IFEAY050 – Rapport de projet final : Simulateur de circuit Combinatoire

Trong Hieu NGUYEN

Viet Anh BUI

Janvier 2022

Contents

1. Présentation du projet :

Le but de projet est de réaliser un simulateur de circuit combinatoire. On va donc réaliser un simple circuit constitué des plusieurs portes logiques de base. Ce circuit doit être capable de donner à l'utilisateur son valeur à partir des portes d'entrée « InputGate » qui sont initialisé avec la valeur false. On peut contrôler sa valeur pour modifier la sortie d'ouput.

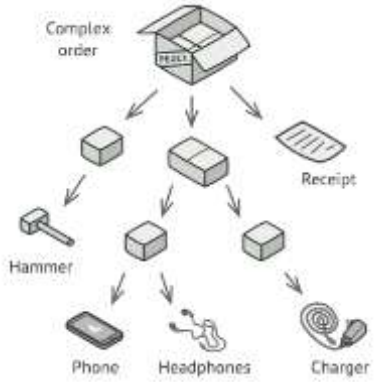
Les extensions pour les circuits seront :

- L'affichage sous une forme 2D.
- Conversion en forme textuelle.
- Lecture une expression textuelle et renvoyer un circuit correspondant.
- Capacité de sauvegarder le circuit dans un fichier et puis le relire pour la reconversion en circuit.

2. Design pattern et UML :

2.1 Design pattern :

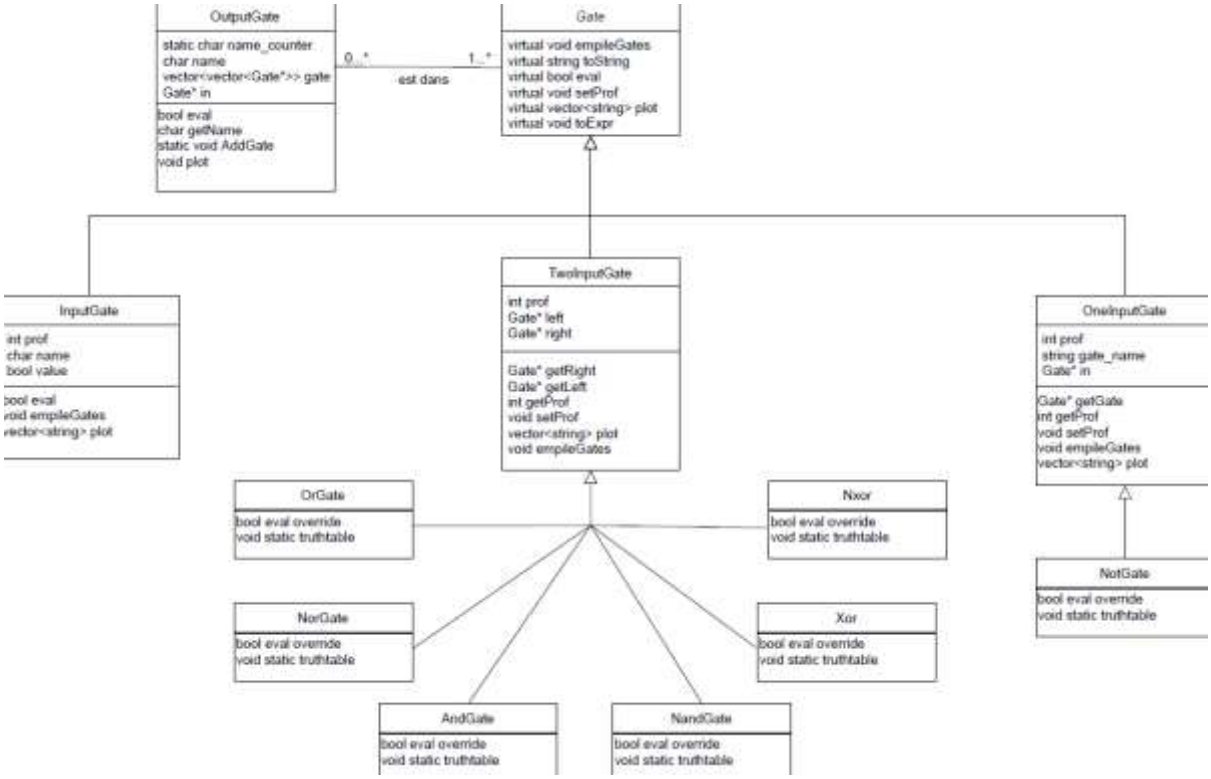
On peut voir facilement qu'un circuit est composé de plusieurs portes logiques et aussi des portes d'entrée. Alors pour évaluer la valeur d'un circuit, il faut chercher récursivement ses gates. On peut considérer un circuit comme un arbre, avec l'output gate comme la racine, les gates logiques sont les nœuds et finalement les gates d'entrée comme les feuilles. Ce qui implique un composite design pattern.



Un exemple de composite design pattern

2.2 UML :

Voici notre UML :



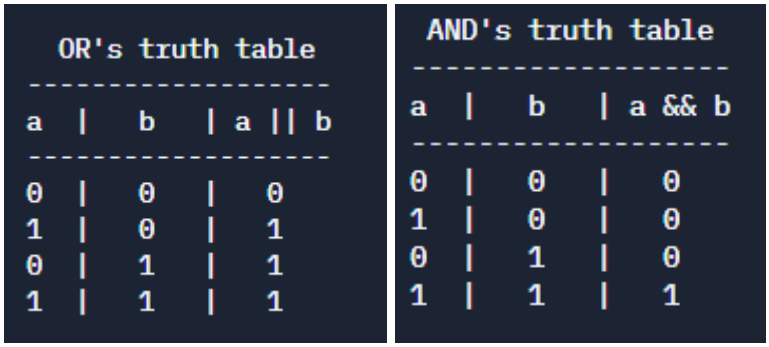
3. Le fonctionnement normal d’un circuit :

3.1 Un circuit c’est quoi ici ?

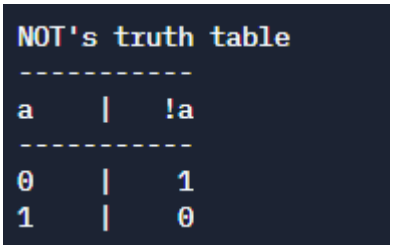
Ici, un circuit est représenté par un OutputGate et des Gates qui sont créés soit par une autre gate ou « connectée par les fils » avec autre gate si c’est une gate logique ou soit par défaut d’une valeur de false si c’est une InputGate.

3.2 Construire une porte logique :

Même principe des couleurs, on peut fabriquer des autres portes logiques à partir des 3 portes de base : AND, OR, NOT.



Fonctions static truthtable() de AndGate et OrGate



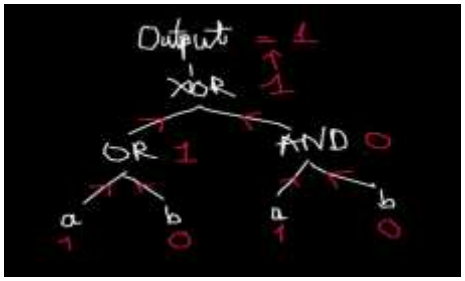
Fonctions static truthtable() de NotGate

Et pour les autres portes logiques, on peut les construire grâce à ces 3 portes, par exemple :

```
bool NorGate::eval() const {
    Gate* a = this->getRight();
    Gate* b = this->getLeft();
    Gate* or_tmp = new OrGate(a,b);
    Gate* not_tmp = new NotGate(or_tmp);
    return not_tmp->eval();
}
```

Eval de NorGate est basé sur un OrGate suivi par un NotGate

Finalement, l'évaluation d'un circuit se fait depuis la racine qui est OutputGate et sa valeur va être renvoyée récursivement. Voici un moche démo :



4. Les extensions :

4.1 Les truthtables :

Chaque porte a sa propre fonction qui affiche sa table de vérité et elles sont créés de façon static pour qu'on puisse faires des appels sur ces tables de vérité n'importe quand.

```
XorGate::truthtable();
```

```
 XOR's truth table
------------------
a  |   b   | a || b
------------------
0  |   0   |   0
1  |   0   |   1
0  |   1   |   1
1  |   1   |   0
```

4.2 L'affichage 2D :

On peut « plot » un circuit simplement par faire appel sur la fonction plot() d'OutputGate qui va exécuter plot() sur ses gates dedans récursivement.

```
InputGate *a = new InputGate('a');
InputGate *b = new InputGate('b');
InputGate *c = new InputGate('c');
Gate *or1 = new OrGate(a,b);
Gate *and1 = new NandGate(a,b);
Gate *and2 = new XorGate(or1,and1);
Gate *nor1 = new NorGate(c,a);
Gate *not1 = new NotGate(nor1);
Gate *and3 = new AndGate(not1,and2);
OutputGate *A = new OutputGate(and3);
A->plot();
```

```
A's 2D representation !!!!!
                A:0
                AND
                 |
        *****************
        |                |
       NOT              XOR
        |                |
        |           *********
        |           |       |
       NOR         OR_    NAND_
        |           |       |
      *****       *****   *****
      |   |       |   |   |   |
    c:0 a:0     a:0 b:0 a:0 b:0
```

4.3 Conversion text <=> circuit :

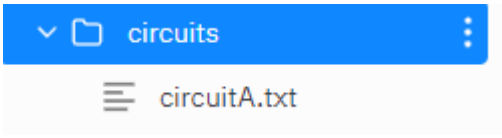
On peut aussi faire des conversions entre text circuit, sauvegarder un circuit dans un fichier et puis le relire. Voici les codes :

```
A->toExpr();
cout << endl << endl;
A->saveToFile();
OutputGate *B = new OutputGate(GateFromExpr("XOR(OR(a,b),AND(a,b))"));
B->plot();
```

```
A can be translated into the following text:
A=AND(NOT(NOR(c,a)),XOR(OR(a,b),NAND(a,b)))

A saved succesfully to directory ./circuit :D
Please check the file circuits/circuitA.txt for the saved circuit

B's 2D representation !!!!!
        B:0
        XOR
         |
    *********
    |       |
   OR_     AND
    |       |
  *****   *****
  |   |   |   |
 a:0 b:0 a:0 b:0
```



Projetkotext/circuits/circuitA.txt

```
Textual expression of circuit :
A can be translated into the following text:
A=AND(NOT(NOR(c,a)),XOR(OR(a,b),NAND(a,b)))

A's 2D representation !!!!!
                                A:0
                                AND
                                 |
                  ******************
                  |                  |
                 NOT                XOR
                  |                  |
                  |             *********
                  |             |       |
                 NOR           OR_    NAND_
                  |             |       |
                *****         *****   *****
                |   |         |   |   |   |
               c:0 a:0       a:0 b:0 a:0 b:0
```

Ses comportements correspondants.