Nota 9: Digital emulation of a quantum circuit

Quantum algorithms can be simulated by general purpose digital computers. However, this type of simulation does not deal efficiently with the parallelism inherent to quantum algorithms. For that reason, nowadays, the design of FPGA-based, special purpose emulators, is considered.

As a very initial step toward the study and development of circuits that emulate the working of a quantum algorithms, a digital circuit that executes quantum operations on a 3-qubit register has been designed.

1. Representation of complex numbers

Complex numbers $x = a+bi$ are represented by pairs of fixed-point 2's complement integers:

$$a = a_9\, a_8\, .\, a_7\, a_6\, a_5\, a_4\, a_3\, a_2\, a_1\, a_0,\; b = b_9\, b_8\, .\, b_7\, b_6\, b_5\, b_4\, b_3\, b_2\, b_1\, b_0.$$

Taking into account that the module $|x|$ of all processed numbers x is smaller than or equal to 1, the coefficients a and b belong to the following interval:

$$-1 \leq a, b \leq 1.$$

Define $A = a\cdot 2^8$ and $B = b\cdot 2^8$. They belong to the interval

$$-2^8 \leq A, B \leq 2^8,$$

and their product $P = A\cdot B$ belongs to the interval

$$-2^{16} \leq P \leq 2^{16},$$

so that, in 2's complement,

$$P = P_{17}\, P_{16}\, P_{15}\, P_{14} \cdots P_0.$$

For example,

$$1100\cdots0 = -2^{17} + 2^{16} = -2^{16}, 0100\cdots0 = 2^{16}.$$

Thus, the product $a\cdot b$ is equal to

$$a\cdot b = A\cdot B\cdot 2^{-16} = P_{17}\, P_{16}\, .P_{15}\, P_{14}\cdots\, P_0 \cong P_{17}\, P_{16}\, .\, P_{15}\, P_{14}\cdots\, P_8.$$

The following VHDL entity (`nota9.vhd`) computes $a\cdot b$:

```
ENTITY FP_MUL IS
   PORT(
      A, B: IN SIGNED(n-1 DOWNTO 0);
      C: OUT SIGNED(n-1 DOWNTO 0)
      );
END FP_MUL;

ARCHITECTURE CIRCUIT OF FP_MUL IS
   SIGNAL long_A, long_B: SIGNED(2*n-1 DOWNTO 0);
   SIGNAL long_C: SIGNED(4*n-1 DOWNTO 0);
BEGIN
   long_A(2*n-1 DOWNTO n)<= (OTHERS => A(n-1));
   long_A(n-1 DOWNTO 0)<= A;
   long_B(2*n-1 DOWNTO n)<= (OTHERS => B(n-1));
   long_B(n-1 DOWNTO 0)<= B;
   long_C <=  long_A*long_B;
   C <= long_C(2*n-3 DOWNTO n-2);
```

2. Representation of quantum register states

The quantum state of a 3-qubit register (q_0, q_1, q_2) is represented by a pair (S, T) of *Q_STATE*-type signals, where

```
TYPE Q_STATE IS ARRAY (NATURAL RANGE <>) OF SIGNED(n-1 DOWNTO 0);
```

Thus, the quantum state of a 3-qubit register is

$(S(0)+iT(0))|000\rangle + (S(1)+iT(1))|001\rangle + (S(2)+iT(2))|010\rangle +$

$(S(3)+iT(3))|011\rangle + (S(4)+iT(4))|100\rangle + (S(5)+iT(5))101\rangle +$

$(S(6)+iT(6))|110\rangle + (S(7)+iT(7))|111\rangle.$

3. Quantum gates

Quantum gates are predefined operations that modify the quantum state. The following gates have been defined:

*H*_gate: Hadamard operator on q_0,

H_gate_1: Hadamard operator on q_1,

H_gate_2: Hadamard operator on q_2,

CX_01: controlled X gate on q_0 (control qubit) and q_1 (target qubit),

CR_PI_2_10: controlled $R_{\pi/2}$ operator on q_1 (control qubit) and q_0 (target qubit),

CR_PI_2_21: controlled $R_{\pi/2}$ operator on q_2 (control qubit) and q_1 (target qubit),

CR_PI_4_20: controlled $R_{\pi/4}$ operator on q_2 (control qubit) and q_0 (target qubit).

In matrix form, H_gate executes the following operation on qubit q_0

$$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix},$$

and doesn't modify the state of qubits q_1 and q_2. Thus, it executes the following transformation of the register state:

$(S(0)+iT(0))|000\rangle + (S(1)+iT(1))|001\rangle +$

$(S(2)+iT(2))|010\rangle + (S(3)+iT(3))|011\rangle +$

$(S(4)+iT(4))|100\rangle + (S(5)+iT(5))101\rangle +$

$(S(6)+iT(6))|110\rangle + (S(7)+iT(7))|111\rangle$

$\rightarrow$

$\frac{1}{\sqrt{2}}(S(0)+iT(0))(|000\rangle +|100\rangle) + \frac{1}{\sqrt{2}}(S(1)+iT(1))(|001\rangle +|101\rangle) +$

$\frac{1}{\sqrt{2}}(S(2)+iT(2))(|010\rangle +|110\rangle) + \frac{1}{\sqrt{2}}(S(3)+iT(3))(|011\rangle +|111\rangle) +$

$\frac{1}{\sqrt{2}}(S(4)+iT(4))(|000\rangle -|100\rangle) + \frac{1}{\sqrt{2}}(S(5)+iT(5))(|001\rangle - |101\rangle) +$

$\frac{1}{\sqrt{2}}(S(6)+iT(6))(|010\rangle -|110\rangle) + \frac{1}{\sqrt{2}}(S(7)+iT(7))(|011\rangle -|111\rangle)$

=

$\frac{1}{\sqrt{2}}(S(0)+iT(0) + S(4)+iT(4))|000\rangle + \frac{1}{\sqrt{2}}(S(1)+iT(1) + S(5)+iT(5)|001\rangle +$

$\frac{1}{\sqrt{2}}(S(2)+iT(2) + S(6)+iT(6))|010\rangle + \frac{1}{\sqrt{2}}(S(3)+iT(3) + S(7)+iT(7))|011\rangle +$

$\frac{1}{\sqrt{2}}(S(0)+iT(0) - S(4) - iT(4))|100\rangle + \frac{1}{\sqrt{2}}(S(1)+iT(1) - S(5)- iT(5))|101\rangle +$

$\frac{1}{\sqrt{2}}(S(2)+iT(2) - S(6) - iT(6))|110\rangle + \frac{1}{\sqrt{2}}(S(3)+iT(3) - S(7) - iT(7))|111\rangle.$

This is the corresponding entity (`nota9.vhd`):

```
ENTITY H_gate IS
PORT(
 S, T: IN Q_STATE(0 TO 7);
 Next_S, Next_T: OUT Q_STATE(0 TO 7)
);
END H_gate;

ARCHITECTURE circuit OF H_gate IS
 COMPONENT FP_MUL IS
   PORT(
      A, B: IN SIGNED(n-1 DOWNTO 0);
      C: OUT SIGNED(n-1 DOWNTO 0)
      );
   END COMPONENT;
  SIGNAL S1, T1: Q_STATE(0 TO 7);
BEGIN
   steps: FOR i IN 0 TO 3 GENERATE
      S1(i) <= S(i) + S(i+4); T1(i) <= T(i) + T(i+4);
      S1(i+4) <= S(i) - S(i+4); T1(i+4) <= T(i) - T(i+4);
      multiplier1: FP_MUL PORT MAP(A => S1(i), B => inv_sqrt_2,
         C => Next_S(i));
      multiplier2: FP_MUL PORT MAP(A => S1(4+i), B => inv_sqrt_2,
         C => Next_S(4+i));
      multiplier3: FP_MUL PORT MAP(A => T1(i), B => inv_sqrt_-2,
         C => Next_T(i));
      multiplier4: FP_MUL PORT MAP(A => T1(4+i), B => inv_sqrt_2,
         C => Next_T(4+i));
   END GENERATE;
END circuit;
```

The other Hadamard gates are defined in a similar way (`nota9.vhd`).

Another example: in matrix form, the $R_{\pi/4}$ operator is

$$\begin{bmatrix} 1 & 0 \\ 0 & \frac{1+i}{\sqrt{2}} \end{bmatrix}.$$

Thus, the controlled $R_{\pi/4}$ operator on q_2 (control qubit) and q_0 (target qubit) executes the following transformation of the register state:

$(S(0)+iT(0))|000\rangle$ + $(S(1)+iT(1))|001\rangle$ +

$(S(2)+iT(2))|010\rangle$ + $(S(3)+iT(3))|011\rangle$ +

$(S(4)+iT(4))|100\rangle$ + $\frac{1+i}{\sqrt{2}}(S(5)+iT(5))101\rangle$ +

$(S(6)+iT(6))|110\rangle$ + $\frac{1+i}{\sqrt{2}}(S(7)+iT(7))|111\rangle$

=

$(S(0)+iT(0))|000\rangle$ + $(S(1)+iT(1))|001\rangle$ +

$(S(2)+iT(2))|010\rangle$ + $(S(3)+iT(3))|011\rangle$ +

$(S(4)+iT(4))|100\rangle$ + $\frac{1}{\sqrt{2}}(S(5)+iT(5)+iS(5)-T(5))101\rangle$ +

$(S(6)+iT(6))|110\rangle$ + $\frac{1}{\sqrt{2}}(S(7)+iT(7)+iS(7)-T(7))|111\rangle$.

This is the corresponding entity (`nota9.vhd`):

```
ENTITY CR_PI_4_20 IS
PORT(
 S, T: IN Q_STATE(0 TO 7);
 Next_S, Next_T: OUT Q_STATE(0 TO 7)
);
END CR_PI_4_20;

ARCHITECTURE circuit OF CR_PI_4_20 IS
 COMPONENT FP_MUL IS
   PORT(
      A, B: IN SIGNED(n-1 DOWNTO 0);
      C: OUT SIGNED(n-1 DOWNTO 0)
      );
   END COMPONENT;
   SIGNAL S5, T5, S7, T7: SIGNED(9 DOWNTO 0);
BEGIN
   Next_S(0) <= S(0) ; Next_T(0) <= T(0);
   Next_S(1) <= S(1) ; Next_T(1) <= T(1);
   Next_S(2) <= S(2) ; Next_T(2) <= T(2);
   Next_S(3) <= S(3) ; Next_T(3) <= T(3);
   Next_S(4) <= S(4) ; Next_T(4) <= T(4);
   Next_S(6) <= S(6) ; Next_T(6) <= T(6);
   S5 <= S(5)-T(5) ; T5 <= S(5) + T(5);
   component1: FP_MUL PORT MAP (S5, inv_sqrt_2, Next_S(5));
   component2: FP_MUL PORT MAP (T5, inv_sqrt_2, Next_T(5));
   S7 <= S(7)-T(7) ; T7 <= S(7) + T(7);
   component3: FP_MUL PORT MAP (S7, inv_sqrt_2, Next_S(7));
   component4: FP_MUL PORT MAP (T7, inv_sqrt_2, Next_T(7));
END circuit;
```

The other controlled gates are defined in a similar way (`nota9.vhd`).

4. Quantum Fourier Transform

As an example of digital emulation, consider the quantum Fourier transform on a 3-qubit register. The corresponding quantum circuit is shown in Fig.1 (Fig.7.6 of [1] with $n = 3$).

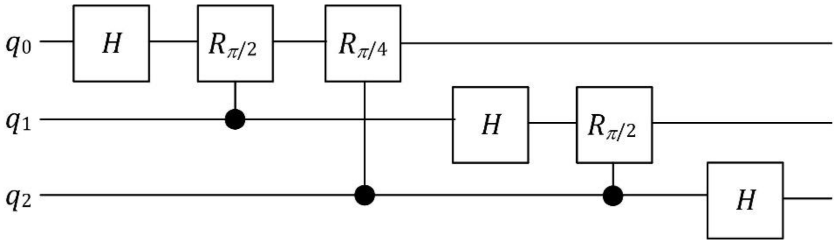

Figure 1 Quantum Fourier transform (3 qubits)

The corresponding digital circuit, based on the previously defined components, is shown in Fig.2.

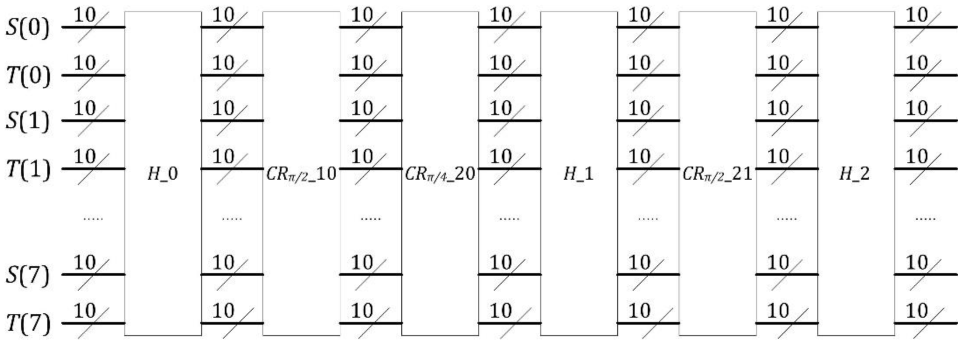

Figure 2 Digital emulation of the Quantum Fourier Transform (3 qubits)

This is the corresponding entity (`nota9.vhd`):

```
ENTITY QFT_3 IS
PORT(
 S, T: IN Q_STATE(0 TO 7);
 Next_S, Next_T: OUT Q_STATE(0 TO 7)
);
END QFT_3;

ARCHITECTURE circuit OF QFT_3 IS
   COMPONENT H_gate IS
     PORT(
      S, T: IN Q_STATE(0 TO 7);
      Next_S, Next_T: OUT Q_STATE(0 TO 7)
     );
   END COMPONENT;
   COMPONENT H_gate_1 IS
     PORT(
      S, T: IN Q_STATE(0 TO 7);
      Next_S, Next_T: OUT Q_STATE(0 TO 7)
     );
   END COMPONENT;
   COMPONENT H_gate_2 IS
     PORT(
      S, T: IN Q_STATE(0 TO 7);
      Next_S, Next_T: OUT Q_STATE(0 TO 7)
     );
   END COMPONENT;
   COMPONENT CR_PI_2_10 IS
     PORT(
      S, T: IN Q_STATE(0 TO 7);
      Next_S, Next_T: OUT Q_STATE(0 TO 7)
     );
   END COMPONENT;
   COMPONENT CR_PI_4_20 IS
     PORT(
      S, T: IN Q_STATE(0 TO 7);
      Next_S, Next_T: OUT Q_STATE(0 TO 7)
     );
   END COMPONENT;
   COMPONENT CR_PI_2_21 IS
     PORT(
      S, T: IN Q_STATE(0 TO 7);
      Next_S, Next_T: OUT Q_STATE(0 TO 7)
     );
   END COMPONENT;
   SIGNAL S1, T1, S2, T2, S3, T3, S4, T4, S5, T5, S7, T7: Q_STATE(0 TO 7);
BEGIN
   component1: H_gate PORT MAP (S, T, S1, T1);
   component2: CR_PI_2_10 PORT MAP (S1, T1, S2, T2);
   component3: CR_PI_4_20 PORT MAP (S2, T2, S3, T3);
   component4: H_gate_1  PORT MAP (S3, T3, S4, T4);
   component5: CR_PI_2_21 PORT MAP (S4, T4, S5, T5);
   component6: H_gate_2 PORT MAP (S5, T5, Next_S,Next_T);
END circuit;
```

A test of this entity, with the following initial value of S and T,

$$S = (0, 0, 0, 0, 1, 0, 0, 0),\ T = (0, 0, 0, 0, 0, 0, 0, 0),$$

that correspond to the basic quantum state $|101\rangle$, gives the following result (hexadecimal notation):

$$Next\_S = (059, 3A6, 000, 000, 3BF, 040, 040, 3BF),$$

$$Next\_T = (000, 000, 059, 3A6, 3BF, 040, 3BF, 040).$$

Taking into account the chosen representation of the complex numbers (Sec.1), the result is

$0.347|000\rangle - 0.351|001\rangle + 0.347i|010\rangle - 0.351i|011\rangle - 0.253(1+i)|100\rangle + 0.25(1+i)|101\rangle + (0.25\text{-}0.253i)|110\rangle + (-0.253 + 0.25i)|111\rangle$.

In order to check the result, the same operation has been simulated with the Python package *cirq* (`nota9.py`). The initial state is set to $|101\rangle$ with X gates.

```
import cirq
j = [1,0,1]
j1, j2, j3 = cirq.LineQubit.range(3)
qft = cirq.Circuit()
if j[0] == 1:
    qft.append(cirq.X(j1))
if j[1] == 1:
    qft.append(cirq.X(j2))
if j[2] == 1:
    qft.append(cirq.X(j3))
CRpi_2 = cirq.S.controlled()
CRpi_4 = cirq.T.controlled()
qft.append([cirq.H(j1),CRpi_2(j2, j1), CRpi_4(j3,j1)])
qft.append([cirq.H(j2),CRpi_2(j3, j2)])
qft.append([cirq.H(j3)])
#print(qft)
un_simulador = cirq.Simulator()
estado_final = un_simulador.simulate(qft)
print(estado_final)
```

This is the result:

```
qubits: (cirq.LineQubit(2), cirq.LineQubit(1), cirq.LineQubit(0))
output vector:
0.354|000⟩ + (-0.25-0.25j)|001⟩ + 0.354j|010⟩ + (0.25-0.25j)|011⟩
- 0.354|100⟩ + (0.25+0.25j)|101⟩ - 0.354j|110⟩ + (-0.25+0.25j)|111⟩
```

Apart from precision errors (fixed point numbers with only 8 fractional bits) and different enumeration order of the qubits (q_2, q_1, q_0 vs. q_0, q_1, q_2), the results are the same.

The VHDL and Python files that correspond to this note (`nota9.vhd` and `nota9.py`) are available at the web site associated to [1].

Referencias

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