Atos QLM – Fast Start Training

ANNEXE:QFT FROM SCRATCH

24/04/2018

Pierre-Antoine Harraud
HPC & Quantum expert
pierre-antoine.harraud@atos.net

Classic Fourier Transform

- Given a vector **x** of dimension 2^N with component x_j
- The Discrete Fourier Transform of vector **x** is a vector **y** of dimension 2^N of component y_k

$$y_k = \sum_{j=0}^{2^N-1} x_j \, e^{2i\pi jk/2^N}$$

- Matrix representation :

$$\boxed{\boldsymbol{y} = W\,\boldsymbol{x}} \quad \text{with} \quad W = \frac{1}{\sqrt{2^N}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{2^N-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(2^N-1)} \\ 1 & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{2^N-1} & \omega^{2(2^N-1)} & \cdots & \omega^{(2^N-1)(2^N-1)} \end{bmatrix}$$

j (→) k (↓)

$$\omega = e^{2i\pi/2^N}$$

Ket notation

- A state $|x>=|x_{q_{N-1}} x_{q_{N-2}} .. x_{q_0}>$ on N qubits corresponds to a vector **x** of dimension 2^N with components x_j

We take the computationnal basis :

$$|00..00>, |00..01>, |00..10> \cdots |11..11>$$

$$\text{i.e } |0>, |1>, |2> \cdots |2^N - 1>$$

$$|x>= \begin{pmatrix} x_0 \\ x_1 \\ \\ \\ x_{2^N-1} \end{pmatrix} \begin{matrix} |00..00> \\ |00..01> \\ \\ \\ |11..11> \end{matrix}$$

- Fourier Transform of state $|x>$ is a state $|y>$,i.e vector of 2^N components y_k

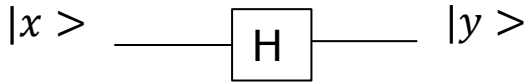
$$|y> = \sum_{k=0}^{2^N-1} y_k |k> = \sum_{k=0}^{2^N-1} \sum_{j=0}^{2^N-1} e^{2i\pi jk/2^N} x_j |k>$$

Case N = 1

- N = 1 , vector of size 2

$$|x>=\begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \quad \begin{matrix} |0> \\ |1> \end{matrix}$$

- $\omega = e^{i\pi} = -1$
- Fourier transform matrice representation: $W \equiv \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
- Quantum circuit : Hadamard gate



Mathematical representation

- Let's take a basis vector $|j> = |j_{N-1}..j_1 j_0>, j_i \in \{0,1\}$

$$|j> \xrightarrow{QFT_N} \sum_{k=0}^{2^N-1} e^{2i\pi kj/2^N} |k>$$

$$= \sum_{k=0}^{2^N-1} \omega^{jk} |k_{N-1}..k_1 k_0>$$

$$|k> = |k_{N-1}..k_1 k_0>$$

$$\omega = e^{2i\pi/2^N}$$

Mathematical set up

$$\underbrace{\sum_{k=0}^{2^N-1} \omega^{jk} \, |k_{N-1}..k_1k_0>}_{QFT_N} = \sum_{k=0}^{2^N-1} \omega^{jk} \, |k_{N-1}..k_1> |k_0>$$

$$k = \sum_{l=0}^{N-1} k_l \, 2^l$$

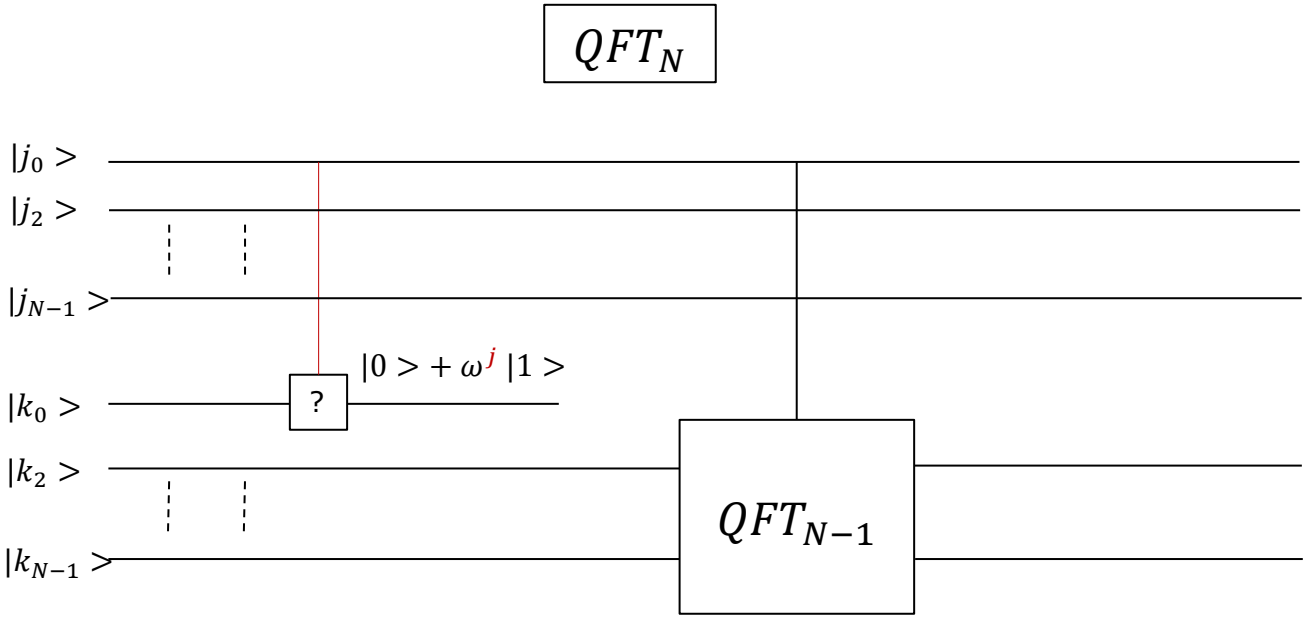
$k_0 = 0$: k even

$k_0 = 1$: k odd

$$= \sum_{k\prime=0}^{2^{N-1}-1} \omega^{j(2k')} \, |k'_{N-1}..k'_1> |0> + \sum_{k\prime=0}^{2^{N-1}-1} \omega^{j(2k'+1)} \, |k'_{N-1}..k'_1> |1>$$

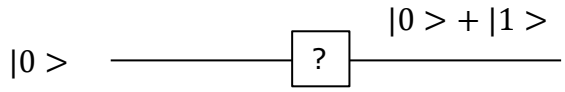
$$= \underbrace{\left(\sum_{k'=0}^{2^{N-1}-1} (\omega^2)^{jk\prime} \, |k'_{N-1}..k'_1> \right)}_{QFT_{N-1}} \left(|0> + \omega^j |1> \right)$$

Problem set up

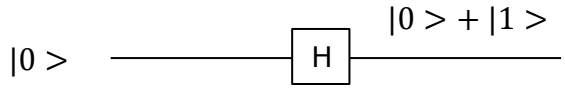


Exercise

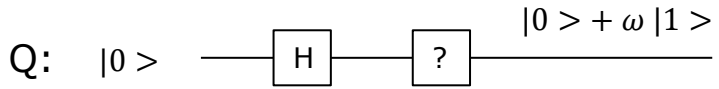
Q:



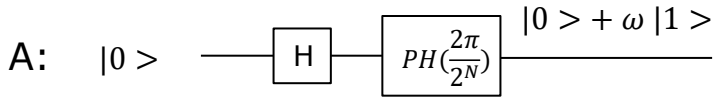
A:



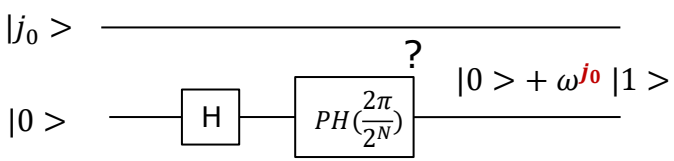
Q:



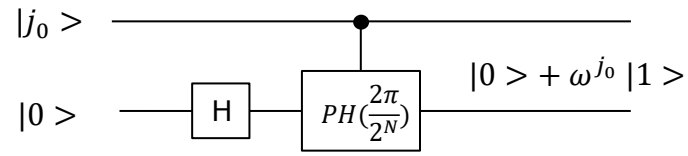
A:



Q:



A:

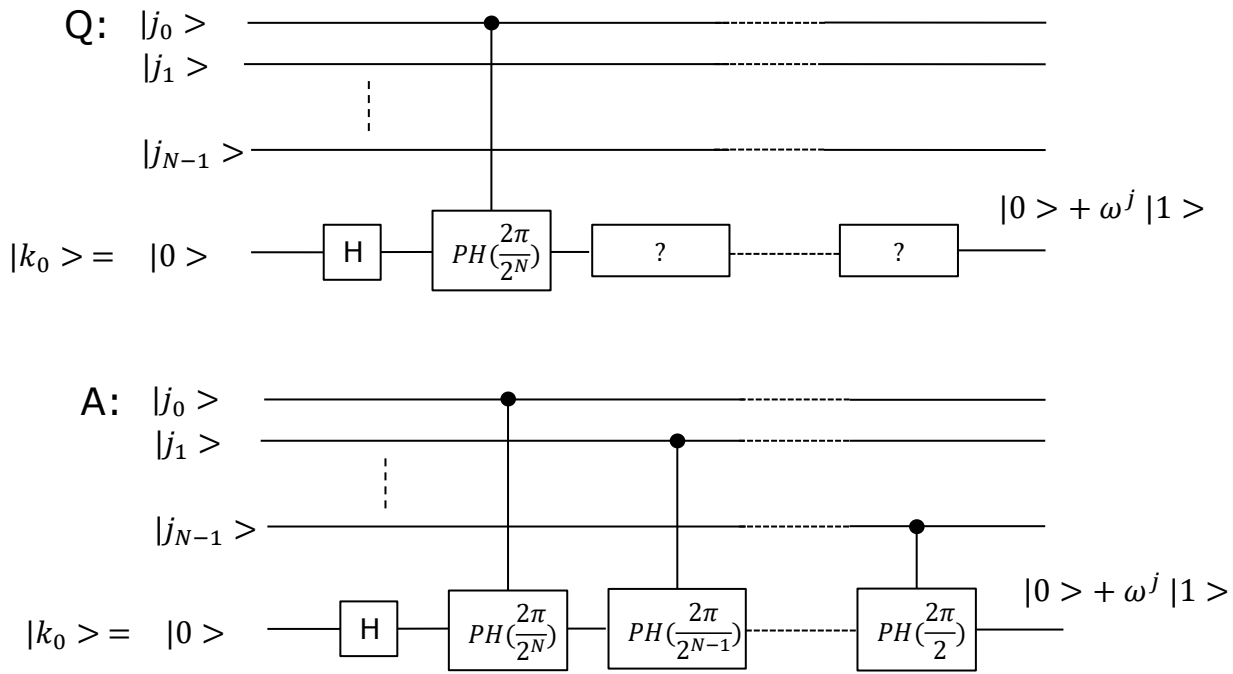


Help: $\omega = e^{2i\pi/2^N}$

$PH[\theta] = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}$

with $j_0 \in \{0,1\}$

First step of recursivity

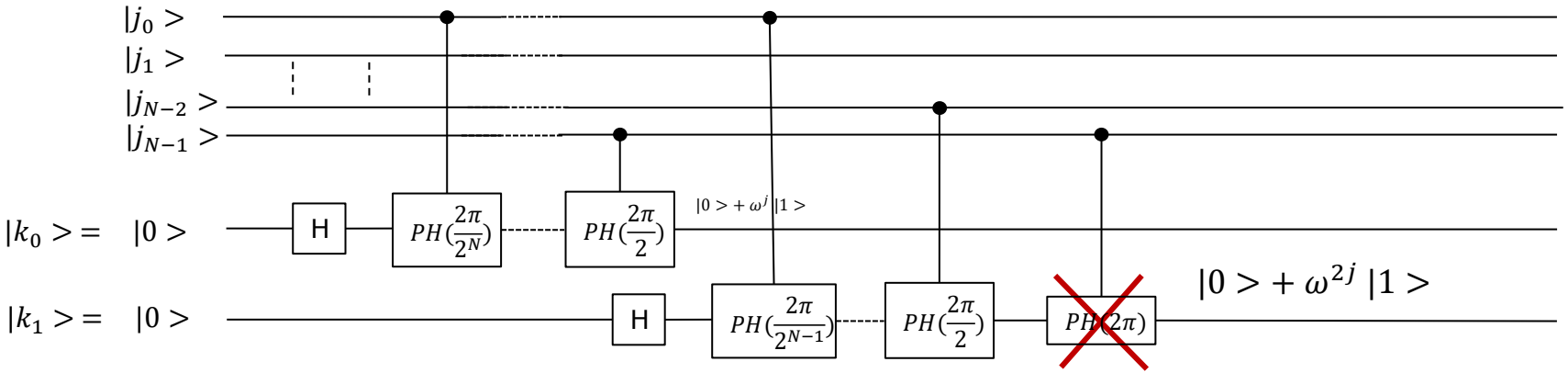


$$j = \sum_{l=0}^{N-1} j_l \, 2^l$$

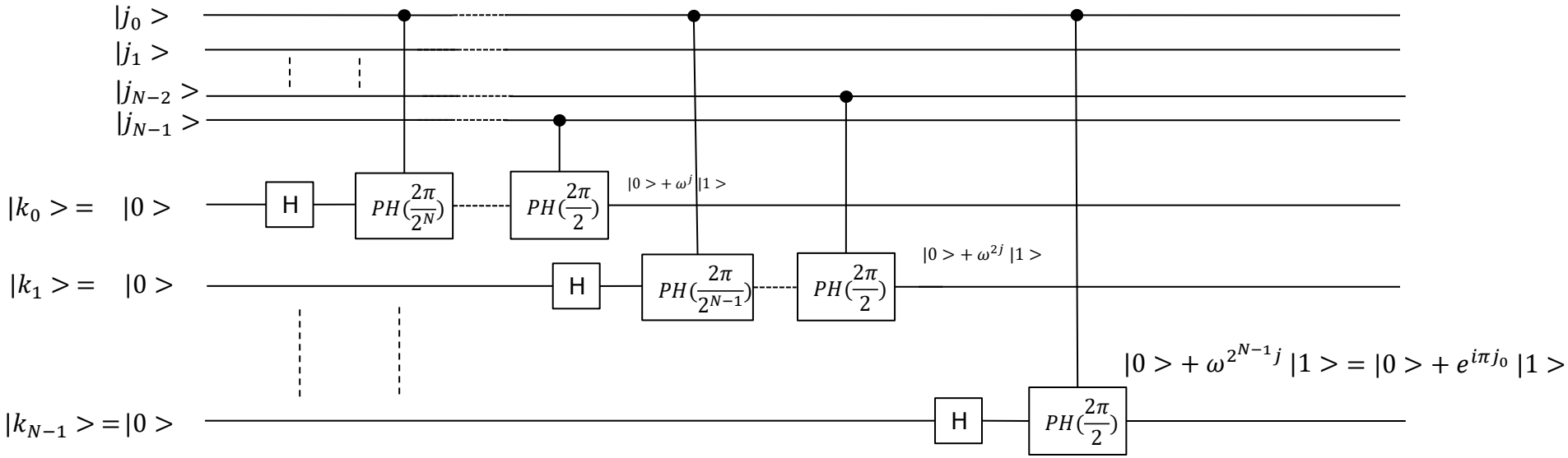
$$\omega^j = \omega^{j_0} \omega^{2 j_1} .. \omega^{2^{N-1} j_{N-1}}$$

$$\omega = e^{2i\pi/2^N}$$

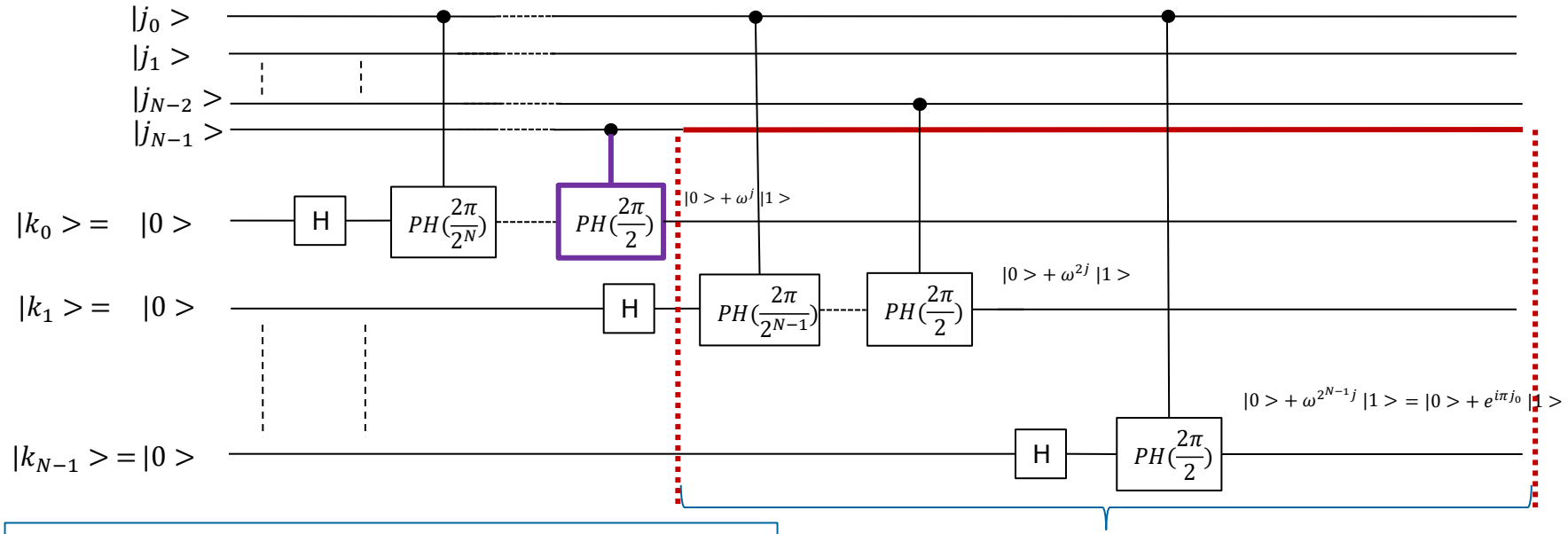
2nd step



Overall picture



Saving qubits

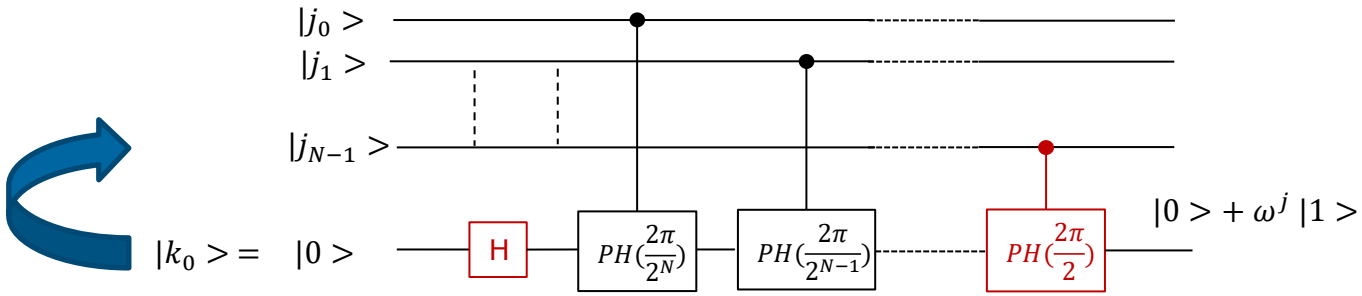


Tricks : use | j_{N-1}> to store result of | k_0>
if the violet part can be integrated

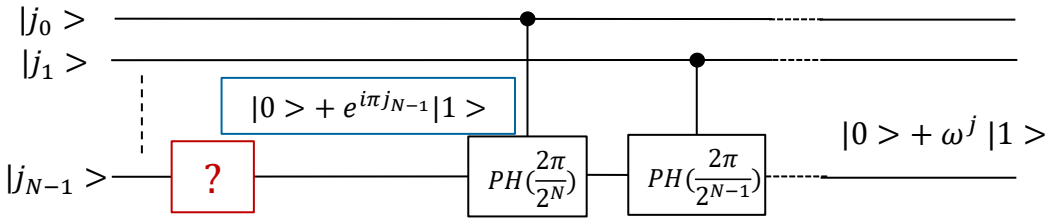
j_{N-1} not needed for this part

Exercise

Points to adress

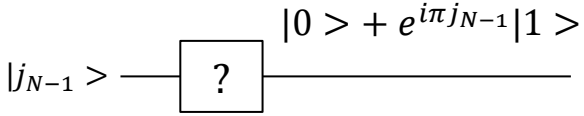


So need

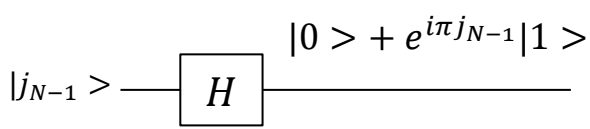


Exercise

Q:

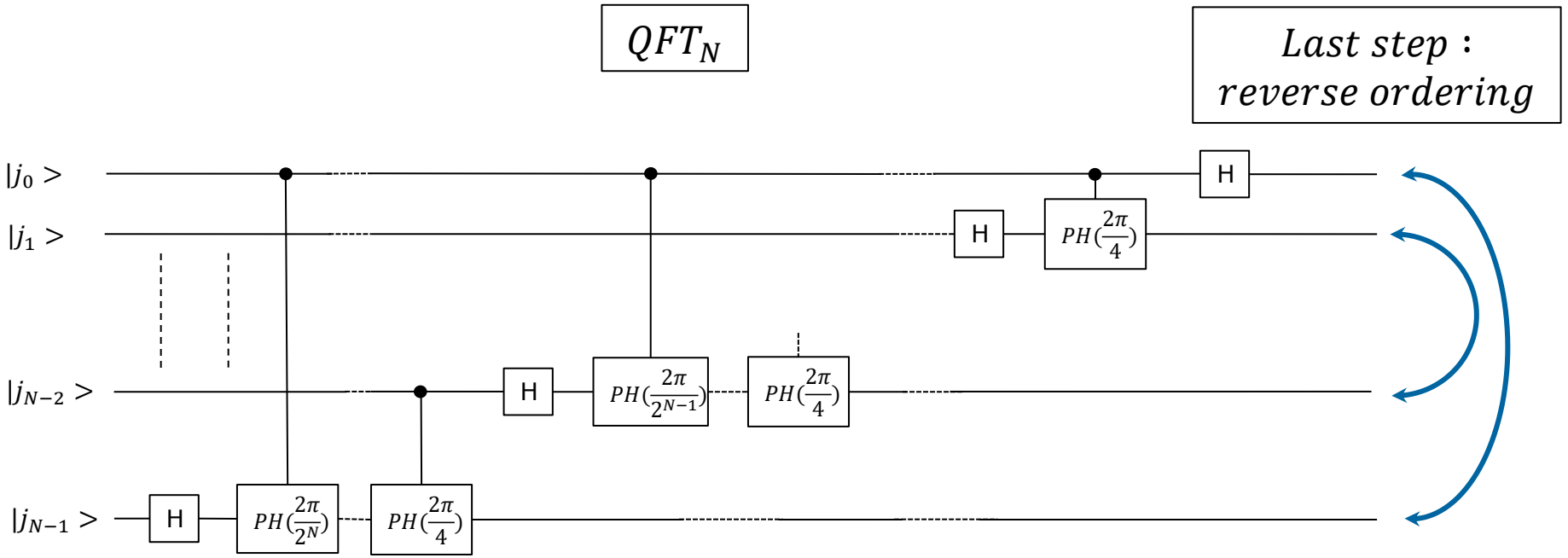


A:



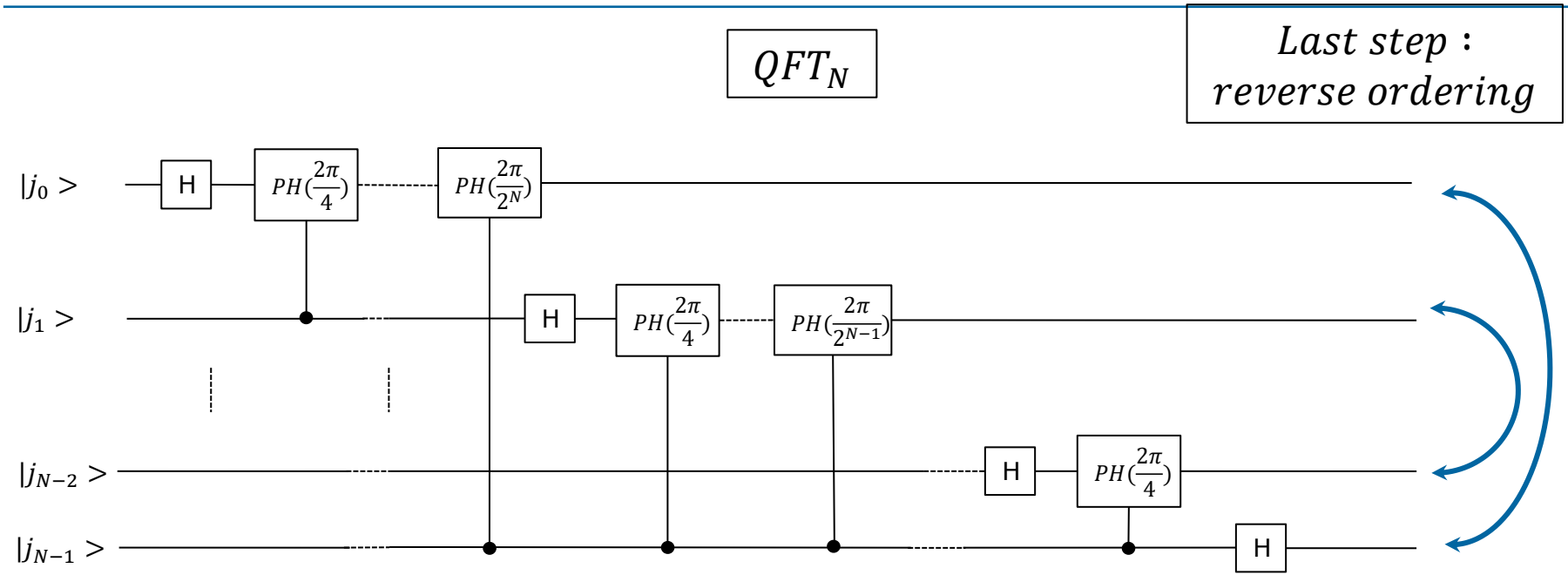
Help: $j_{N-1} \in \{0,1\}$, do both cases

Final picture



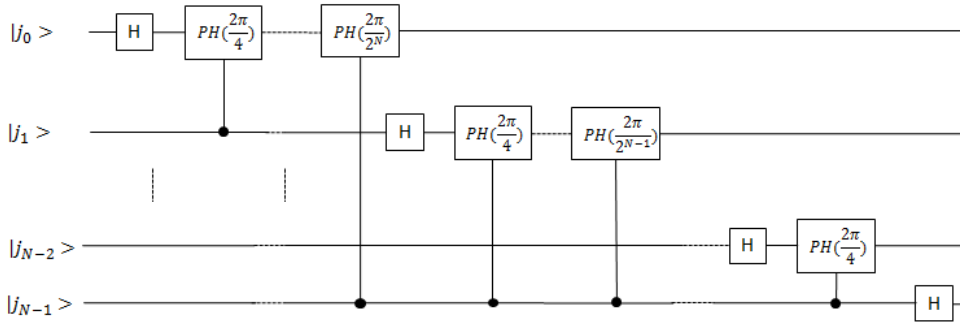
Final picture

Implementation in msb format



Final picture for implementation

cd QFT
vi QFT_exercise.py




```
def QFT(n):
    rout_qftn = QRoutine()
    for i in range(n):
        rout_qftn.apply(XXX, XXX)
        for j in range(XXX, XXX):
            angle = XXX
            rout_qftn.apply(PH(angle).ctrl(), XXX, XXX)
    return rout_qftn
```

Control operator applied on a gate. First qubits after is the control qubit, the other the target qubits

Complexity

- Counting number of gates: $\sum_{n=1}^{N} n = \frac{N(N+1)}{2} = O(N^2)$
- Last reversed ordering step : $\frac{N}{2}$ swaps gates, with 1 swap using 3 CNOT
- Then QFT on 2^N as a complexity $O(N^2)$
- Classical Fourier Transform using FFT : $O(N2^N)$
- So exponential speed-up ! But input/output problematic