Création de circuits quantiques pour l'encodage de fonctions booléennes

On étudie ici la problématique de pouvoir construire systématiquement une fonction booléenne avec un ordinateur quantique, suivant plusieurs modèles.

En informatique classique, l'ensemble des fonctions booléennes peuvent être décrites à l'aide des opérateurs **NAND** et **NOR**. Il s'agit donc de pouvoir les transcrire en quantique, et de pouvoir établir un système de combinaison de ces portes, pour permettre l'élaboration des circuits.

Modèle n-qubits d'entrée, m-qubits de sortie

Dans ce premier modèle, on considère un registre de n qubits composant l'entrée du système, un registre de m qubits composant la sortie du système, et un registre de k qubits auxiliaires pour certaines opérations intermédiaires.

Pour cette construction, on se base sur la porte quantique **X** et ses équivalents composés **CNOT**, **CCNOT** (Toffoli), etc. On fourni alors un certain nombre de circuits de base pouvant être recomposés pour former des circuits plus complexes.

La compilation d'une fonction booléenne passe alors par 4 étapes :

1. Écriture de la table de vérité,
2. Pour chaque sortie donnant 1, former une porte **NOT** controlée. Chaque entrée va servir de contrôle, par 1 si l'entrée est à 1, et par 0 si l'entrée est à 0,
3. Développer le circuit résultant pour n'avoir que des portes **NOT** controlées par 0,
4. Simplifier le circuit en éliminant les doublons.

Étape 1 : établissement des premières portes contrôlées

La figure 1 représente une porte **NOT** contrôlée. On note que les qubits de contrôle sont indiqués par • (contrôle par 1) et par ∘ (contrôle par 0). Le dernier qubit est la cible (*target*). On effectue l'opération **NOT** sur la cible si et seulement si les bits de contrôles respectent leur condition (si il sont à 1

pour ceux qui sont contrôlés par 1, et si ils sont à 0 pour ceux contrôlés par 0).

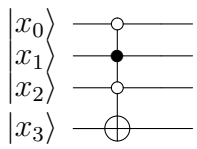


FIGURE 1 – Porte **NOT** contrôlée

Cette porte en revanche ne peut pas être construite, on ne dispose en effet que des portes **NOT** contrôlées par 1 et pas de celles contrôlées par 0.

Exemple 1 *Soit la fonction booléenne* $f(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee (x_3 \wedge \neg x_2) \vee (x_1 \wedge x_3)$. *Sa table de vérité est la suivante :*

x_1	x_2	x_3	$F(x_1, x_2, x_3)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

On a quatre sorties à 1. La figure 2 représente donc le circuit initial qu'on obtient.

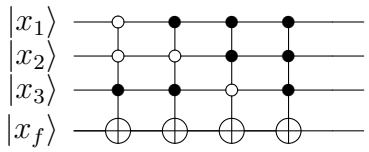


FIGURE 2 – Circuit quantique pour $f(x_1, x_2, x_3)$

Étape 2 : Développement du circuit

La deuxième étape consiste à prendre le circuit obtenu précédemment et à le développer de façon à n'obtenir que des portes **NOT** contrôlées par 0. Dans le principe, une porte ayant un mélange de contrôle par 0 et par 1 va être équivalent à la combinaison des portes contrôlées par 0, qui vont avoir des contrôles de moins en combinaison sur les contrôles par 1. Un exemple

est plus clair pour comprendre. Reprenons la porte de la figure 1. Elle est en fait équivalente au circuit 3 :

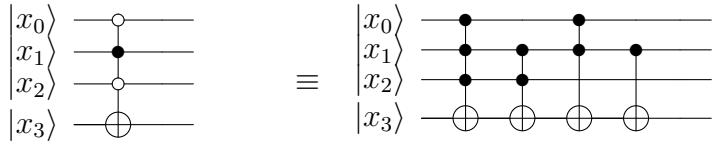


FIGURE 3 – Équivalent sans contrôles par 0

Exemple 2 *On reprends notre exemple de fonction booléenne* $f(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee (x_3 \wedge \neg x_2) \vee (x_1 \wedge x_3)$*. Une fois développée, son circuit équivalent est illustré à la figure 4*

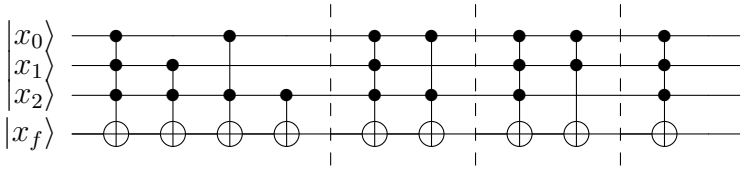
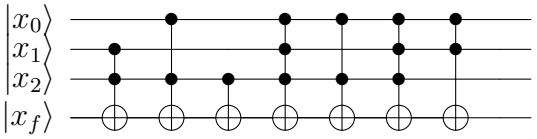


FIGURE 4 – Circuit quantique développé pour $f(x_1, x_2, x_3)$

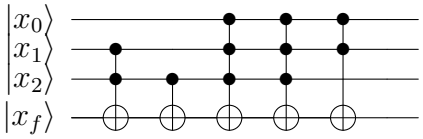
Étape 3 : Simplification du circuit

La dernière étape permet d'obtenir un circuit comportant le moins de portes possibles, en se basant sur le principe suivant : lorsqu'un circuit (ici composé de **CNOT**) est entouré de deux mêmes **CNOT**, alors celles-ci s'annulent et on peut alors enlever la paire doublon sans changer le résultat. En effectuant ce raisonnement récursivement, on arrive à obtenir un circuit minimal.

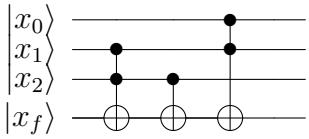
En reprenant l'exemple de la fonction booléenne précédente, on peut faire par étapes la simplification :



(a) Première simplification



(b) Deuxième simplification



(c) Dernière simplification

FIGURE 5 – Simplifications successives pour $f(x_1, x_2, x_3)$

Extension à des circuits plus complexes

Avec cette méthode, il est facile de construire l'ensemble des fonctions booléennes à 2 bits, ainsi qu'a 3 bits. Pour construire des circuits plus complexes, on dispose alors de deux façons de procéder.

Tout d'abord, on peut refaire ces étapes sur la fonction booléenne plus complexe, et trouver un circuit minimal étant composé de n entrées, et d'un seul qubit auxiliaire pour la sortie.

En revanche, si on ne veut pas établir la table de vérité avant la construction du circuit, on peut utiliser les fonctions élémentaires et construire un circuit a partir de la chaine de caractère représentant la fonction. Il est important de noter que la méthode présentée ici ne modifie pas les entrées de la fonction, on peut donc les réutiliser comme on le souhaite pour venir greffer des fonctions supplémentaires au circuit. Cette méthode s'apparente algrithmiquement à la compilation des fonctions sur les ordinateurs classiques. En revanche, par rapport au circuit minimal qu'on pourrait trouver, on aura ici un qubit auxiliaire par sous-circuit, et un nombre de portes bien plus important. Pour des considérations de limitations matérielles qui sont pour le moment importantes, cette deuxième façon de procéder peut ne pas donner des résultats implémentables pour de trop grosses fonctions.