Nota 10: Digital emulation of quantum circuits, TR1

Technical report 1, April 2024

A program that simulates quantum circuits is reported. The goal is obviously not to replace packages such as *qisqit* or *cirq* by an own made program. The goal is the exploration of the computation resources necessary to emulate a quantum circuit.

1. Quantum register states

Consider a quantum circuit made up of n qubits q_0, q_1, …, q_{n-1}. Its state

$$|q_0\rangle \times |q_1\rangle \times \ldots \times |q_{n-2}\rangle \times |q_{n-1}\rangle$$

is defined by an expression such as

$$a_0|00\ldots00\rangle + a_1|00\ldots01\rangle + a_2|00\ldots10\rangle + a_{N-1}|11\ldots11\rangle,$$

where $N = 2^n$ and the coefficients a_k are complex numbers. The physical meaning of this expression is the following: when measuring the register state, the result will be 00…00 with a probability equal to $|a_0|^2$, 00…01 with a probability equal to $|a_1|^2$, and so on. This is one of the enormous differences between digital and quantum circuits: the state of an n-bit register is a dimension-n vector over the binary field, while the state of an n-qubit register is a dimension-2^n vector over the complex field C.

The following piece of program (`nota10_1.py`) defines an n-qubit register modelled by a list of complex numbers and initialized in the ground state $|0\ldots00\rangle$:

```
state = [1]
for i in range(2**n-1):
    state = state + [0]
```

Thus, with n = 5, the initial state is

[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

2. Quantum gates

Quantum gates are predefined unitary operations that modify the quantum state. A fundamental property of quantum operations is that any unitary operator on an n-qubit register can be decomposed into 1-qubit unitary operations and 2-qubit CX operations ([1], [2]).

2.1. Unary operators

In matrix form, the unary operators are defined by 2×2 matrices over the complex field:

$$U = \begin{bmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{bmatrix}.$$

Assume that the operator U acts on the qubit number k, without modifying the state of the other qubits. The corresponding operation is defined by a $2^n \times 2^n$ matrix

$$A = I_k \times U \times I_{n\text{-}k\text{-}1},$$

where I_m denotes the $2^m \times 2^m$ identity matrix and $\times$ is the Kronecker product.

The following function defines the operation executed by a generic unary operator:

```
def U(k,u00,u01,u10,u11):
    m = 2**(n-k-1)
    p = 2**k
    for j in range (p):
        for i in range (2*m*j,(2*j+1)*m):
            a = u00*state[i] + u01*state[m+i]
            state[m+i] = u10*state[i] + u11*state[m+i]
            state[i] = a
```

The following functions define the transformations executed by a Hadamard gate, an X gate and an R_φ gate, on the qubit number k:

```
### HADAMARD OPERATOR ON QUBIT NUMBER K ###
def H(k):
    U(k,1/(2**0.5),1/(2**0.5),1/(2**0.5),-1/(2**0.5))
```

```
### X OPERATOR ON QUBIT NUMBER K ###
def X(k):
    U(k,0,1,1,0)

### Rphi OPERATOR ON QUBIT NUMBER K ###
def Rphi(k,phi):
    U(k,1,0,0,np.exp(1j*phi))
```

For example (`nota10_1.py, example 1`), execute

```
print(state,"\n")
X(1)
H(1)
Rphi(1,np.pi/2)
print(state,"\n")
```

with n = 5 and initial state = $|00000\rangle$. The result is

```
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0]

[0.7071067811865475, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
(-4.329780281177466e-17-0.7071067811865475j), 0j, 0j, 0j, 0j, 0j, 0j, 0j,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0j, 0j, 0j, 0j, 0j, 0j, 0j, 0j]
```

Chek that ($10^{-17} \cong 0$)

$$|00000\rangle \xrightarrow{X(1)} |01000\rangle \xrightarrow{H(1)} \frac{1}{\sqrt{2}}|00000\rangle - \frac{1}{\sqrt{2}}|01000\rangle \xrightarrow{R_{\pi/2}(1)} \frac{1}{\sqrt{2}}|00000\rangle - \frac{i}{\sqrt{2}}|01000\rangle.$$

2.2. *CU* operators

Consider a *CU* operator that acts on the qubit number *k*, under the control of the qubit number *l*, without modifying the state of the other qubits. It is defined by the following function in which `ToBin(i,n)` returns a list whose elements are the binary representation of *i* with *n* bits. It is derived from the definition of `U(k,u00,u01,u10,u11)`:

```
def CU(l,k,u00,u01,u10,u11):
    m = 2**(n-k-1)
    p = 2**k
    for j in range (p):
        for i in range (2*m*j,(2*j+1)*m):
            bin = ToBin(i,n)
            if bin[l] == 1:
                a = u00*state[i] + u01*state[m+i]
                state[m+i] = u10*state[i] + u11*state[m+i]
                state[i] = a
```

The following functions define the transformations executed by a CX gate and by a CR_φ gate, on qubit number l under the control of qubit number k:

```
### OPERATOR CX ON QUBIT NUMBER L CONTROLLED BY QUBIT NUMBER K ###
def CX(k,l):
    CU(k,l,0,1,1,0)

### OPERATOR CRphi ON QUBIT NUMBER L CONTROLLED BY QUBIT NUMBER K ###
def CRphi(k,l,phi):
    CU(k,l,1,0,0,np.exp(1j*phi))
```

For example (`nota10_1.py, example 2`), execute

```
print(state,"\n")
X(1)
X(2)
CX(1,2)
X(3)
CRphi(1,3,np.pi/4)
print(state,"\n")
```

with n = 5 and initial state = $|00000\rangle$. The result is

```
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0]

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, (0.7071067811865476+0.7071067811865476j),
0j, 0, 0, 0j, 0j, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0j, 0j, 0, 0, 0j, 0j]
```

Chek that

$$|00000\rangle \xrightarrow{X(2)X(1)} |01100\rangle \xrightarrow{CX(1,2)} |01000\rangle \xrightarrow{X(3)} |01010\rangle \xrightarrow{R_{\pi/4}(1,3)} e^{i\pi/4}|01010\rangle.$$

2.3. Operator *SWAP*

The $SWAP$ operator can be synthesized with three CX operators. The following function define an operator that swap qubits number k and l:

```
### OPERATOR SWAP(K,L) ###
def SWAP(k,l):
    CX(k,l)
    CX(l,k)
    CX(k,l)
```

2.4. Example

The following program (`nota10_1.py, example 3`) computes the quantum Fourier on qubits number 1 to 4, with initial state = $|00101\rangle$:

```
X(2)
```

```
X(4)
print(state,"\n")
H(1)
CRphi(2,1,np.pi/2)
CRphi(3,1,np.pi/4)
CRphi(4,1,np.pi/8)
H(2)
CRphi(3,2,np.pi/2)
CRphi(4,2,np.pi/4)
H(3)
CRphi(4,3,np.pi/2)
H(4)
SWAP(1,2)
SWAP(3,4)
SWAP(2,3)
SWAP(1,2)
SWAP(3,4)
SWAP(2,3)
print(state,"\n")
```

The result is

```
[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0]

[(0.24999999999999992+0j),
(-0.09567085809127238+0.2309698831278216j),
(-0.17677669529663684-0.17677669529663684j),
(0.2309698831278216-0.09567085809127242j),
(1.530808498934191e-17+0.24999999999999992j),
(-0.2309698831278216-0.09567085809127236j),
(0.1767766952966368-0.17677669529663684j),
(0.09567085809127245+0.2309698831278216j),
(-0.24999999999999992+0j),
(0.09567085809127238-0.2309698831278216j),
(0.17677669529663684+0.17677669529663684j),
(-0.2309698831278216+0.09567085809127242j),
(-1.530808498934191e-17-0.24999999999999992j),
(0.2309698831278216+0.09567085809127236j),
(-0.1767766952966368+0.17677669529663684j),
(-0.09567085809127245-0.2309698831278216j),
0j, 0j, 0j, 0j, 0j, 0j, 0j, 0j, 0j, 0j, 0j, 0j, 0j, 0j, 0j, 0j]
```

Apart from two computation inaccuracies ($10^{-17} \cong 0$), the result is the same as in [1], example 7.3, that was computed with *cirq*.

3. Digital circuit

A specific digital circuit able to emulate the execution of quantum algorithms is specified.

3.1. Memory resources

As only unary and binary operations are considered, the quantum state could be stored within a dual port *RAM*, as long as the number n of qubits is not too high. This memory stores 2^n complex numbers. Assuming that each complex number is represented by a pair of t fixed-point binary numbers (nota 9, Sec.1), the memory size is equal to $t \cdot 2^{n+1}$ bits. Observe that, even in the case of a few tenths of qubits, this could be an astronomical number.

3.2. Computation resource

The computations executed by the functions $U(k,u_{00},u_{01},u_{10},u_{11})$ and $CU(l,k,u_{00},u_{01},u_{10},u_{11})$, are

$$next\_state(i) = u_{00}\, state(i) + u_{01}\, state(m+i),$$

$$next\_state(m+i) = u_{10}\, state(i) + u_{11}\, state(m+i).$$

The symbol of a computation resource that executes those operations is shown in Fig.1.

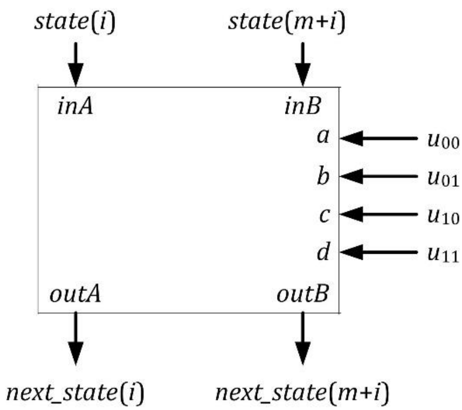

Figure 1 Computation resource

In total, this resource computes four products of two complex numbers, that is, sixteen products of two fixed point numbers. Nevertheless, in most cases, the number of products is smaller. For example, if every coefficient

u_{ij} is either real, or imaginary, but not strictly complex, the number of products is equal to eight. This is the case of the operators H, X, Y, Z, $R_x(\gamma)$, $R_y(\gamma)$. Another example is when two coefficients u_{ij} are equal to 0. This is the case of the operators $R_y(\gamma)$ and R_φ.

3.3. Circuit structure

An example of circuit structure is shown in Fig.2. It consists of

- a data memory (a dual port *RAM*) that stores the quantum state;
- a program memory that stores a sequence of operations, for example

```
initialize
X(2)
X(4)
H(1)
CRphi(2,1,np.pi/2)
…
```

- a control unit that interprets the instructions.

The control unit executes an interpretation program. For example, in seudo Python code:

```
reset
not_end = true
while not_end:
    read_next operation
    if operation == initialize:
        a = 0
        b = 0
        c = 0
        d = 0
        for j in range (2**n):
            write data
    elif operation == U(k,u00,u01,u10,u11):
        m = 2**(n-k-1)
        p = 2**k
        a = u00
        b = u01
        c = u10
        d = u11
        for j in range (p):
            for i in range (2*m*j,(2*j+1)*m):
                addA = i
                addB = m+i
```

```
                write data
        elif operation == CU(l,k,u00,u01,u10,u11):
            m = 2**(n-k-1)
            p = 2**k
            a = u00
            b = u01
            c = u10
            d = u11
            for j in range (p):
                for i in range (2*m*j,(2*j+1)*m):
                    addA = i
                    addB = m+i
                    if addA(l) == 1:
                        write data
        else:
            not_end = false
```

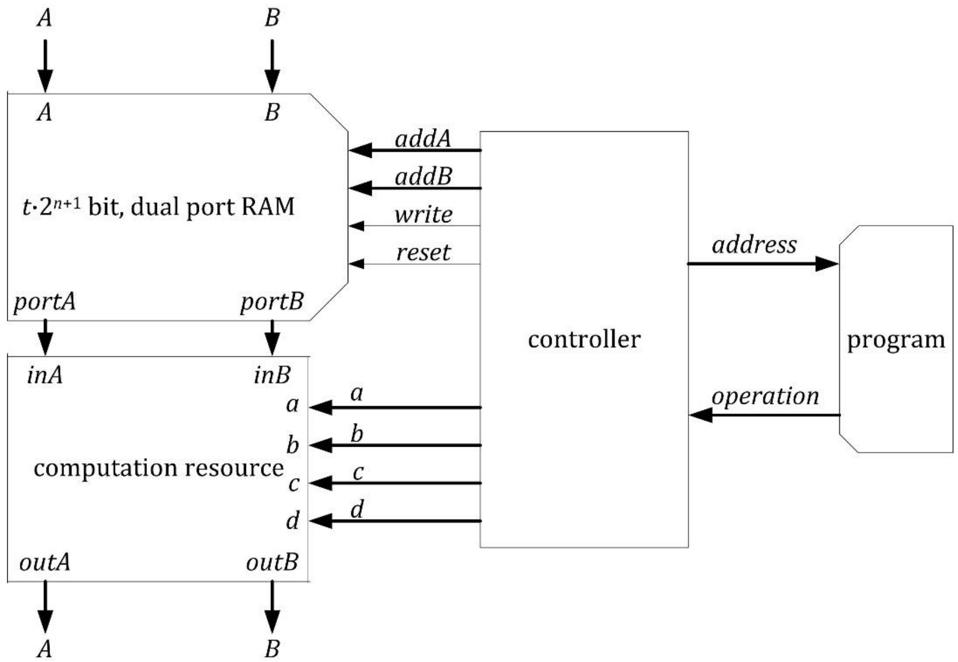

Figure 2 Circuit structure

The computation time of every operation is proportional to 2^n. Thus, the time complexity, as well as the space complexity, are proportional to 2^n, so that the emulation of quantum circuits including more than a few tenth of qubits is not possible. Examples of emulation circuits, up to 30 qubits, are reported in [3].

References

[1] J.P.Deschamps, Computación Cuántica, Marcombo, Barcelona, 2023.

[2] M.A.Nielsen and I.L.Chuang, Quantum Computation and Quantum Information, Cambridge, Cambridge University Press, 2010.

[3] N.Mahmud, E.El-Araby and D.Caliga, Scaling reconfigurable emulation of quantum algorithms at high precision and high throughput, Quantum Engineering, Wiley, 2019,1:e19.