Algorithme de Shor

Mathis Beaudoin, C2T3, Trois-Rivières

Été 2024

Table des matières

Avant-propros

Le présent document a été conçu lors de mon stage au C2T3 durant l'été 2024. Principalement, son but est de consolider mes apprentissages ainsi que de laisser une trace de mes travaux au C2T3. De plus, il permet à quiconque le lisant d'en apprendre davantage sur l'algorithme de Shor. Évidemment, je ne prétends aucunement être un expert de cet algorithme, car ce stage est la seule expérience que j'ai jusqu'à maintenant. De ce fait, ce texte ne présente qu'un sommaire de ma compréhension sur l'algorithme de Shor et peut ainsi contenir quelques erreurs.

Pour ce qui est de la structure du document, les sections 1 à 8 présentent les différents concepts nécessaires à la réalisation de l'algorithme de Shor. Puis, la section 9 expose la procédure en elle-même. Finalement, les références se trouvent complètement à la fin du document.

J'espère que ce document pourra être utile. Bonne lecture !

- Mathis Beaudoin, étudiant au baccalauréat en sciences de l'information quantique à l'UdS

1 Arithmétique modulaire

L'arithmétique modulaire est souvent décrite comme étant l'arithmétique des restes [1]. Dans cette arithmétique, on utilise les entiers avec le modulo comme opération principale. De manière générale, on écrit $x \bmod N$ pour désigner cette opération où $N \geq 1$. Si x est positif, le modulo consiste à soustraire N de x autant de fois que nécessaire pour atteindre un nombre entre 0 et $N-1$. Dans le cas où x est négatif, on suit la même procédure mais en additionnant au lieu de soustraire. De plus, l'opération modulo produit un résultat qui est cyclique. Par exemple, $1 \bmod 3 = 1$, $2 \bmod 3 = 2$, $3 \bmod 3 = 0$, $4 \bmod 3 = 1$, $5 \bmod 3 = 2$, $6 \bmod 3 = 0$, etc...

Par ailleurs, cette arithmétique fait appel au concept de congruence désignant une équivalence qu'on représente par le symbole $\equiv$. Par exemple, on sait que $7 \bmod 3 = 1$. De manière équivalente, on peut écrire $7 \equiv 1 \bmod 3$ parce que le reste de 7 vaut 1 quand il s'agit du modulo 3. On dit alors que « 7 est congru à 1 mod 3 ». En d'autres mots, $x \equiv y \bmod N \iff x = k \cdot N + y$ pour un certain entier k.

Il existe quelques identités reliées à l'arithmétique modulaire qu'on utilisera souvent, dont la dernière qu'on désignera par « exponentiation modulaire » :

$$(x + y) \bmod N = ((x \bmod N) + (y \bmod N)) \bmod N \tag{1}$$

$$(x \cdot y) \bmod N = ((x \bmod N) \cdot (y \bmod N)) \bmod N \tag{2}$$

$$(x^n) \bmod N = \underbrace{(x...(x(x \bmod N) \bmod N)...) \bmod N}_{\text{n fois}} \tag{3}$$

Pour le modulo, la notion d'inverse multiplicatif est plus subtile. À proprement parler, il n'existe pas d'opération inverse comme la division le serait pour la multiplication sur les nombres réels. En fait, pour un entier x quelconque, on s'attend pour la multiplication modulo N à ce que $x \cdot x^{-1} \equiv 1 \bmod N$. Attention, $x^{-1} \neq \frac{1}{x}$, car on travaille sur les entiers. Il se trouve que l'inverse x^{-1} existe si et seulement si x et N sont copremiers, c'est-à-dire si $\text{pgcd}(x, N) = 1$. De plus, on peut trouver cet inverse en temps polynomial s'il existe. La section C du document complémentaire explique plus en profondeur des concepts clés de l'arithmétique modulaire.

2 Transformée de Fourier discrète (DFT)

La DFT est l'analogue discret de la transformée de Fourier (voir section A du document complémentaire) [2]. La transformée de Fourier discrète transforme un vecteur $\vec{x} = (x_0, ..., x_{N-1})^T \in \mathbb{C}^N$ en un autre vecteur $\vec{y} = (y_0, ..., y_{N-1})^T \in \mathbb{C}^N$ avec coefficients

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{-\frac{2\pi i}{N} jk} \tag{4}$$

On étudie son impact sur la base canonique $\{\vec{e}_0, ..., \vec{e}_{N-1}\}$, c'est-à-dire $\{(1, 0, ..., 0)^T, (0, 1, 0, ..., 0)^T, ..., (0, ..., 0, 1)^T\}$. Par (4), on sait que l'application de la DFT sur $\vec{e}_j$ donne des coefficients $y_k = \frac{1}{\sqrt{N}} e^{-\frac{2\pi i}{N} jk}$, car $\vec{e}_j$ n'a qu'un élément non nul. Ainsi,

$$DFT(\vec{e}_j) = \frac{1}{\sqrt{N}} (1, e^{-\frac{2\pi i}{N} j}, ..., e^{-\frac{2\pi i}{N} j(N-1)})^T = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{-\frac{2\pi i}{N} jk} \vec{e}_k = \vec{u}_j$$

Donc, la DFT transforme $\{\vec{e}_j\}$ en un autre ensemble de vecteurs $\{\vec{u}_j\}$. On s'assure que $\{\vec{u}_j\}$ forme une base orthonormée.

$$\vec{u}_a \cdot \vec{u}_b = \left(\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{-\frac{2\pi i}{N} ak} \vec{e}_k \right)^\dagger \cdot \left(\frac{1}{\sqrt{N}} \sum_{k'=0}^{N-1} e^{-\frac{2\pi i}{N} bk'} \vec{e}_{k'} \right) = \frac{1}{N} \sum_{k=0}^{N-1} \sum_{k'=0}^{N-1} e^{-\frac{2\pi i}{N}(k'b-ka)} \vec{e}_k \cdot \vec{e}_{k'}$$

On distingue alors deux cas.

$$a = b : \ \vec{u}_a \cdot \vec{u}_b = \frac{1}{N} \sum_{k=0}^{N-1} \sum_{k'=0}^{N-1} e^{-\frac{2\pi i}{N} b(k'-k)} \vec{e}_k \cdot \vec{e}_{k'} = \frac{1}{N} \sum_{k=0}^{N-1} e^{-\frac{2\pi i}{N} b(k-k)} = \frac{1}{N} \sum_{k=0}^{N-1} 1 = 1$$

$$a \neq b : \ \vec{u}_a \cdot \vec{u}_b = \frac{1}{N} \sum_{k=0}^{N-1} \sum_{k'=0}^{N-1} e^{-\frac{2\pi i}{N}(k'b-ka)} \vec{e}_k \cdot \vec{e}_{k'} = \frac{1}{N} \sum_{k=0}^{N-1} e^{-\frac{2\pi i}{N} k(b-a)} = \frac{1}{N} \sum_{k=0}^{N-1} \omega^{k(b-a)} = \frac{1}{N} \frac{1-\omega^{N(b-a)}}{1-\omega^{(b-a)}} = 0$$

Ainsi, $\{\vec{u}_j\}$ forme une base orthonormée. Donc, la DFT effectue un changement de base et on peut écrire la matrice de changement de base U.

$$U = (\vec{u}_0, ..., \vec{u}_{N-1}) \ \text{ où } \ U_{jk} = \frac{1}{\sqrt{N}} \, e^{-\frac{2\pi i}{N} jk} = \frac{1}{\sqrt{N}} \, \omega^{jk} \ \text{ où } \ \omega = e^{-\frac{2\pi i}{N}}$$

On remarque que U est symétrique par définition. De surcroît, on peut facilement montrer que U est unitaire grâce au fait que $\{\vec{u}_j\}$ forme une base orthonormée.

$$U^\dagger U = [\vec{u}_j \cdot \vec{u}_k]_{j,k=0}^{N-1} = \mathbb{I}$$

On dit que U correspond à la DFT et que $U^\dagger$ correspond à la DFT inverse.

3 Transformée de Fourier quantique (QFT)

La QFT est l'équivalent quantique de la DFT inverse à cause de certaines conventions de signes [3]. Ainsi, on travaille avec des bras/kets dans un espace de Hilbert de dimension $N = 2^n$ où n est le nombre de qubits. On regarde son effet sur un vecteur de base $|j\rangle$ en gardant en tête que $k = \sum_{l=0}^{n-1} k_l 2^l$ est la représentation binaire de k où $k_l \in \{0, 1\}$.

$$\text{QFT} \, |j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi i}{N} jk} \, |k\rangle = \frac{1}{\sqrt{N}} \sum_{k_{n-1}=0}^{1} ... \sum_{k_0=0}^{1} e^{2\pi i j \sum_{l=0}^{n-1} k_l \frac{2^l}{2^n}} \, |k_{n-1}...k_0\rangle$$

$$= \frac{1}{\sqrt{N}} \sum_{k_{n-1}=0}^{1} ... \sum_{k_0=0}^{1} e^{2\pi i j \sum_{l=0}^{n-1} \frac{k_l}{2^{n-l}}} \, |k_{n-1}...k_0\rangle = \frac{1}{\sqrt{N}} \sum_{k_{n-1}=0}^{1} ... \sum_{k_0=0}^{1} \prod_{l=0}^{n-1} e^{2\pi i j \frac{k_l}{2^{n-l}}} \, |k_{n-1}\rangle \otimes ... \otimes |k_0\rangle$$

Par convention, on voudrait plutôt que l'ordre du produit soit inversé, ce qu'on fait en changeant 2^{n-l} par 2^{l+1}. Ainsi,

$$\frac{1}{\sqrt{N}} \sum_{k_{n-1}=0}^{1} ... \sum_{k_0=0}^{1} \prod_{l=0}^{n-1} e^{2\pi i j \frac{k_l}{2^{l+1}}} |k_{n-1}\rangle \otimes ... \otimes |k_0\rangle = \frac{1}{\sqrt{N}} \left(\sum_{k_{n-1}=0}^{1} e^{2\pi i j \frac{k_{n-1}}{2^n}} |k_{n-1}\rangle \otimes ... \otimes \sum_{k_0=0}^{1} e^{2\pi i j \frac{k_0}{2}} |k_0\rangle \right)$$

$$= \frac{1}{\sqrt{N}} \left[\Big(|0\rangle + e^{2\pi i \frac{j}{2^n}} |1\rangle \Big) \otimes ... \otimes \Big(|0\rangle + e^{2\pi i \frac{j}{2}} |1\rangle \Big) \right] = \frac{1}{\sqrt{N}} \bigotimes_{l=0}^{n-1} \Big(|0\rangle + e^{2\pi i \frac{j}{2^{l+1}}} |1\rangle \Big)$$

Comme j est un entier qu'on pourrait représenter en binaire par $j = j_{n-1}...j_0$, alors $\frac{j}{2^{l+1}}$ décale les bits de j vers la droite par $l+1$ positions. On aurait donc $\frac{j}{2^{l+1}} = j_{n-1}...j_{l+1}.j_l...j_0$ où le point sépare la partie entière (à gauche) de la partie fractionnaire (à droite). Par contre, puisque cette quantité se trouve dans une exponentielle complexe, la partie entière n'a aucun impact sur le calcul. De ce fait, on peut simplement garder la partie fractionnaire $0.j_l...j_0$ dans l'exponentielle. La notation $0.j_l...j_0$ équivaut à $j_l 2^{-1} + ... + j_0 2^{-(l+1)}$ et correspond à l'écriture en binaire d'un nombre décimal. On a alors la forme finale de la QFT :

$$|\phi(j)\rangle = \text{QFT}\,|j\rangle = \frac{1}{\sqrt{N}} \bigotimes_{l=0}^{n-1} \big(|0\rangle + e^{2\pi i \ 0.j_l...j_0} |1\rangle \big) \tag{5}$$

où chaque qubit $|j_l\rangle$ de l'état de base $|j\rangle$ devient

$$|\phi_l(j)\rangle = \frac{1}{\sqrt{2}} \big(|0\rangle + e^{2\pi i \ 0.j_l...j_0} |1\rangle \big)$$

On cherche maintenant à construire le circuit permettant d'effectuer la QFT sur un état de base $|j\rangle$, c'est-à-dire de trouver une implémentation équivalente à (5). On utilisera la porte Hadamard H et la porte de phase R_k.

$$R_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{bmatrix}$$

L'application de H sur $|j_{n-1}\rangle$ entraîne l'ajout de j_{n-1} au début de l'exponentielle.

$$H\,|j_{n-1}\rangle\ |j_{n-2}...j_0\rangle = \frac{1}{\sqrt{2}} \big(|0\rangle + (-1)^{j_{n-1}} |1\rangle \big)\ |j_{n-2}...j_0\rangle = \frac{1}{\sqrt{2}} \big(|0\rangle + e^{2\pi i \ 0.j_{n-1}} |1\rangle \big)\ |j_{n-2}...j_0\rangle$$

Puis, en appliquant R_2 contrôlée par $|j_{n-2}\rangle$ sur le qubit de poids fort, on peut ajouter ce dernier à l'exponentielle.

$$CR_2 \left(\frac{1}{\sqrt{2}} \big(|0 j_{n-2}\rangle + e^{2\pi i \ 0.j_{n-1}} |1 j_{n-2}\rangle \big) \right)\ |j_{n-3}...j_0\rangle = \frac{1}{\sqrt{2}} \big(|0\rangle + e^{2\pi i \ 0.j_{n-1}j_{n-2}} |1\rangle \big)\ |j_{n-2}...j_0\rangle$$

En continuant ainsi de suite pour $CR_3, ..., CR_n$, on se retrouve avec

$$\frac{1}{\sqrt{2}} \big(|0\rangle + e^{2\pi i \ 0.j_{n-1}...j_0} |1\rangle \big)\ |j_{n-2}...j_0\rangle$$

ce qui correspond bien à l'état du dernier qubit de (5). Pour appliquer la QFT au complet, il suffit de suivre ce même principe pour chaque qubit de l'état de base $|j\rangle$ afin d'obtenir globalement l'état correspondant à (5). Depuis ce circuit, il est par la suite facile d'obtenir $\text{QFT}^\dagger$ qu'on nommera la QFT inverse.

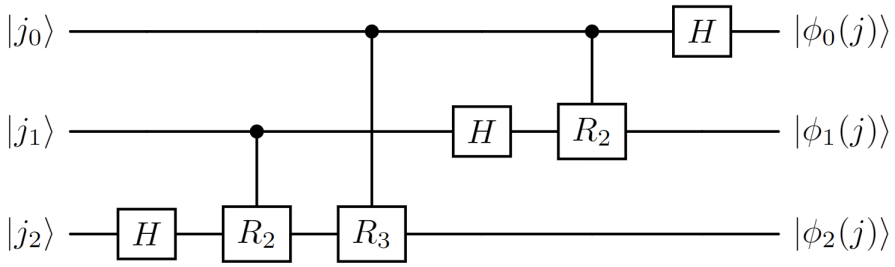

FIGURE 1 – Circuit de la QFT pour un état de base $|j\rangle$ à 3 qubits

La QFT nécessite n portes Hadamard et $\sum_{j=1}^{n}(n-j)$ portes de phase $\implies \mathcal{O}(n^2)$ où n est le nombre de qubits. La FFT (analogue classique de la QFT) a une complexité $\mathcal{O}(Nlog(N)) = \mathcal{O}(2^n n)$. Il y a donc un avantage exponentielle à utiliser la QFT plutôt que la FFT.

4 Estimation de phase quantique (QPE)

Soient une matrice unitaire quelconque U et un de ses vecteurs propres $|\psi\rangle$. Puisque la matrice est unitaire, on sait que la valeur propre associée à $|\psi\rangle$ se situe sur le cercle du plan complexe, c'est-à-dire que $U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle$ avec $\theta \in [0,1)$. U est une boîte noire qu'on peut utiliser mais dont on ne connaît rien, il n'est donc pas possible d'utiliser sa représentation matricielle pour trouver ses valeurs propres. L'estimation de phase quantique permet de donner une estimation de la valeur de θ (donc de la phase, d'où le nom) avec un certain degré de précision [3]. En effet, on peut spécifier le nombre de qubits sur lesquels on stocke l'estimation de θ. Donc, si θ s'écrit avec t bits, alors une QPE avec un degré de précision à $m \geq t$ qubits donnera la réponse exacte. Au contraire, une QPE avec $m < t$ qubits de précision donnera seulement une estimation à m qubits de θ.

Afin de réaliser la QPE, il faudra deux registres. Le premier registre contiendra m qubits tous à $|0\rangle$. C'est sur ces qubits qu'on stockera l'approximation de θ. Le deuxième registre contient lui le vecteur propre $|\psi\rangle$ dont on cherche la valeur propre. Donc, on suppose qu'on est capable de préparer $|\psi\rangle$ au préalable de manière efficace. La première étape de la QPE consiste à appliquer une porte Hadamard sur chacun des qubits du premier registre afin de créer une superposition uniforme. Puis, on applique une série de portes U^{2^k} pour $k \in [0, ..., m-1]$ chacune contrôlée par un qubit différent du premier registre. Son action permet d'ajouter une phase relative $e^{2\pi i 2^k \theta}$.

$$CU2^k\left(\frac{1}{\sqrt{2}}\left(|0\psi\rangle + |1\psi\rangle\right)\right) = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i 2^k\theta}|1\rangle\right)|\psi\rangle$$

Comme θ est compris dans l'intervalle $[0,1)$ et qu'on a m qubits de stockage, on peut utiliser la représentation binaire décimale pour dire que $\theta = 0.\theta_{m-1}...\theta_0 = 0 + \theta_{m-1}2^{-1} + ... + \theta_0 2^{-m}$. Ainsi, $2^k\theta$ décale les bits constituant θ de k positions vers la gauche. Par exemple,

$$2^0\theta = 0.\theta_{m-1}...\theta_0,\ 2^1\theta = \theta_{m-1}.\theta_{m-2}...\theta_0,\ ...,\ 2^{m-1}\theta = \theta_{m-1}...\theta_1.\theta_0$$

La partie entière de $2^k\theta$, comme on le sait, n'a aucun impact sur l'exponentielle $e^{2\pi i 2^k\theta}$. De ce fait, on ne garde que la partie fractionnaire.

$$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i 2^k \theta}|1\rangle\right)|\psi\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i\ 0.\theta_{m-1-k}\ldots\theta_0}|1\rangle\right)|\psi\rangle$$

Si on applique une porte CU^{2^0} contrôlée par le dernier qubit du premier registre, une porte CU^{2^1} contrôlée par l'avant-dernier qubit du premier registre et ainsi de suite jusqu'à une porte $CU^{2^{m-1}}$ contrôlée par le premier qubit du premier registre, alors on se retrouve avec un état particulier sur le premier registre.

$$\left[\frac{1}{\sqrt{2^m}}\bigotimes_{k=0}^{m-1}\left(|0\rangle + e^{2\pi i\ 0.\theta_k\ldots\theta_0}|1\rangle\right)\right]|\psi\rangle = \frac{1}{\sqrt{2^m}}\left[\left(|0\rangle + e^{2\pi i\ 0.\theta_{m-1}\ldots\theta_0}|1\rangle\right)\otimes\ldots\otimes\left(|0\rangle + e^{2\pi i\ 0.\theta_0}|1\rangle\right)\right]|\psi\rangle$$

Il s'agit de l'état $|\theta\rangle$ auquel on aurait appliqué la QFT. Comme la valeur de θ se trouve dans cet état, on applique ensuite la QFT inverse pour l'obtenir. Finalement, on mesure les qubits du premier registre afin d'avoir le résultat de l'approximation de θ, ce qui permet d'approximer la valeur propre du vecteur propre $|\psi\rangle$ de U. Puisque θ est une valeur décimale par définition, il faut faire attention à convertir sa valeur binaire en sa valeur numérique selon la représentation binaire décimale. Comme le vecteur propre dans le second registre reste inchangé, on peut relancer la QPE plusieurs fois en choisissant comme phase le résultat le plus fréquent dans le but d'avoir une meilleure approximation.

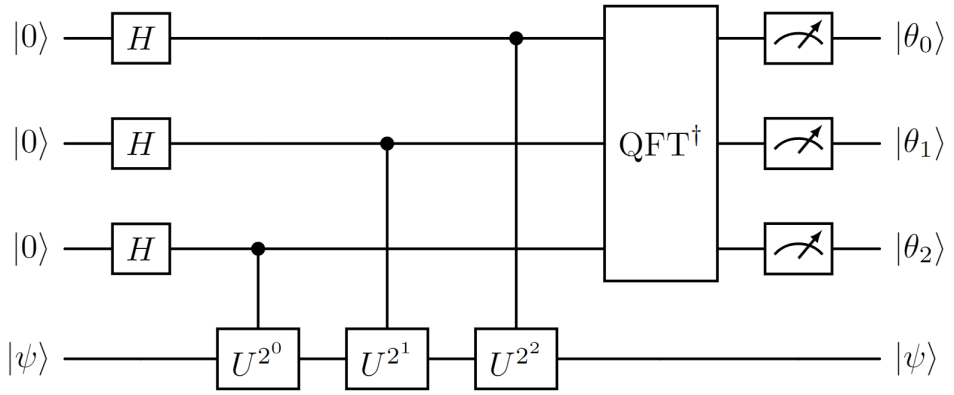

FIGURE 2 – Circuit de la QPE pour $m = 3$

Pour utiliser moins de qubits dans le premier registre, on peut se servir du *one qubit trick* [4]. Au lieu d'avoir m qubits qui stockent l'approximation ou la valeur exacte de θ, on peut en prendre un seul afin de connaître bit par bit la valeur de la phase. On commence par développer le circuit de la QFT† à la figure 2.

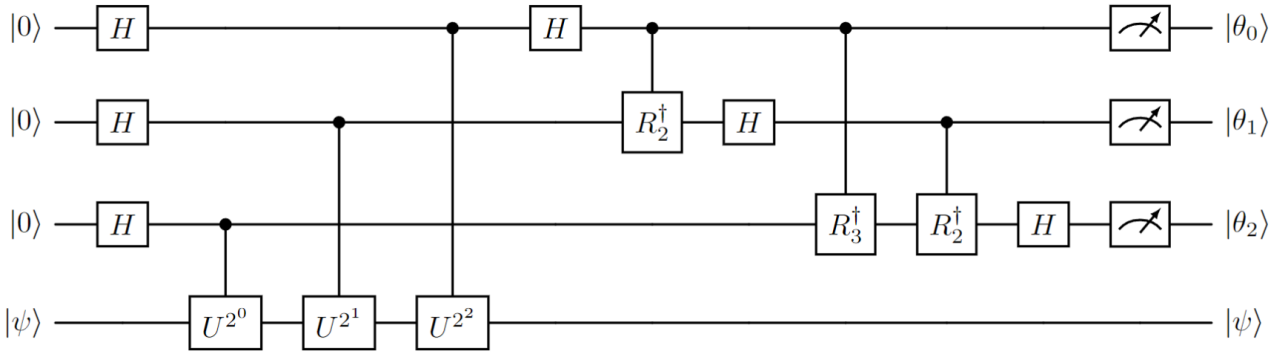

FIGURE 3 – Développement du circuit de la QPE pour $m = 3$

On s'attarde ensuite au premier qubit (celui tout en haut de la figure 3). Après la deuxième porte Hadamard, ce qubit sert de qubit de contrôle pour des portes de phase qui ne l'affectent aucunement. Puis, on mesure ce qubit. Si l'état après la mesure est $|0\rangle$, puisque les portes contrôlées de phase ne changent pas ce qubit, cela veut dire que le qubit était aussi dans l'état $|0\rangle$ avant leur application sur les autres qubits. La même logique s'applique si on avait eu l'état $|1\rangle$. De ce fait, on pourrait mesurer le premier qubit avant les portes de phase puis les ajouter après coup au circuit selon la mesure. De surcroît, en poussant plus loin la précédente réflexion, on voit que toutes les opérations sur le premier qubit (du tout début du circuit jusqu'à la mesure) peuvent être effectuées avant même celles des autres qubits. En fait, si on applique cette logique sur tout le premier registre, on voit qu'il est possible d'effectuer les opérations de chaque qubit successivement en appliquant manuellement les portes de phase selon la valeur des mesures passées. Grâce à cela, on peut utiliser un seul qubit afin de faire ce qu'on ferait normalement avec m qubits sur le premier registre. C'est ce qu'on appelle le *one qubit trick* et il permet, au détriment de la profondeur, de réduire le nombre de qubits requis pour estimer la phase.

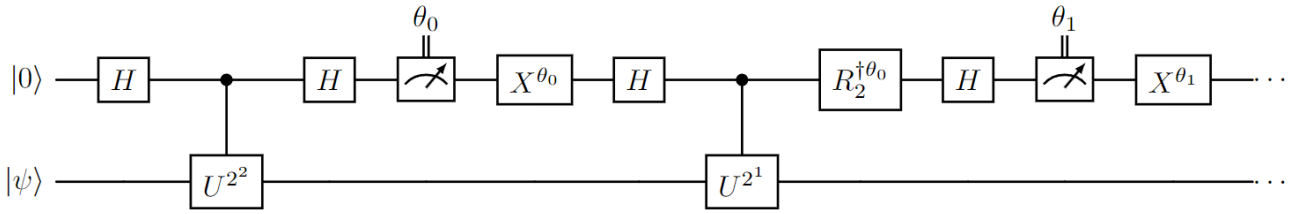

FIGURE 4 – Le *one qubit trick* pour une partie du circuit de la figure 3

5 Problème de la recherche d'ordre

La QPE peut être utilisée pour résoudre plusieurs problèmes dont le problème de la recherche d'ordre [3]. Il consiste à trouver pour $a, N \in \mathbb{N}^*$ le plus petit entier $r \leq N$ (qu'on appelle l'ordre) tel que $a^r \equiv 1 \bmod N$ où $a < N$ et $\text{pgcd}(a, N) = 1$. Pour résoudre ce problème avec la QPE, on emploie l'unitaire suivante :

$$U_a |x\rangle = |ax \bmod N\rangle \tag{6}$$

On ajoute aussi que U_a agit seulement lorsque $0 \leq x < N$. Autrement, on considère qu'il s'agit de la matrice identité. On remarque par (6) que $U_a^r |x\rangle = |a^r x \bmod N\rangle = |(a^r \bmod N)(x \bmod N) \bmod N\rangle = |(1 \cdot x) \bmod N\rangle = |x\rangle$, c'est-à-dire que $U_a^r = \mathbb{I}$. Ainsi, les valeurs propres λ_s des vecteurs propres $|u_s\rangle$ de U_a sont de la forme $e^{2\pi i \frac{s}{r}}$ où s est un certain entier, puisqu'il faut que $\lambda_s^r = 1$. De plus, on montre que les vecteurs propres $|u_s\rangle$ de U_a sont $|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi i k \frac{s}{r}} |a^k \bmod N\rangle$.

$$U_a |u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi i k \frac{s}{r}} |a(a^k \bmod N) \bmod N\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi i k \frac{s}{r}} |(a \bmod N)(a^k \bmod N) \bmod N\rangle$$

$$= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi i k \frac{s}{r}} |a^{k+1} \bmod N\rangle = \frac{1}{\sqrt{r}} \sum_{k'=1}^{r} e^{-2\pi i (k'-1) \frac{s}{r}} |a^{k'} \bmod N\rangle = e^{2\pi i \frac{s}{r}} \frac{1}{\sqrt{r}} \sum_{k'=1}^{r} e^{-2\pi i k' \frac{s}{r}} |a^{k'} \bmod N\rangle$$

$$= e^{2\pi i \frac{s}{r}} \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi i k \frac{s}{r}} |a^k \bmod N\rangle = e^{2\pi i \frac{s}{r}} |u_s\rangle$$

Trouver l'ordre semble alors facile, car on aurait qu'à se servir de la QPE pour approximer $\frac{s}{r}$ et de là en tirer r. Mais avant de pouvoir utiliser la QPE, il faut s'assurer que U_a soit unitaire et qu'un vecteur propre $|u_s\rangle$ (ou qu'une superposition de ces vecteurs propres) peut facilement être préparé.

On vérifie d'abord que U_a est inversible. L'application de U_a sur un état quelconque $|x\rangle$ produit $|ax \bmod N\rangle$. On s'attend donc à ce que U_a^{-1} effectue l'inverse multiplicatif modulo N dont on a parlé à la section 1. Puisqu'on a comme condition initiale que $\text{pgcd}(a, N) = 1$, on sait par la section 1 que a^{-1} existe et donc que U_a est inversible.

$$U_a^{-1}\,|ax \bmod N\rangle = |a^{-1}(ax \bmod N) \bmod N\rangle = |(a^{-1} \bmod N)(ax \bmod N) \bmod N\rangle = |(a^{-1}ax) \bmod N\rangle = |x\rangle$$

Il reste maintenant à vérifier que $U_a^{-1} = U_a^\dagger$, ce qui est équivalent à montrer que $U_a^\dagger U_a = \mathbb{I}$. On calcule

$$\langle x'|\,U_a^\dagger U_a\,|x\rangle = \langle ax' \bmod N|ax \bmod N\rangle = \delta_{x,x'} \implies U_a^\dagger U_a = \mathbb{I} \implies U_a^\dagger = U_a^{-1}$$

et on voit bien que U_a est unitaire.

À priori, préparer un état propre $|u_s\rangle$ n'est pas évident, car selon sa définition, il faudrait connaître l'ordre r. Cependant, c'est exactement cette quantité qu'on cherche. Toutefois, on remarque que

$$\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|u_s\rangle = \frac{1}{\sqrt{r}}\sum_{k,s=0}^{r-1} e^{-2\pi i k\frac{s}{r}}\,|a^k \bmod N\rangle = \sum_{k=0}^{r-1}\left(\frac{1}{r}\sum_{s=0}^{r-1}e^{-2\pi i k\frac{s}{r}}\right)|a^k \bmod N\rangle$$

Puis, on utilise la série géométrique

$$\sum_{s=0}^{r-1}e^{-2\pi i k\frac{s}{r}} = \sum_{s=0}^{r-1}\left(e^{-2\pi i\frac{k}{r}}\right)^s = \frac{1-e^{-2\pi i k}}{1-e^{-2\pi i\frac{k}{r}}} = \delta_{k,0}$$

pour en arriver à

$$\sum_{k=0}^{r-1}\delta_{k,0}\,|a^k \bmod N\rangle = |1\rangle$$

Ainsi, l'état $|1\rangle$ correspond à une certaine superposition de vecteurs propres de U_a et permet une initialisation simple pour la QPE. Comme toutes les conditions sont remplies, la QPE sera capable d'obtenir une des valeurs propres $\frac{s}{r}$ d'un des états dans la superposition. À partir de là, on peut retrouver r grâce aux fractions continues (section D du document complémentaire). Malgré tout, on ne sait absolument pas comment faire pour implémenter la transformation unitaire avec un circuit quantique. Les prochaines sections montreront les différents blocs nécessaires à sa construction.

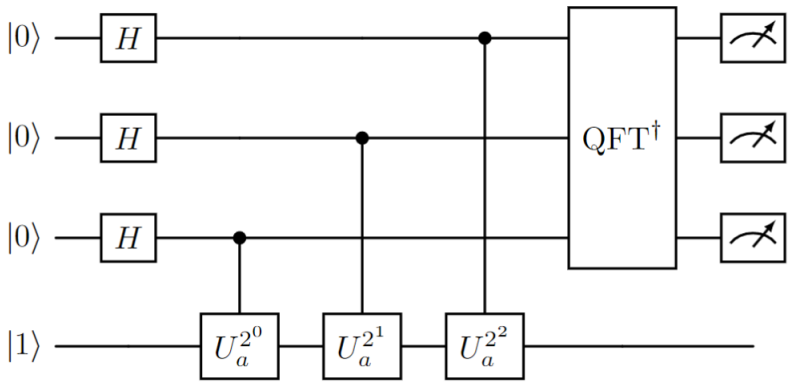

FIGURE 5 – Circuit du problème de la recherche d'ordre avec $m = 3$

6 Additionneurs quantiques de Draper

Thomas G. Draper a proposé une méthode utilisant la QFT afin d'implémenter un additionneur quantique capable d'additionner la valeur binaire d'un nombre classique à celle d'un registre quantique [5]. D'ailleurs, on peut l'adapter pour additionner la valeur binaire de deux registres quantiques [5]. On présente les détails mathématiques ainsi que leur implémentation avec un circuit quantique.

6.1 Additionneur ΦADD$_a$

On montre dans cette section comment implémenter un additionneur quantique qui ajoute une valeur classique à un registre quantique. Soient a et b deux entiers binaires non-signés sur n bits. On commence par encoder dans un registre quantique de n qubits la valeur de b pour obtenir $|b\rangle$. Puis, on lui applique la QFT. Ensuite, sur chaque qubit $|\phi_j(b)\rangle$, on applique une porte

$$A_{j+1} = \prod_{k=1}^{j+1} R_k^{a_{j+1-k}} = \begin{bmatrix} 1 & 0 \\ 0 & \prod_{k=1}^{j+1} e^{2\pi i a_{j+1-k}/2^k} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i\left(\frac{a_j}{2}+\frac{a_{j-1}}{4}+\ldots+\frac{a_0}{2^{j+1}}\right)} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i a/2^{j+1}} \end{bmatrix} \tag{7}$$

de telle sorte à obtenir

$$A_{j+1}\,|\phi_j(b)\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i \frac{b}{2^{j+1}}} e^{2\pi i \frac{a}{2^{j+1}}}\,|1\rangle\right) = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i \frac{(a+b)}{2^{j+1}}}\,|1\rangle\right) = |\phi_j(a+b)\rangle$$

Ainsi, après l'application des A_{j+1} sur tous les qubits $|\phi_j(b)\rangle$, on se retrouve globalement avec $|\phi(a+b)\rangle$. On peut finalement appliquer la QFT inverse pour obtenir le résultat de l'addition. Il faut cependant faire attention à un potentiel débordement, car la somme de deux entiers à n bits peut résulter en un entier à $n+1$ bits. Pour empêcher cela, on peut mettre b dans un registre à $n+1$ qubits et mettre implicitement a sur $n+1$ bits aussi (leur bit de poids fort étant forcément à 0) pour ensuite faire la même procédure. Au final,

$$\Phi\text{ADD}_a\,|\phi(b)\rangle = |\phi(a+b)\rangle \tag{8}$$

qu'on peut aisément implémenter avec un circuit quantique.

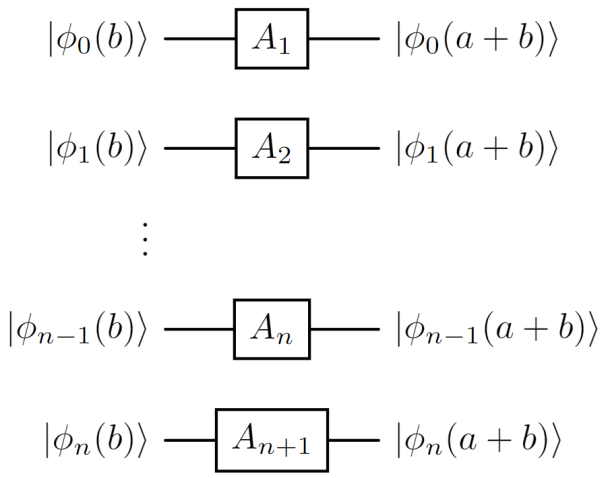

FIGURE 6 – Circuit pour ΦADD$_a$ avec la gestion du débordement

6.2 Additionneur QΦADD

On peut facilement adapter ΦADD_a afin d'additionner deux entiers binaires non-signés a et b sur n bits contenus chacun dans un registre quantique. Dans cette version, le résultat de l'addition se trouvera dans le registre contenant initialement b. On peut s'inspirer de la section précédente pour reproduire les A_{j+1} en utilisant des opérations contrôlées par les qubits du registre contenant a. Effectivement, par (7), on voit que les portes R_k sont appliquées ou non selon la valeur de a_{j+1-k}, puisque $R_k^1 = R_k$ et $R_k^0 = I$. Là aussi, on peut inclure la gestion du débordement avec un registre de $n+1$ qubits pour b. Dans ce cas, a n'a pas aussi besoin d'être dans un registre de $n+1$ qubits, car comme a est un entier sur n bits, forcément $|a_n\rangle = 0$ et il ne permettra d'appliquer aucune porte. En fait, on réalise le circuit comme si on avait deux registres à $n+1$ qubits, mais on omet $|a_n\rangle$ et ce qu'il contrôle. En résumé, on a

$$Q\Phi\text{ADD}(\,|a\rangle\;|\phi(b)\rangle) = \;|a\rangle\;|\phi(a+b)\rangle \tag{9}$$

qu'on réalise facilement avec des portes R_k contrôlées.

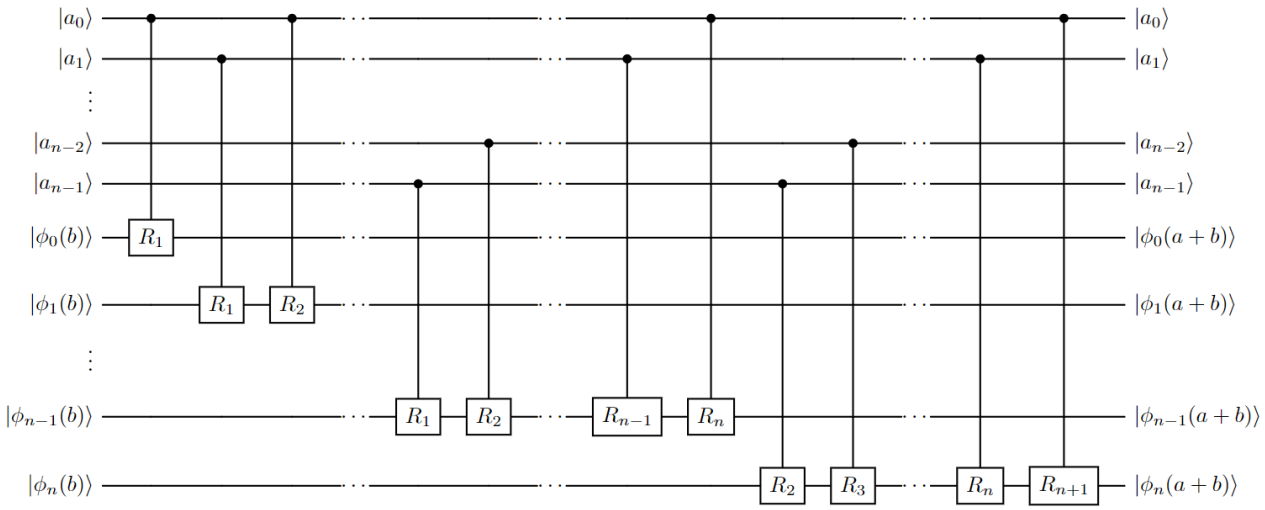

FIGURE 7 – Circuit pour QΦADD avec la gestion du débordement

6.3 $\Phi\text{ADD}_a{}^\dagger$ et $Q\Phi\text{ADD}^\dagger$

Comme on s'en doute, $\Phi\text{ADD}_a{}^\dagger$ et $Q\Phi\text{ADD}^\dagger$ correspondent à des circuits qui effectuent une soustraction.

$$A_{j+1}^\dagger\,|\phi_j(b)\rangle = \frac{1}{\sqrt{2}}\left(\,|0\rangle + e^{2\pi i\frac{b}{2^{j+1}}}e^{2\pi i\frac{-a}{2^{j+1}}}\,|1\rangle\right) = \frac{1}{\sqrt{2}}\left(\,|0\rangle + e^{2\pi i\frac{(b-a)}{2^{j+1}}}\,|1\rangle\right) = \;|\phi_j(b-a)\rangle$$

Cependant, a et b sont des entiers binaires non-signés et l'équation n'a du sens que si $b-a \geq 0$. Que se passe-t-il si $b-a<0$?

Prenons comme exemple les valeurs $b = 1010_2$ et $a = 1100_2$ sur 4 bits. Alors, $b-a = -0010_2$, ce qui n'a pas de sens pour les nombres binaires non-signés. Dans ce cas, on veut trouver un entier positif équivalent à $b-a$. Pour tous les entiers, il est trivial de dire que $(b-a)+a = b$. Donc, $(b-a)+1100_2 = 1010_2$. Certes, la relation fonctionne avec -0010_2, mais aussi avec 1110_2 qui est un entier positif. 1110_2 fonctionne, car on travaille sur 4 bits. Normalement, $1110_2 + 1100_2 = 11010_2$, mais on ne garde que les 4 premiers bits. Cela ramène le résultat à $11010_2 \bmod 2^4 = 1010_2$ et la relation triviale tient. De ce fait, $-0010_2 \equiv 1110_2$.

De manière générale, en écrivant a et b sur $n+1$ bits pour éviter un débordement, on trouve que $b-a = ((b-a)+2^{n+1}) \bmod 2^{n+1}$ [6]. Si $b \geq a$, alors forcément $((b-a)+2^{n+1}) \bmod 2^{n+1} = b-a$. Sinon, $(b-a)+2^{n+1}$ est forcément entre 0 et 2^{n+1}, ce qui fait que $((b-a)+2^{n+1}) \bmod 2^{n+1} = (b-a)+2^{n+1}$.

Ainsi,

$$\Phi\text{ADD}_a^\dagger \left|\phi(b)\right\rangle = \begin{cases} \left|\phi(b-a)\right\rangle, & \text{si } \ b \geq a \\ \left|\phi((b-a)+2^{n+1})\right\rangle, & \text{sinon} \end{cases} \tag{10}$$

$$Q\Phi\text{ADD}^\dagger(\left|a\right\rangle \left|\phi(b)\right\rangle) = \begin{cases} \left|a\right\rangle \left|\phi(b-a)\right\rangle, & \text{si } \ b \geq a \\ \left|a\right\rangle \left|\phi((b-a)+2^{n+1})\right\rangle, & \text{sinon} \end{cases} \tag{11}$$

7 Multiplieur modulaire quantique de Pavlidis

Pavlidis a proposé une manière de réaliser un multiplieur modulaire quantique, c'est-à-dire un circuit quantique qui calcule le résultat de $ax \bmod N$ [7]. Cependant, vu la demande en ressources que nécessite ce multiplieur modulaire autant classiquement que sur un ordinateur quantique, il s'avère complètement inutile pour l'utilisation qu'on veut en faire. La personne intéressée peut se référer à l'article pour en apprendre davantage, car cette méthode ne sera pas expliquée ici.

8 Multiplieur modulaire quantique de Beauregard

Stéphane Beauregard propose aussi une manière de réaliser un multiplieur modulaire quantique qui utilise beaucoup moins de qubits que l'approche de Pavlidis [6]. Cela rend la méthode intéressante, car les ressources étant moindres, on va pouvoir exécuter le circuit qui en découle et faire des tests. On présente quelques sous-routines qui, lorsque combinées, permettront de réaliser le multiplieur modulaire.

8.1 CΦADD$_a$ et CCΦADD$_a$

Tout d'abord, il nous faut des versions contrôlées de ΦADD$_a$. Construire ces portes n'est pas compliqué, car il suffit d'ajouter un ou deux qubits qui contrôlent l'application de chaque A_{j+1} de la figure 6.

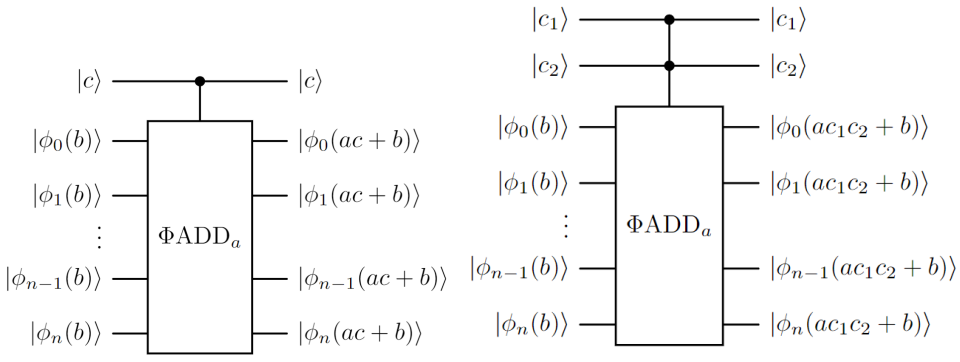

FIGURE 8 – La porte CΦADD$_a$ (à gauche) et la porte CCΦADD$_a$ (à droite)

8.2 Additionneur modulaire CCΦADD$_a$MOD$_N$

L'additionneur modulaire prend des entiers positifs a, b et N sur n bits où $a, b < N$. On utilisera ces quantités sur $n+1$ bits encore une fois pour gérer le débordement. L'additionneur modulaire permet de calculer $(a+b) \bmod N$, c'est-à-dire de calculer $a+b$ et d'enlever N si $a+b \geq N$.

Comme préparation, on stocke b dans un registre quantique à $n+1$ qubits et on lui applique la QFT pour avoir $|\phi(b)\rangle$. En plus d'un qubit ancillaire $|0\rangle$, il y a aussi deux qubits de contrôle $|c_1\rangle$ et $|c_2\rangle$. On s'attend donc à ce que CCΦADD$_a$MOD$_N(|c_1\rangle\,|c_2\rangle\,|\phi(b)\rangle\,|0\rangle) = |c_1\rangle\,|c_2\rangle\,|\phi((a \cdot c_1c_2 + b) \bmod N)\rangle\,|0\rangle$. On présente d'abord le circuit correspondant à CCΦADD$_a$MOD$_N$, puis les détails mathématiques.

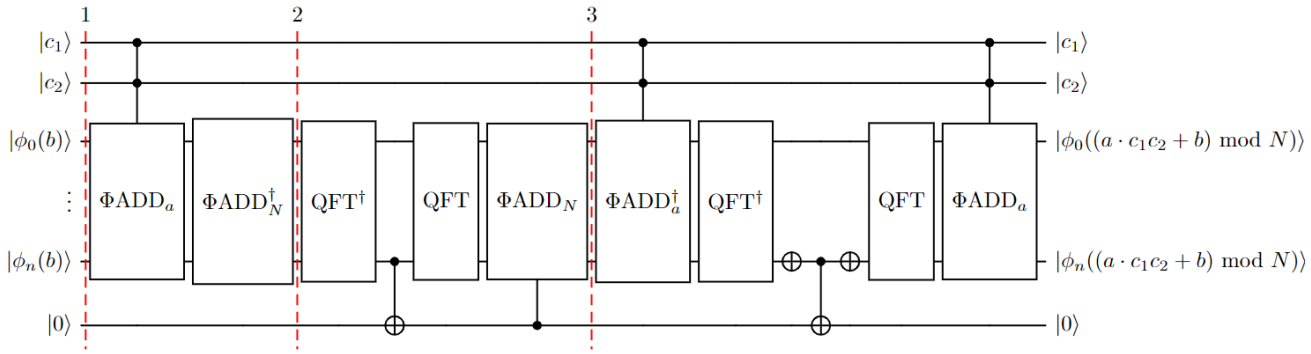

FIGURE 9 – Circuit pour CCΦADD$_a$MOD$_N$

Partie 1

On commence par appliquer CCΦADD$_a$ sur $|\phi(b)\rangle$ pour avoir $|\phi(a \cdot c_1c_2 + b)\rangle$. Puis, on utilise ΦADD$^\dagger_N$ pour avoir $|\phi(a \cdot c_1c_2 + b - N)\rangle$. On distingue alors deux cas. D'abord, il se peut que $a \cdot c_1c_2 + b - N < 0$, ce qui veut dire qu'on n'avait pas à soustraitre N pour calculer le modulo. Puis, il est possible que $a \cdot c_1c_2 + b - N \geq 0$, ce qui indique que la soustraction par N était nécessaire pour faire le modulo.

Partie 2

Pour savoir si on doit ou non rajouter N pour calculer correctement le modulo, il faut avoir accès à la valeur du qubit de poids fort de $|a \cdot c_1c_2 + b - N\rangle$. Effectivement, selon la section 6.3, il vaut 0 si $a \cdot c_1c_2 + b - N \geq 0$ et 1 sinon. Pour y arriver, il faut appliquer la QFT inverse sur $|\phi(a \cdot c_1c_2 + b - N)\rangle$, puis copier la valeur du qubit de poids fort dans le qubit ancillaire grâce à une porte CNOT. On réapplique ensuite la QFT pour revenir à $|\phi(a \cdot c_1c_2 + b - N)\rangle$. Si le qubit ancillaire vaut $|0\rangle$, le modulo a été calculé correctement et il ne faut pas additionner N. Cependant, si le qubit ancillaire est dans l'état $|1\rangle$, il faut rajouter N. Donc, il suffit d'utiliser CΦADD$_N$ sur $|\phi(a \cdot c_1c_2 + b - N)\rangle$ qu'on contrôle par le qubit ancillaire pour obtenir $|\phi((a \cdot c_1c_2 + b) \bmod N)\rangle$.

Partie 3

Il faut maintenant réinitialiser le qubit ancillaire à $|0\rangle$. Pour ce faire, on se base sur l'identité suivante :

$$(a \cdot c_1c_2 + b) \bmod N \geq a \cdot c_1c_2 \iff a \cdot c_1c_2 + b - N < 0$$

On sait donc que si $(a \cdot c_1c_2 + b) \bmod N \geq a \cdot c_1c_2$, alors le qubit ancillaire est à $|1\rangle$. Sinon, il est à $|0\rangle$. On utilise

$\text{CC}\Phi\text{ADD}_a^\dagger$ pour soustraire $a \cdot c_1c_2$ de $|\phi((a \cdot c_1c_2 + b) \bmod N)\rangle$, puis on applique la QFT inverse. Cette fois, selon l'identité, le qubit de poids fort vaut $|0\rangle$ si $a \cdot c_1c_2 + b - N < 0$ (donc le qubit ancillaire est forcément $|1\rangle$) et $|1\rangle$ sinon (donc le qubit ancillaire est forcément $|0\rangle$). Pour faire la correction appropriée sur le qubit ancillaire, on inverse la valeur du qubit de poids fort qui contrôle ensuite une porte CNOT sur le qubit ancillaire. On vient alors de réinitialiser le qubit ancillaire et le reste du circuit ramène le qubit de poids fort à sa bonne valeur avant de rajouter $a \cdot c_1c_2$. Pour résumer, on a calculé le modulo et le qubit ancillaire est de retour à $|0\rangle$.

Finalement, l'inverse de l'additionneur modulaire produit $|\phi(b - a \cdot c_1c_2)\rangle$ si $b \geq a$ et $|\phi(b - a \cdot c_1c_2 + N)\rangle$ sinon, ce qui est équivalent pour les deux cas à $|\phi((b - a \cdot c_1c_2) \bmod N)\rangle$.

8.3 Multipieur modulaire $\text{C}\Phi\text{MULT}_a\text{MOD}_N$

Pour des entiers positifs a, b, x et N sur n bits, le mutiplieur modulaire calcule $(b + ax) \bmod N$. Puisque $x = \sum_{j=0}^{n-1} x_j 2^j$, il va de soi que $(b + ax) \bmod N = (b + ax_0 2^0 + ... + ax_{n-1}2^{n-1}) \bmod N$. En utilisant les identités de la section 1, on a

$$
\begin{aligned}
&(b + ax_0 2^0 + ... + ax_{n-1}2^{n-1}) \bmod N = [(b + ax_0 2^0 + ... + ax_{n-2}2^{n-2}) \bmod N + (ax_{n-1}2^{n-1}) \bmod N] \bmod N \\
&= [((b + ax_0 2^0 + ... + ax_{n-3}2^{n-3}) \bmod N + (ax_{n-2}2^{n-2}) \bmod N) \bmod N + (ax_{n-1}2^{n-1}) \bmod N] \bmod N = ... \\
&= [...((b) \bmod N + (ax_0 2^0) \bmod N) \bmod N + ... + (ax_{n-1}2^{n-1}) \bmod N] \bmod N \\
&= [...\underbrace{(b + (ax_0 2^0) \bmod N) \bmod N}_{\text{CC}\Phi\text{ADD}_{a2^0 \bmod N}\text{MOD}_N} + ... + (ax_{n-1}2^{n-1}) \bmod N] \bmod N
\end{aligned}
$$

On voit alors que le multiplieur modulaire peut être réalisé en appliquant successivement $\text{CC}\Phi\text{ADD}_{a2^j \bmod N}\text{MOD}_N$ pour j allant de 0 à $n-1$. Pour ce faire, on initialise un registre de n qubits contenant $|x\rangle$ et un qubit de contrôle $|c\rangle$ s'ajoute aussi pour autoriser ou non l'opération. De nouveau, on prépare un second registre de $n+1$ qubits avec $|\phi(b)\rangle$. Il y a aussi un qubit ancillaire nécessaire pour utiliser $\text{CC}\Phi\text{ADD}_a\text{MOD}_N$ qui est implicitement présent à la figure 10. Au total, il nous faut $2n+3$ qubits pour réaliser le multiplieur modulaire. L'opération désirée correspond donc à

$$
\text{C}\Phi\text{MULT}_a\text{MOD}_N(a, N, |c\rangle, |x\rangle, |\phi(b)\rangle) = |c\rangle\ |x\rangle\ |\phi((b + ax \cdot c) \bmod N)\rangle \tag{12}
$$

Il faut noter qu'il est nécessaire de calculer classiquement $2^j a \bmod N$ avant de le donner en argument à chaque porte. De cette manière, toutes les valeurs vont tenir sur $n+1$ qubits et on diminue le nombre de qubits qu'il aurait fallu autrement pour faire le calcul. Le calcul classique du modulo se fait en temps polynomial. Pour terminer, $\text{C}\Phi\text{MULT}_a\text{MOD}_N^\dagger$ permet de produire $|\phi((b - ax \cdot c) \bmod N)\rangle$.

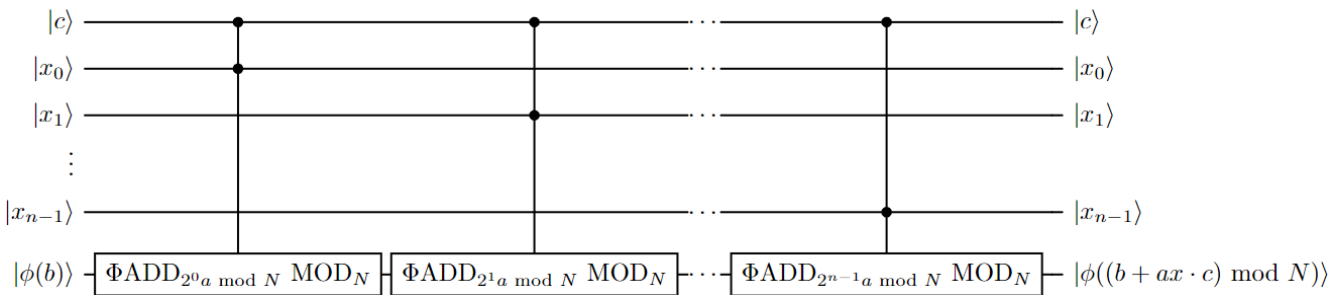

FIGURE 10 – Circuit pour le multiplieur modulaire $\text{C}\Phi\text{MULT}_a\text{MOD}_N$

8.4 CU_a

On a maintenant tous les blocs nécessaires pour faire la porte CU_a. D'abord, l'initialisation des registres est la même que pour la section précédente, sauf qu'on pose $b = 0$. De plus, on rajoute comme condition que $\text{pgcd}(a, N) = 1$. Puis, on applique $\text{C}\Phi\text{MULT}_a\text{MOD}_N$ afin d'avoir $|x\rangle$ sur le premier registre et $|ax \bmod N\rangle$ sur le deuxième registre (après la porte $\text{QFT}^\dagger$). Ensuite, une porte SWAP contrôlée par $|c\rangle$ échange tout le premier registre avec les n premiers qubits du deuxième registre, ce qui amène $|ax \bmod N\rangle$ sur le premier registre et $|\phi(x)\rangle$ sur le second (après la porte QFT). Finalement, une porte $\text{C}\Phi\text{MULT}_{a^{-1}}\text{MOD}_N^\dagger$ permet de réinitialiser le deuxième registre à $|\phi(0)\rangle$ tout en gardant $|ax \bmod N\rangle$ dans le premier registre. On rappelle que a^{-1} est l'inverse multiplicatif modulo N de a qui existe grâce au fait que $\text{pgcd}(a, N) = 1$. Donc, $\text{C}\Phi\text{MULT}_{a^{-1}}\text{MOD}_N^\dagger$ change $|\phi(x)\rangle$ pour l'état $|\phi((x - a^{-1}(ax \bmod N)) \bmod N)\rangle$ sur le second registre qui équivaut, en utilisant les identités de la section 1, à $|(x - a^{-1}ax) \bmod N\rangle$. Comme on sait que $a^{-1}a \equiv 1 \bmod N$, l'état final correspond en fait à $|\phi(x - 1 \cdot x) \bmod N\rangle = |\phi(0)\rangle$. Au final, on retrouve bien la porte CU_a.

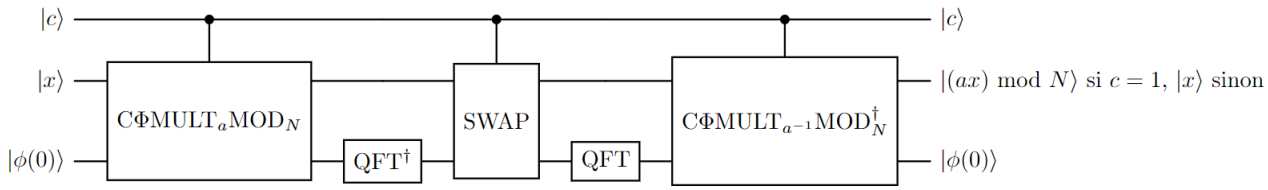

FIGURE 11 – Circuit pour CU_a

Si on se ramène à la section sur le problème de la recherche d'ordre, on voudrait appliquer $CU_a^{2^k}$. En fait, $U_a^{2^k}|x\rangle = |a^{2^k} \bmod N\rangle = U_{a^{2^k}}|x\rangle$ où on peut calculer classiquement a^{2^k} en temps polynomial. Dès lors, on peut résoudre la recherche d'ordre à l'aide de la QPE. Au final, la recherche d'ordre nécessite $2n + 2 + m$ qubits où m est le nombre de qubits de précision de la QPE. Avec le *one qubit trick*, le nombre de qubits diminue à $2n + 3$. Globalement, la recherche d'ordre avec la méthode de Beauregard a une complexité polynomiale [6].

9 Algorithme de Shor

Algorithme de Shor

Entrée : Un nombre N composé et positif
Sortie : Un facteur non-trivial de N

1: **Si** N est pair **alors :**
2: **retourner** 2
3:
4: **Si** N est une puissance pure, c'est-à-dire si $N = x^y$ pour des entiers $x \geq 1$ et $y \geq 2$, **alors :**
5: **retourner** (x, y)
6:
7: Choisir aléatoirement a où $1 < a < N - 1$.
8: **Si** $\text{pgcd}(a, N) > 1$, **alors :**
9: **retourner** $\text{pgcd}(a, N)$
10:
11: Utiliser la recherche d'ordre pour trouver l'ordre r de $a^r \equiv 1$ modulo N.
12: **Si** r est pair et que $a^{\frac{r}{2}} \not\equiv -1 \bmod N$, **alors :**
13: Calculer $\text{pgcd}(a^{\frac{r}{2}} \pm 1, N)$.
14: **Si** $\text{pgcd}(a^{\frac{r}{2}} + 1, N)$ et/ou $\text{pgcd}(a^{\frac{r}{2}} - 1, N)$ sont des facteurs de N, **alors :**
15: **retourner** $\text{pgcd}(a^{\frac{r}{2}} + 1, N)$ et/ou $\text{pgcd}(a^{\frac{r}{2}} - 1, N)$
16:
17: Revenir à la ligne 7.

L'algorithme de Shor est un algorithme quantique qui permet de trouver les facteurs premiers d'un entier N positif et composé [8]. La procédure trouve un facteur premier en complexité polynomiale [3], ce qui est meilleur que n'importe quel algorithme classique à ce jour. En fait, il s'agit d'un des algorithmes les plus prometteurs en informatique quantique vu sa puissance qui permettrait, en théorie, de briser le chiffrement RSA. Le chiffrement RSA, même s'il est plus complexe que cela, se base justement sur la difficulté de factoriser efficacement des nombres pour chiffrer des informations sensibles. Bref, l'algorithme de Shor peut avoir, si les ordinateurs quantiques se développent suffisamment, un impact considérable dans le domaine de la cryptographie.

La procédure développée par Peter Shor emploie, entre autres, la recherche d'ordre dont on a parlé à la section 5. Le reste de l'algorithme est classique bien que la recherche d'ordre, qui s'effectue sur un ordinateur quantique, soit le plus gros du travail pour trouver un facteur.

On se ramène maintenant au pseudocode présenté en début de section (qu'on explique plus en détails dans les compléments). Les deux premiers « **Si** » de la procédure sont assez évidents. Effectivement, un nombre pair a forcément 2 comme facteur premier et une puissance pure a comme seul facteur x^y. Puis, on choisit une valeur aléatoire a qu'on nommera la base. Le troisième « **Si** » est encore assez simple puisqu'un pgcd supérieur à 1 indique qu'il divise a et N, donc qu'il est un facteur de N. Autrement, cette étape s'assure que a et N soient copremiers. On arrive ensuite à la recherche d'ordre qui est l'unique section quantique de l'algorithme. On peut démontrer à l'aide de théorèmes que $\text{pgcd}(a^{\frac{r}{2}}+1, N)$ et/ou $\text{pgcd}(a^{\frac{r}{2}}-1, N)$ sont des facteurs non-trivaux de N si les conditions sur r et $a^{\frac{r}{2}}$ sont respectées (voir section E dans les compléments). Il se peut que l'algorithme échoue et, dans ce cas, on retourne en arrière pour choisir une nouvelle base. On peut montrer qu'à chaque itération, la probabilité de succès de l'algorithme de Shor est supérieure à $\frac{1}{2}$ (voir section E des compléments). On peut maintenant factoriser des nombres !

If computers that you build are quantum,

Then spies of all factions will want 'em.

Our codes will all fail,

And they'll read our email,

Till we've crypto that's quantum, and daunt 'em.

— Jennifer et Peter Shor

To read our E-mail, how mean

of the spies and their quantum machine ;

Be comforted though,

they do not yet know

how to factorize twelve or fifteen.

— Volker Strassen

Remerciements

Je tiens d'abord à remercier Derek Courchesne qui m'a donné l'opportunité de travailler avec lui sur ce projet et qui a grandement aidé à ma compréhension des concepts clés de l'algorithme. Ton soutien a été essentiel dans mon apprentissage et dans mon développement de nouvelles compétences. De plus, je souhaite adresser mes remerciements à Stepan Gorgutsa et à toute l'équipe du C2T3 pour cette opportunité de stage incroyable. Merci également à Elisabeth, à Moras et à toute la chatastrophe pour leur compagnie tout au long de cet été enrichissant.

Références

[1] N. Koblitz, *A course in number theory and cryptography.* Berlin, Heidelberg : Springer-Verlag, 1987.

[2] E. Stein and R. Shakarchi, *Fourier Analysis : An Introduction.* Princeton lectures in analysis, Princeton University Press, 2011.

[3] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information.* Cambridge University Press, 2000.

[4] S. Parker and M. B. Plenio, "Efficient factorization with a single pure qubit and¡mml :math xmlns :mml="http ://www.w3.org/1998/math/mathml" display="inline"¿¡mml :mi¿log¡/mml :mi¿¡mml :mi mathvariant="italic"¿n¡/mml :mi¿¡/mml :math¿mixed qubits," *Physical Review Letters*, vol. 85, p. 3049–3052, Oct. 2000.

[5] T. G. Draper, "Addition on a quantum computer," 2000.

[6] S. Beauregard, "Circuit for shor's algorithm using 2n+3 qubits," 2003.

[7] A. Pavlidis and D. Gizopoulos, "Fast quantum modular exponentiation architecture for shor's factorization algorithm," 2013.

[8] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, p. 1484–1509, Oct. 1997.