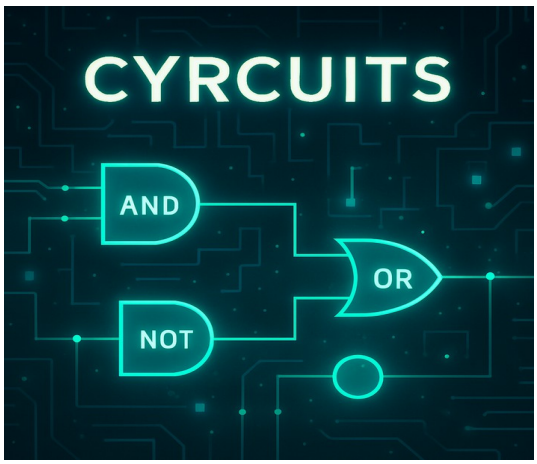


v1.0 Projet Cyrcuits

FILIERE ING1-GI • 2024-2025
AUTEURS E.ANSERMIN – R. GRIGNON
E-MAILS eva.ansermin@cyu.fr – romuald.grignon@cyu.fr

DESCRIPTION DU PROJET

- Le but de ce projet est de créer une application de conception de circuits logiques.
  Elle nécessite donc une vision du circuit, une simulation de ce dernier, et bien entendu la possibilité de relier et déplacer les différents éléments graphiques à l'écran.
- L'utilisateur sera capabloe de créer ses propres circuits, de les sauvegarder sur le disque dur, et de les réutiliser plus tard.
  Pour la simulation, il faut pouvoir modifier manuellement les différentes entrées et lancer une simulation automatique à partir de données pré-construites.
- Ce projet met l'accent sur la gestion des événements utilisateurs au niveau JavaFX.

INFORMATIONS GENERALES

- **Taille de l'équipe**
  Ce projet est un travail d'équipe. Il est autorisé de se réunir en groupe de 5 personnes. Si le nombre d'étudiants n'est pas un multiple de 5 et/ou si des étudiants n'arrivent pas à constituer des groupes, c'est aux chargés de projet de statuer sur la formation des groupes. Pensez-donc à anticiper la constitution de vos groupes pour éviter des décisions malheureuses.
- **Démarrage du projet et jalons**
  Vous obtiendrez de plus amples informations quant aux dates précises de rendu, de soutenance, les critères d'évaluation, le contenu du livrable, ..., quand le projet démarrera officiellement.
  Quel que soit le planning initial du projet, vous veillerez à planifier plusieurs rendez-vous d'avancement avec votre chargé(e) de projet. C'est à votre groupe de prendre cette initiative. L'idéal étant de faire un point 1 ou 2 fois par semaine mais cette fréquence est laissée libre en fonction des besoins identifiés avec le tuteur de projet.

- **Versions de l'application**
  Pour éviter d'avoir un projet non fonctionnel à la fin, il vous est demandé d'avoir une version en ligne de commande fonctionnelle. Ceci vous permettra de tester votre modèle de données indépendamment de l'interface graphique. C'est la version avec l'interface graphique JavaFX qui sera bien entendu évaluée mais dans le cas où certaines fonctionnalités ne seraient pas visibles avec cette interface, vous devez pouvoir présenter toutes les fonctionnalités de votre code Java en ligne de commande.
- **Dépôt de code**
  Vous devrez déposer la totalité des fichiers de votre projet sur un dépôt central Git. Il en existe plusieurs disponibles gratuitement sur des sites web comme ***github.com*** ou ***gitlab.com***. La fréquence des commits sur ce dépôt doit être au minimum de 1 commit / 2 jours.
- **Rapport du projet**
  Un rapport écrit est requis, contenant une brève description de l'équipe et du sujet. Il décrira les différents problèmes rencontrés, les solutions apportées et les résultats. L'idée principale est de montrer comment l'équipe s'est organisée, et quel était le flux de travail appliqué pour atteindre les objectifs du cahier des charges. Le rapport du projet peut être rédigé en français.
  Ce rapport contiendra en plus les éléments techniques suivants : un document de conception UML (diagramme de classe) de votre application, et un document montrant les cas d'utilisations.
- **Démonstration**
  Le jour de la présentation de votre projet, votre code sera exécuté sur la machine de votre chargé(e) de TD. La version utilisée sera la **dernière** fournie sur le dépôt Git **avant** la date de rendu. Même si vous avez une nouvelle version qui corrige des erreurs ou implémente de nouvelles fonctionnalités le jour de la démonstration, c'est bien la version du rendu qui sera utilisée.
  En parallèle, il vous faudra obligatoirement une deuxième machine avec votre application fonctionnelle car une partie de votre groupe aura des modifications de code à faire pendant que l'autre partie fera la présentation/démonstration de votre projet.
- **Organisation de l'équipe**
  Votre projet sera stocké sur un dépôt git (ou un outil similaire) tout au long du projet pour au moins trois raisons :
  - éviter de perdre du travail tout au long du développement de votre application
  - être capable de travailler en équipe efficacement
  - partager vos progrès de développement facilement avec votre chargé(e) de projet.

  De plus il est recommandé de mettre en place un environnement de travail en équipe en utilisant divers outils pour cela (Slack, Trello, Discord, ...).

CRITERES GENERAUX

- Le **but principal** du projet est de fournir une **application fonctionnelle** pour l'utilisateur. Le programme doit correspondre à la description en début de document et implémenter toutes les fonctionnalités listées.
- Votre code sera généreusement **commenté**.
- Tous les éléments de **votre code** (variables, fonctions, commentaires) seront écrits **en anglais** obligatoirement.
- Votre code devra être commenté de telle manière que l'on puisse utiliser un outil de génération de documentation automatique de type **JavaDoc**. Un dossier contenant la doc générée par vos soins sera présent dans votre dépôt de code lors de la livraison.
- Votre projet doit être utilisable au clavier ou à la souris en fonction des fonctionnalités nécessaires.
- Votre application ne doit jamais s'interrompre de manière **intempestive** (crash), ou tourner en boucle indéfiniment, quelle que soit la raison.
  Toutes les erreurs doivent être gérées correctement. Il est préférable de d'avoir une application stable avec moins de fonctionnalités plutôt qu'une application contenant toutes les exigences du cahier des charges mais qui plante trop souvent.
  Une application qui se stoppe de manière imprévue à cause d'une exception, par exemple, sera un événement très pénalisant.
- Votre application devra être **modulée** afin de ne pas avoir l'ensemble du code dans un seul et même fichier par exemple. Apportez du soin à la conception de votre projet avant de vous lancer dans le code.
- Le livrable fourni à votre chargé(e) de TD sera simplement l'**URL** de votre **dépôt Git** accessible **publiquement**.

FONCTIONNALITES DU PROJET

- Le projet doit pouvoir permettre d'afficher les différents éléments de base qui seront déjà disponibles pour l'utilisateur.
  - Entrées logiques
  - Entrées fixes (masse, alimentation, …)
  - Sorties logiques
  - Taille des bus de liaisons (nombre de bits pour chaque fil) et séparateurs/agrégateurs de bus.
  - Portes **AND**, **OR**, **NOT**, NOR, NAND, XOR, XNOR
    (il est conseillé de penser la conception des portes non-affichées en gras comme si elles étaient des circuits utilisateurs créés manuellement, de manière à prévoir la suite des fonctionnalités) *
  - Horloges
  - Afficheurs (base 10, base 16, …)

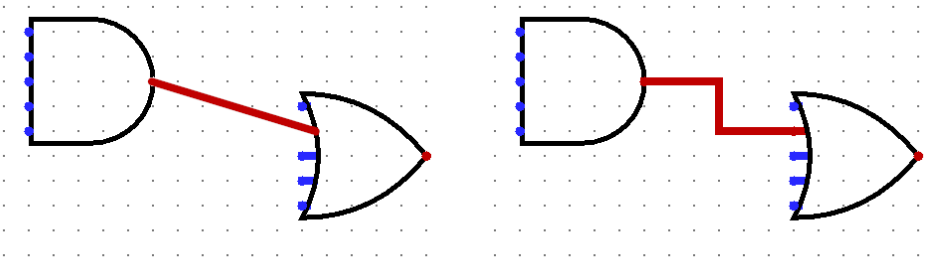
- Le programme doit permettre choisir des blocs logiques et les placer sur la fenêtre principale (soit un placement totalement analogique, soit un placement aligné sur une grille, configurable ou non)

- Le programme doit permettre de lancer une simulation du circuit : à vous de déterminer l'algorithme le plus astucieux : il existe plusieurs manières de faire, mais en fonction de votre choix de départ, il peut

s'avérer complexe de simuler certains circuits notamment des bascules ou des boucles combinatoires qui pourraient ne jamaius rester stables (ex : porte NOR avec une entrée à 0 et l'autre entrée reliés sur sa sortie).
Il faut donc penser à la propagation des signaux dans les fils (pas forcément de manière hyper réaliste mais au moins intégrer dans votre algorithme une notion du temps qui passe par pas)

- Vous pouvez fournir directement d'autres schémas plus complexes comme des bascules, des multiplexeurs, comparateurs, additionneurs, ou des registres (en ajoutant les classes d'objets nécessaires dans votyre programme) mais vous pouvez utiliser la fonctionnalité suivante pour créer vos propres circuits.
- Le programme doit pouvoir permettre de créer un circuit et le réutiliser plus tard (c'est dire récupérer le nombre d'entrées, de sorties, et d'afficher un simple rectangle graphiquement qui sera intégré à votre schéma complet). Il faut pouvoir bien entendu nommer vos entrées, sorties et circuits pour un affichage ergonomique.
  Cela implique de pouvoir sauvegarder restaurer des circuits utilisateurs dans des fichiers sur le disque dur (format libre).

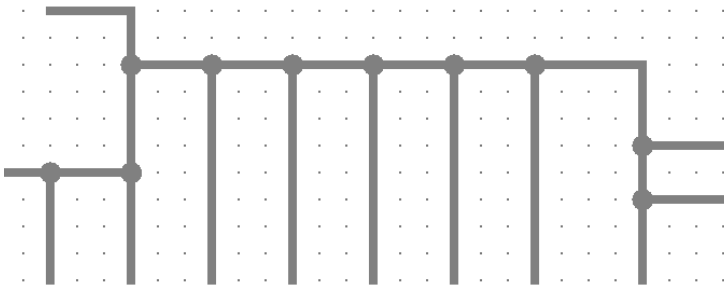
- Pour la liaison des points de contact, dans un premier temps faire une liaison directe (ce qui va créer des fils dans toutes les directions mais possède l'avantage d'être plus simple à réaliser), mais au final, un affichage aligné sur les axes sera plus harmonieux visuellement (on part du principe que les fils sortent orthogonalement des circuits et il faut deux angles droits pour former le 'S' typique de ce genre de logiciel).



- Pour la fusion de 2 fils, il faudra là-aussi gérer la conception de votre modèle : le modèle de 'réseau' de fils pourrait être une bonne solution pour représenter un ensemble de fils interconnectés comme étant un 'réseau' qui ne possède qu'une seule et unique valeur (sauf si un court-circuit survient).

  L'utilisateur doit donc pouvoir relier des entrées et des sorties les unes aux autres. Le programme pourra alors vérifier que 2 sorties ne sont pas en court-circuit. Vous pouvez gérez l'état 'flottant' (sortie à 3 états) pour éviter cette erreur. Dans tous les cas, si un court-circuit est détecté .

Graphiquement il faudra pouvoir ajouter un élément indiquant la connexion entre 2 fils dans un même réseau



- La dernière fonctionnalité importante est la simulation, du circuit : l'utilisateur doit pouvoir modifier les entrées manuellement, et/ou lancer une horloge afin de voir en direct les modifications sur les sorties.
  La fonctionnalité simulation doit pouvoir modifier les entrées à l'aide d'un fichier de simulatrion qui contiendrait les valeurs d'entrées en fonction du temps. Le format de ces fichiers est laissé libre : il faudra que l'utilisateur puisse accéder à une documentation pour pouvoir l'utiliser.

  Exemple :
  - Dans un circuit qui contient 3 entrées A, B, et C, un fichier pourrait contenir les lignes suivantes :
    A ; B ; C
    0 ; 1 ; 1
    0 ; 1 ; 0
    1 ; 1 ; 1
    0 ; 0 ; 0

  Les entrées correspondantes prendraient les valeurs successives indiquées dans le fichier avec un pas réglable au niveau du temps pour pouvoir visualiser les différentes étapes.
- Un paramètre 'boucle' doit pouvoir s'activer ou non, afin de faire tourner la simulation à l'infini, ou au contraire s'arrêter à la dernière ligne du fichier.

RESSOURCES UTILES

- **Github**
  https://www.github.com
  https://docs.github.com/en/get-started/quickstart/hello-world
- **Patrons de conception**
  https://fr.wikipedia.org/wiki/Patron_de_conception
- **Sérialisation**
  https://fr.wikipedia.org/wiki/Sérialisation