Projet de programmation Java : Personnaliser Logisim

Validation vendredi 11 janvier 2013

1 Introduction

Un **circuit logique** est constitué de trois types de composants principaux reliés entre eux par des lignes :

- **Entrées** : représentent des variables logiques dont la valeur est booléenne ou binaire (0 ou 1).
- **Portes logiques (Gates)** : chaque porte logique (e.g., "AND Gate", "OR Gate", "NOT Gate", etc.) possède une table de vérité et/ou une formule logique qui définit son résultat en fonction de son ou ses entrées. La liste des composants classiques est donnée par la figure 1.
- **Sortie(s)** : chaque circuit logique peut avoir une ou plusieurs sorties dont chacune peut avoir deux états possibles : 0 ou 1.

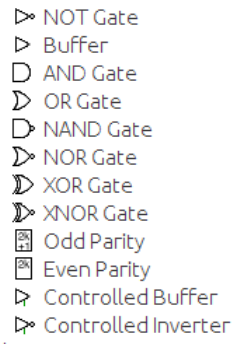

FIGURE 1 – Différentes portes logiques

Dans ce projet, on se limite aux circuits logiques combinatoires (pas de notion de temps). Chaque circuit logique peut être représenté par une ou des formules du calcul des propositions. Chaque formule est écrite grâce à quatre opérateurs ou fonctions logiques suivants :

- **OR** est représenté par le symbole $+$,
- **AND** est représenté par le symbole $\bullet$,
- **NOT** est représenté par une barre au-dessus de la variable inversée ou par un '/' ou '$\sim$' ou '$\neg$' devant la variable inversée,
- e**X**clusive **OR** est représenté par un plus encerclé $\bigoplus$.

Par exemple, la formule $\overline{a}.b + a.\overline{b}$ est schématisée comme dans la figure 2.

2 Logiciel de simulation des circuits logiques

2.1 Description

Logisim [1] est un logiciel open-source (développé en Java) permettant de dessiner et de simuler les circuits logiques (voir la figure 3) mais aussi de construire des circuits à partir de formules logiques.

1. `http://ozark.hendrix.edu/~burch/logisim/`

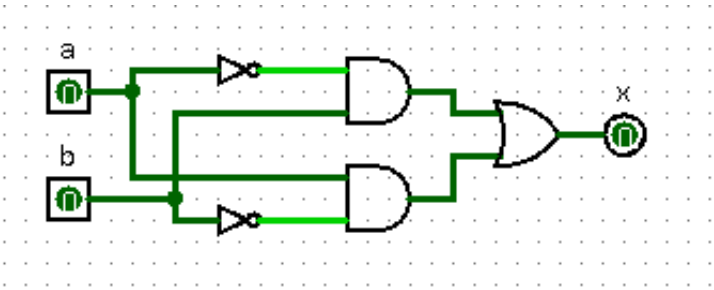

FIGURE 2 – Schéma du circuit logique

Nous présentons ici rapidement l'interface graphique GUI de Logisim afin de faire le tour des fonctionnalités du logiciel. L'interface est principalement divisée en cinq parties :

1. **canevas** : représente une zone où on peut dessiner et modifier les attributs (position, couleur, etc.) des objets graphiques,
2. **barre de menus** : comprend plusieurs menus et sous-menus permettant de créer, sauvegarder ou d'ouvrir un circuit, etc. En particulier, le menu "Simulate" permet de simuler un circuit logique. Étant données les valeurs d'entrées, Logisim calcule le résultat (la sortie) du circuit.
3. **barre d'outils** : affiche quelques outils de base (e.g., entrées, sorties, portes logiques, etc.) qui sont nécessaires pour dessiner et simuler un circuit logique. L'ajout des composants au circuit logique peut se faire par un drag-and-drop sur le canevas. Pour relier les deux composants donnés, on sélectionne le premier et dirige le pointeur de la souris vers le deuxième.
4. **panneau d'exploration** : contient tous les outils nécessaires pour dessiner un circuit logique. Il permet de sélectionner, d'ajouter des composantes au circuit logique (avec un drag-and-drop sur le canevas).
5. **table d'attributs** : affiche les attributs d'un objet spécifique (circuit, porte, entrées, sorties, fils de connexion, etc.).

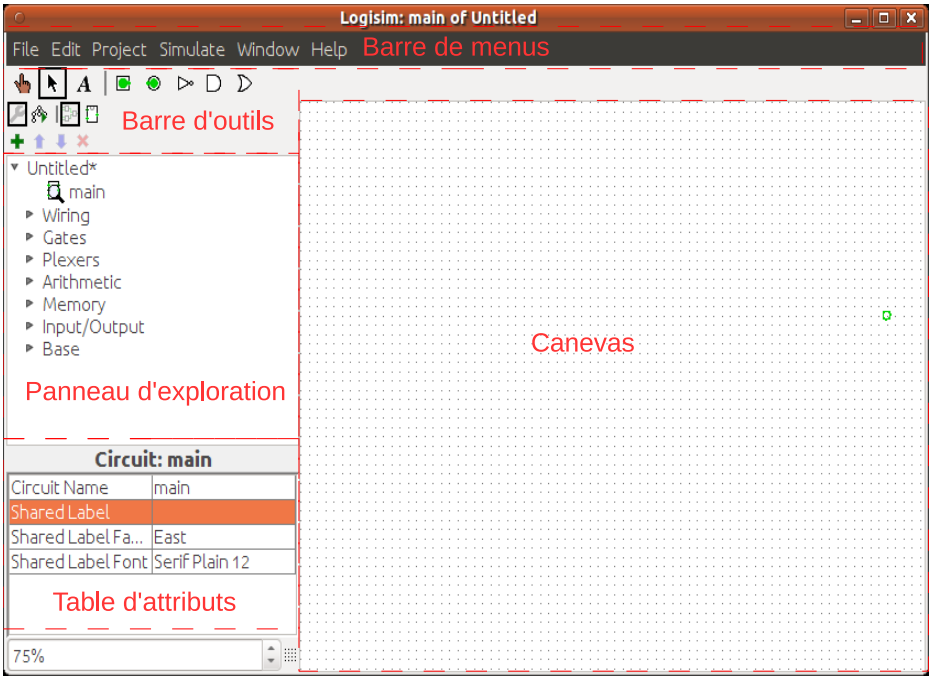

FIGURE 3 – Interface graphique de Logisim

Pour avoir plus de renseignements sur le logiciel Logisim, vous êtes invités à lire la documentation [2] en anglais du projet sur le site Web de Logisim.

Il est possible de charger des bibliothèques personnalisées dans Logisim à l'aide du menu Project → Load Library.

2.2 Téléchargement et exécution

Logisim est téléchargeable sur le site de SourgeForge.net en utilisant le lien suivant :
`http://sourceforge.net/projects/circuit/`
Vous le trouverez également dans l'archive disponible sous moodle.

Si vous souhaitez (préférable pour comprendre le sujet) exécuter le programme, soit vous double-cliquez sur le fichier .jar soit vous tapez la ligne de commande suivante :

java -jar *logisim-xx.jar*

sachant que JRE ou JDK version 5+ doit être installé sur le système.

Le fichier .jar peut également être importé comme une bibliothèque de Java pour développer des applications simulant des circuits logiques.

3 Travail demandé

L'objectif de ce projet est de développer des fonctionnalités pour un logiciel existant (Logisim). Les fonctionnalités peuvent exister et il faudra les personnaliser, elles peuvent ne pas exister et il faudra les ajouter. Vous aurez donc à développer une bibliothèque java (.jar) qui devra être compatible avec Logisim.

3.1 Partie 1 : Ajout de composants

Vous devez redéfinir les composants classiques (OR, NAND, XOR...) unaire et binaires. Cet exercice est essentiel pour identifier les classes à ajouter. Vous vous baserez sur le squelette fourni pour lequel les portes AND et NOT sont proposées (voir figure 4).

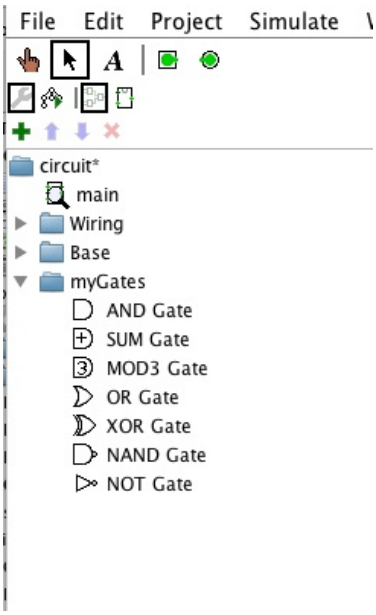

FIGURE 4 – Exemple de menu "myGates"

2. `http://ozark.hendrix.edu/~burch/logisim/docs.html`

Vous aurez ensuite à étendre votre panel de composants possibles parmi lesquels au moins les 2 composants suivants :

– le composant "modulo 3" qui possède 4 entrées booléennes (correspondant à un nombre entier en représentation binaire) et une sortie booléenne qui indique si cet entier est multiple de 3 (voir figure 5). Exemples : $0011 = (3)_2$ est multiple de 3, $0110 = (10)_2$ n'est pas multiple de 3.

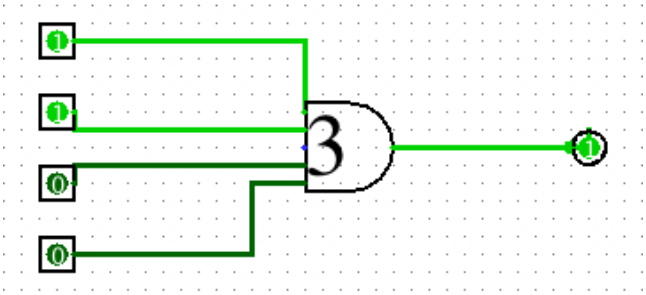

FIGURE 5 – Exemple de circuit modulo 3 avec (0011)

– le composant "somme" qui possède 2 entrées entières et une sortie entière qui indique la somme des deux entrées (voir figure 6). Dans un premier temps, vous ajouterez un affichage sur la sortie standard pour la valeur calculée, ensuite, vous ajouterez une fenêtre avec toutes les valeurs précédemment calculées.

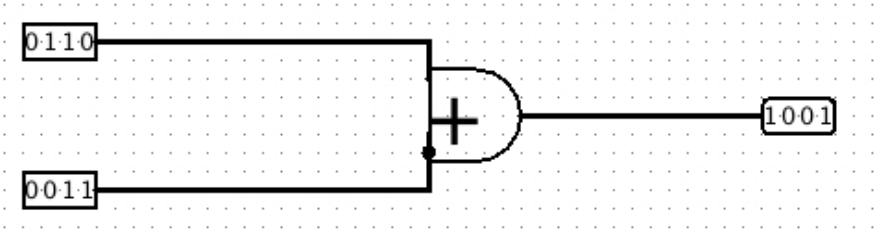

FIGURE 6 – Exemple de circuit somme

Pour les trinômes, vous devez ajouter des composants entrées/sorties plus adaptés que ceux proposés (visuellement) figure 6. Vous afficherez les entiers plutôt que leur représentation en binaire.

3.2 Partie 2 : Personnaliser l'interface graphique

Dans cette partie, l'objectif principal est de développer un projet qui permet de saisir une formule composée des entrées, une sortie et des opérateurs logiques (AND, OR, NOT).

L'interface graphique du nouveau simulateur est illustrée par la figure 7. L'utilisateur saisit une expression en utilisant des lettres (a-z) pour désigner les variables d'entrée/sortie et des symboles (+, '.', '!') pour désigner les opérateurs OR, AND et NOT. Le système la normalise (revert) et la convertit en une formule conventionnelle en remplaçant le symbole '!' par une barre au dessus de la variable. Ensuite, une table de vérité associée à la formule est affichée. A la fin, quand l'utilisateur clique sur le bouton "Build circuit", le circuit logique correspondant est automatiquement visualisé sur l'écran.

Pour résumer, l'interface graphique doit permettre d'exécuter les fonctions suivantes :

- F1 : saisir une expression algébrique au clavier,
- F2 : dessiner le circuit logique correspondant à l'expression algébrique saisie au clavier,
- F3 : afficher la table de vérité associée à l'expression algébrique.

Suggestion : Organiser votre code source afin de respecter les règles d'héritage des classes Java sans modifier les classes de Logisim.

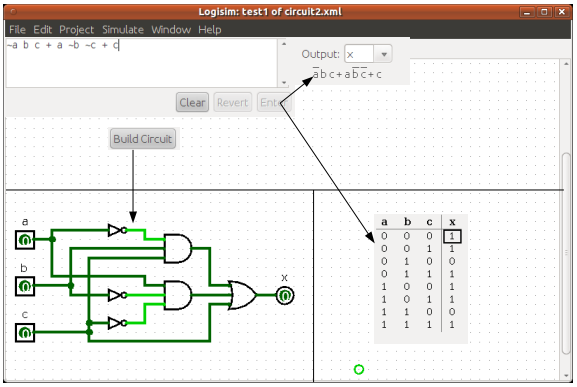

FIGURE 7 – Interface graphique du nouveau simulateur

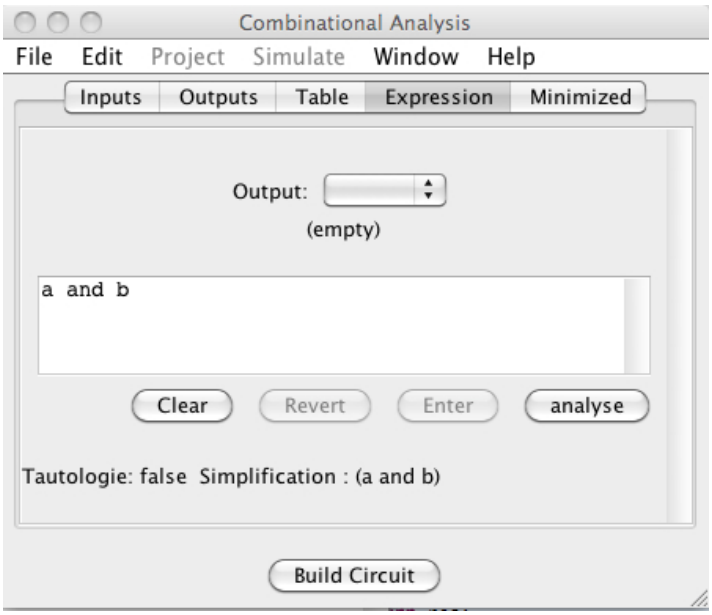

FIGURE 8 – Interface de saisie d'une formule à analyser

3.3 Partie 3 : Lien avec Ocaml

Dans cette partie vous ferez le lien avec votre analyseur de formule écrit en OCaml. Le but est d'étendre l'interface de saisie des expressions accessible via le menu : Projet > Analyze Circuit. Vous rajouterez un bouton "analyse" qui permettra d'appeler votre programme OCaml en lui passant la formule saisie comme paramètre. La capture d'écran 8 montre le résultat attendu.

3.4 Rapport

Vous devez rédiger un rapport de 8 à 10 pages qui décrit la façon dont vous avez abordé le problème posé. Vous pouvez y mettre des schémas qui décrivent l'architecture de vos classes, comment vous avez résolu un problème particulier, la façon dont vous vous êtes organisés dans le temps et en groupe... Vous trouverez sur moodle des conseils pour rédiger un rapport, en particulier : n'attendez pas la dernière minute.

4 Détails

4.1 Le squelette fourni

Vous trouverez sur moodle le squelette de la bibliothèque que vous devez construire. Et voici quelques commentaires pour vous y retrouver dans le squelette.

- le package **customLogisim** permet de définir la classe Main qui exécute Logisim avec la bibliothèque que vous définissez [3].
- le package **fr.irit.projetl3info** contient l'interface Java Component qui doit être implémentée par vos propres composants. La classe ComponentFactory permet de fabriquer lesdits composants. Vous devrez mettre à jour cette classe.
- le package **fr.irit.projetl3info.logisimInterface** contient les classes qui vont faire le lien avec Logisim. Nous vous fournissons les classes MyAndGate et MyNotGate pour vous donner un exemple. Vous aurez à utiliser ces modèles pour définir les autres composants. Vous pouvez évidemment améliorer ces classes pour obtenir un meilleur affichage ou pour augmenter leur généricité. La classe ProjetComponentsLibrary est utilisée en particulier pour la barre d'outils dans Logisim. Elle aussi doit être mise à jour selon les composants que vous ajouterez.

A partir de ce squelette vous devez concevoir et développer votre(vos) propre(s) package(s) qui permet(tent) de définir vos composants. Ainsi la définition d'une classe pour la porte AND devrait implémenter l'interface Component pour assurer le lien vers Logisim. Lors de la conception, pensez à la possibilité de définir des composants unaires, binaires ou n-aires ou même des composants ayant des entrées/sorties entières (plutôt que booléennes) afin d'être le plus "objet" possible et de minimiser le travail de développement pour les méthodes "ressemblantes".

4.2 Le développement

Créer un projet Java avec Eclipse (ou NetBeans), importer le répertoire “src” ainsi que les bibliothèques .jar. Dans les propriétés du projet, ajouter la bibliothèque de logisim dans le "Java Build Path".

- Pour l'interface graphique La classe “**Frame**” du package “**com.cburch.logisim.gui.main**” représente l'interface graphique principale de Logisim. Par exemple, pour ajouter les nouveaux objets de Swing (JButton, JTextField, etc.) à l'interface graphique, on étudie la classe “**com.cburch.logisim.gui.main.Frame**”.
  On y ajoute les objets et les méthodes comme suit :

```
/********************************** PROJET JAVA L3 ******************/

/***
 * Composants de l'interface
 */
JPanel customPanel; // panel a personnaliser
JLabel lblExpression;
JTextField txtExpression; // champ de saisie de l'expression logic
JButton btnBuildCircuit; // bouton "build circuit"
JButton btnShowTruthTable; // bouton affichage de la table de verite
JPanel tblTruthTable; // panel de la table de verite

private void addNewComponents() {

customPanel = new JPanel();

lblExpression = new JLabel();
lblExpression.setText("Logic expression: ");

// Expression logique
txtExpression = new JTextField(20);

// Build circuit
btnBuildCircuit = new JButton("Build circuit");
btnBuildCircuit
.setToolTipText("Build and display the logic circuit on the canvas.");
btnBuildCircuit.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {
        buildCircuit();
}
});
// Table de verite
btnShowTruthTable = new JButton("Show truth table");
btnShowTruthTable
.setToolTipText("Show truth table corresponding to the logic expression.");
btnShowTruthTable.addActionListener(new ActionListener() {
public void actionPerformed(java.awt.event.ActionEvent evt) {
        showTruthTable();
}
});
```

3. la bibliothèque fait s'exécuter Logisim avec ses propres caractéristiques. Il est aussi possible de générer le ".jar" de votre projet puis de le charger à partir du menu de Logisim

```
tblTruthTable = new JPanel();

// ajouter et afficher les labels, boutons, champs de texte, etc. sur la grille
GridBagLayout gb = new GridBagLayout();
GridBagConstraints gc = new GridBagConstraints();

customPanel.setLayout(gb);
gc.anchor = GridBagConstraints.LINE_START;
gc.fill = GridBagConstraints.NONE;

gc.gridx = 0;
gc.gridy = 0;

gb.setConstraints(lblExpression, gc);
customPanel.add(lblExpression);

gc.gridx = 1;
gb.setConstraints(txtExpression, gc);
customPanel.add(txtExpression);

gc.gridy = 2;
gc.gridx = 1;
JPanel buttonPanel = new JPanel();

gb.setConstraints(buttonPanel, gc);
buttonPanel.add(btnBuildCircuit);
buttonPanel.add(btnShowTruthTable);
customPanel.add(buttonPanel);
layoutCanvas.add(customPanel);

gc.gridx = 2;
gb.setConstraints(tblTruthTable, gc);
layoutCanvas.add(tblTruthTable);
tblTruthTable.setVisible(false);
}

/**
 * Construire et affichier le circuit sur le canvas
 */
private void buildCircuit() {
        // Aller voir les classes com.cburch.logisim.analyze.gui.Analyzer.java et
        // com.cburch.logisim.analyze.gui.BuildCircuitButton
}

/**
 * Affichage de la table de verite associee a son expression logique
 */
private void showTruthTable() {
// effacer la table
if (tblTruthTable != null) {
        layoutCanvas.remove(tblTruthTable);
        // pack();
}

tblTruthTable = new JPanel();
tblTruthTable.add(new JLabel("Truth table"));

// MODIFIER LE CODE POUR AFFICHER LA TABLE DE VERITE

tblTruthTable.add(new JLabel(txtExpression.getText()));
tblTruthTable.add(new JLabel("|"));
tblTruthTable.add(new JLabel("x"));

layoutCanvas.add(tblTruthTable);
        tblTruthTable.setVisible(true);
        pack(); // refresh frame
}
```

Enfin, on appelle la méthode “addNewComponents” dans la fonction Constructor de la classe Frame.

```
public Frame(Project proj) {
        this.proj = proj;
        setBackground(Color.white);
        setDefaultCloseOperation(WindowConstants.DO_NOTHING_ON_CLOSE);
        addWindowListener(new MyWindowListener());

        proj.addProjectListener(myProjectListener);
        proj.addLibraryListener(myProjectListener);
        proj.addCircuitListener(myProjectListener);
        computeTitle();

        // Initialisation des elements pour la vue "Layout"
        layoutToolbarModel = new LayoutToolbarModel(this, proj);
        layoutCanvas = new Canvas(proj);
        layoutZoomModel = new BasicZoomModel(AppPreferences.LAYOUT_SHOW_GRID,
        AppPreferences.LAYOUT_ZOOM, ZOOM_OPTIONS);

        // Ajouter les nouveaux elements au layout
        addNewComponents();

        ....
}
```

5 Organisation

5.1 Organisation du travail

Vous devrez déposer :

– une archive ".jar" qui correspond à la bibliothèque demandée (avec javadoc inclue) avant le vendredi 11 janvier 13h.
– un rapport (.pdf)

Attention, vos fichiers/répertoires doivent respecter les extensions demandées et commencer par vos noms de famille par ordre alphabétique (exemple : Nom1Nom2Rapport.pdf). Pensez à commenter votre code afin qu'il soit lisible par le correcteur. Le programme doit compiler sous peine d'une note nulle.

Lors de la validation, vous devrez
– présenter un programme qui fonctionne,
– pouvoir modifier votre programme à la demande,
– pouvoir justifier de vos choix de conception,

Vous êtes en binôme ou trinôme, mais notés aussi individuellement, pensez à vous répartir astucieusement le travail.

5.2 Du temps

Vous avez une salle de TP réservée par groupe du lundi 7 janvier au jeudi 10 janvier de 9h à 18h. Tous les enseignants seront présents lundi 7 janvier à 9h afin de faire une brève présentation et prévoir le planning de la semaine. Votre présence ce jour-là nous parait indispensable pour vérifier la bonne compréhension du sujet, poser des questions, organiser le travail en groupe...

La validation du projet est prévu vendredi matin. La présence est obligatoire.

5.3 Des ressources

Vous avez à votre disposition les sites internet de java, de logisim et d'autres sites d'aide au développement. Votre enseignant vous proposera une organisation afin de pouvoir le rencontrer régulièrement. Vous pouvez également utiliser le forum de moodle.

5.4 Rappel des groupes

Groupe 1.1	Ileana Ober	Groupe 1.2	Ileana Ober
Groupe 2.1	El Arbi Aboussoror	Groupe 2.2	El Arbi Aboussoror
Groupe 3.1	Ileana Ober	Groupe 3.2	Sergei Soloviev

6 Evaluation

La note du module est composée : d'une note sur le travail Ocaml, d'une note sur le travail Java (rapport inclus). La note de chaque partie prend en compte non seulement la réalisation des programmes demandés, mais aussi la qualité de la programmation (commentaires, ocaml ou javadoc, efficacité des calculs, solutions proposées, propositions supplémentaires...). Il vous est demandé un rapport de synthèse sur le travail Java.

7 Plagiat

Toute fraude, même partielle sera sanctionnée par des notes de groupe ET individuelle nulles (et éventuellement par un recourt au conseil de discipline). En cas de travail collectif entre différents groupes de projet, les points des portions collectives pourront être partagés ou attribués entièrement SI le travail est réellement collectif et uniquement si cela a été signalé aux encadrants du projet. Rappel : il est illégal d'utiliser des sources sans en citer les auteurs.