单元组合门后再叠加起来，理论上是可以只使用5个CNOT门的。

我们再尝试另一个方法，如下是在4qubit的情形下的线路：

|  |
| --- |
|  |

值得指出的是，即使这里指定辅助位不能使用同样方法制备，而是要用SWAP，效果不会改变，只是所有此类线路的门数要增加一个相同值而已。

如下图所示：

|  |
| --- |
|  |

但是如果可以使用相同制备方法，那么增加辅助量子比特显然可以帮助我们减少CNOT门的数量，这也是为什么前面矩阵的式子里，逆在左边，因为这样该算子的等效量子门可以放在最左边，从而避免放最右边时还要在辅助量子比特里加入新的CNOT门这种尴尬的情形。

类似于上面的方法，我们可以来玩一点新花样，如下所示，我们设



构建线路算子



即如下线路

|  |
| --- |
|  |

我们设



编程求解出：

RZ1\_3;

RZ2\_3;

RZ3\_3;

Z0\_3=kron(eye(2),Z0);

Z1\_3=kron(eye(2),Z1);

Z2\_3=kron(eye(2),Z2);

Z3\_3;

prodZ\_3=Z0\_3\*Z1\_3+Z1\_3\*Z2\_3+Z0\_3\*Z2\_3;

Ufinal=expm(-1j\*pi/8\*prodZ\_3);

J=kron(CNOT1,CNOT1)\*kron(kron(eye(2),expm(-1j\*pi/8\*Z)),kron(eye(2),expm(-1j\*pi/8\*Z)))\*kron(CNOT1,CNOT1);

M1=RZ2\_3\*J;

M2=J\*RZ2\_3;

Kf1=(M1)\Ufinal

Kf1 = 16×16 complex

1.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.7071 + 0.7071i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.7071 + 0.7071i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i -0.0000 + 1.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 1.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.7071 - 0.7071i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.7071 - 0.7071i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 - 1.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.7071 - 0.7071i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 1.0000 - 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i

Kf2=M2\Ufinal

Kf2 = 16×16 complex

1.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.7071 + 0.7071i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.7071 + 0.7071i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i -0.0000 + 1.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 1.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.7071 - 0.7071i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.7071 - 0.7071i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 - 1.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.7071 - 0.7071i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i  
 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 1.0000 - 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i

Kf2==Kf1

ans = 16×16 logical array

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

我们使用相同的方法分解得到：



把简写为，分解成克劳里克积的形式可得：



它显然具有十分优美的形式，虽然不符合题意，但是我们把它实现如下

|  |
| --- |
|  |

请注意这里我们是如何实现这两个门的，这里因为初始态为，所以要忽略q[2]上的所有制备门操作，同理，对q[6]进行X操作后把它变换为，同时也忽略所有对它的制备门操作，这样就可以得到两簇通过分量的量子系统，再让它们分别通过，测量分布，并叠加表示，即可得到与原运算相同的结果，也就是在使用4个CNOT的基础上完成了题目要求，其实我们猜测CNOT门的个数可以更少，最少应该2个就足够，只不过没能验证，这种方法主要的困难在于描述产生湮灭门，以及多簇量子如何分别经过同一量子门。

对于本题，提交的结果是一种简单的思路，使用4 qubit 6 CNOT，线路如下：

|  |
| --- |
|  |

需要做一些小的转换，因为最下方协助比特还处在制备态，所以可以从测得的4比特状态中还原出前三比特：



由此式编写python代码，得到输出：

|  |
| --- |
|  |