Examen Programmation Orientée Objet

1. Introduction

L'objectif de ce projet est de proposer un fichier exécutable (**LogicSim**) qui permet de simuler un circuit numérique constitué à partir de composants logiques combinatoires et séquentiels standards.
La description du circuit sera faite à partir d'un fichier **netlist** au format **Spice**. Pour chaque circuit, un fichier contenant les valeurs des entrées sera fourni. L'exécutable recevra un paramètre : le nom du circuit (sans son extension) à simuler. Les stimulus d'entrée seront lus à partir de fichiers stimulus (.**stm**) fournis. Il générera en sortie des fichiers probes (.**prb**) contenant les états des sorties. En fin de simulation, le résultat de la simulation sera affiché dans un format ASCII compréhensible :

```
c:\...\...\tp_info7_9_10>logicSim circuit_11

    IN_C : ___-__-_____________-_______-________-__-___
    IN_A : -__________-__-__-___-----______-__________
   OUT_B : ___-------_________--______-----___-------
   OUT_D : ---________--------__-----______---________

Process returned 0 (0x0)   execution time : 1.556 s
Press any key to continue.
```

2. Éléments constitutifs

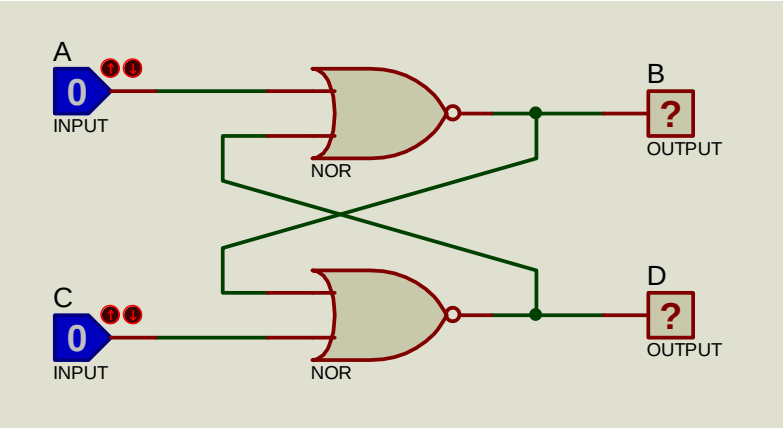
Les éléments constitutifs sont :

- La description du circuit,
- Les composants,
- Les équipotentielles (net),
- Les niveaux logiques,
- Les stimulus d'entrée,
- Les fichiers de probes (sorties obtenues).

a) Description du circuit :

Le circuit est fourni sous la forme d'une **netlist**, par exemple au circuit suivant :

Schéma du **circuit_11**	Contenu du fichier **circuit_11.cir**

	`IN_A 1000 INPUT`
	`IN_C 1003 INPUT`
	`OUT_B 1001 OUTPUT`
	`OUT_D 1002 OUTPUT`
	`U16 1000 1002 1001 NOR`
	`U17 1001 1003 1002 NOR`

Le format d'une ligne est :

`U16`	`1000`	`1002`	`1001`	`NOR`
Identifiant du composant	Net1	Net2	Net3	Modèle du composant.

Le fichier circuit fourni ne comportera pas d'erreurs.

b) Les composants

Le fonctionnement des composants sera intégré à votre programme à partir de sa table de vérité.

Seuls des composants standards seront utilisés.

- Buffer,
- fonction NOT,
- fonction AND,
- fonction NAND,
- fonction NOR,

Chaque composant aura une fonction **`propagation`** qui calculera les nouvelles valeurs des sorties en fonction des valeurs de ses entrées et du temps de propagation du circuit (fixé à 1 unité de temps).

c) Les équipotentielles (Net)

Les équipotentielles sont définies par leur valeur et les connexions réalisées.

d) Les niveaux logiques

Les niveaux logiques 0,1 et U seront définis. Le niveau U (undefined) sera le niveau d'initialisation d'une équipotentielle.

La valeur U sera propagée, sauf si elle peut être résolue.

Dans le cas de la fonction logique ET :

0 et U → 0,
1 et U → U
1 et 1 → 1
1 et 0 → 0

Par exemple

- Not(1)=0
- Not(U)=U
- AND(1,1)=1
- AND(1,U)=U
- AND(0,U)=0
- NOR(1,U)=0
- NOR(0,U)=U
- ...

e) Les stimulus d'entrée

Lors de la création ou du chargement du circuit, chaque entrée définira les stimulus appliqués au circuit. La succession d'états logiques est définie dans une chaîne de caractère qui est lue dans un fichier.
Chaque caractère encode un niveau logique, les temps d'applications sont de 1000 unités de temps de simulation pour chaque symbole.
Le nom du fichier est :
`nom_du_circuit_nom_de_l_entrée.stm`

Cette chaîne contient une seule ligne constituée de caractère '_' pour un 0 et '-' pour un 1.

```
'__--_-__---'
'00110100111'
```

Temps	Valeur
0	0
1000	0
2000	1
3000	1
4000	0
5000	1
6000	0
7000	0
8000	1
9000	1
10000	1

f) Les sondes des sorties

Chaque composant de sortie contiendra une chaîne de caractère. En fin d'un pas de simulation des entrées sorties l'état actuel de la sortie (dans le même format qu'une entrée) sera ajouté à cette chaîne.
En fin de simulation le contenu de la chaîne sera affiché et sauvé dans un fichier ayant comme nom : **`nom_du_circuit_nom_de_la_sortie.prb`**

g) Les évènements

Un évènement est constitué d'un temps, d'une équipotentielle et d'une valeur à imposer à l'équipotentielle. C'est l'élément de base du moteur de simulation.

3. Fonctionnement du simulateur

Le simulateur se base sur le principe de pile d'évènements ordonnée.
À chaque instant de simulation, on ajoute à la pile de simulation les modifications apportées aux équipotentielles par les modifications des entrées des composants. Ces modifications sont des évènements à venir. Lorsque la pile est vide, un état stable a été atteint.
Cette pile est traitée de façon chronologique.

Le simulateur utilisera deux piles d'évènements, la **pile d'évènements d'entrée/sortie** correspondant aux évènements externes (entrées / sorties). Une **pile d'évènements de simulation** correspondant aux évènements générés lors d'un pas de simulation.

a) Pseudo-code

Voici une proposition de pseudo-code du fonctionnement du simulateur :

- Lire le circuit
  - Lire les chaînes de caractères correspondant aux valeurs des entrées
- Tant que la **pile d'évènements d'entrée/sortie** n'est pas vide
  - Trier la **pile des évènements d'entrée/sortie** par valeur temporelle croissante.
  - Placer dans **pile d'évènements de simulation** l'ensemble des évènements en tête de la **pile d'évènements d'entrée sortie** ayant la même référence temporelle.
    - Tant que la **pile d'évènements de simulation** n'est pas vide :
      - Modifier la valeur des nœuds pour l'ensemble des premiers évènements ayant la même référence temporelle.
      - Appeler la fonction de **propagation** des composants
        - Ajouter à la **pile d'évènements de simulation** les évènements correspondant à la modification des sorties. Ces évènements auront comme référence temporelle le temps du pas de simulation incrémenté du temps de propagation du composant (qui sera toujours égal à 1).
  - Mémoriser dans les sorties leur valeur actuelle
- Sauvegarder pour chaque nœud de sortie la chaîne de caractère correspondant aux états atteints lors de chaque fin de pas de simulation.
- Afficher les chaînes de caractères des composants d'entrée sortie.

4. Identification des fichiers

Le fichier contenant la **netlist** du circuit aura l'extension .**cir**
Les fichiers contenant les stimulus d'entrée auront comme nom :

- **nom_du_circuit_nom_entree.stm**

Les fichiers contenant les résultats auront comme nom :

- **nom_du_circuit_nom_sortie.prb**

Les fichiers de références auront comme nom :

- **nom_du_circuit_nom_sortie_ref.prb**

5. Mise en œuvre

Pour chaque circuit fourni, votre programme devra en réaliser la simulation et afficher l'évolution des entrées et des sorties en fin de simulation.
Pour chaque sortie un fichier de sortie sera généré.
Afin de valider votre programme, des fichiers de référence des sorties peuvent être fournis.
Vous pouvez valider le fonctionnement de votre simulateur en vérifiant qu'il n'y a pas de différences entre vos résultats et les fichiers de référence.
Le programme s'exécutera dans la console. Il ne prendra qu'un seul paramètre le nom du circuit.
En fin d'exécution il affichera les valeurs des entrées et des sorties obtenues au format suivant :

```
   IN_C : ___-__-_____________-_______-________-__-___
   IN_A : -__________-__-__-___-----_____-__________
  OUT_B : ___-------_________--_____-----___-------
  OUT_D : ---________--------__-----_____---________
```

6. Suivi du développement

Un projet sur la forge de l'université sera créé pour votre groupe. La collaboration dans l'équipe doit se faire exclusivement en utilisant cet outil. Une partie de la note correspondra à une utilisation rationnelle et pertinente du SVN et des demandes en particulier.

7. Validation

Le dépôt de votre projet devra contenir une version fonctionnelle du code source ainsi qu'un fichier CMakeLists.txt permettant la génération de l’exécutable.

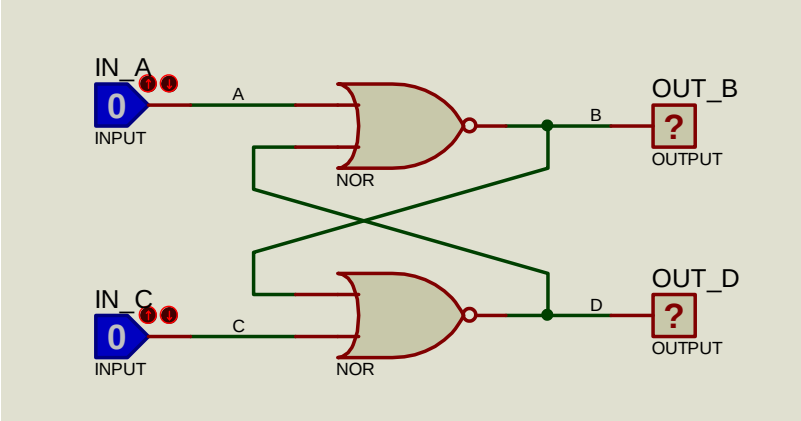
Vous pouvez tester le fonctionnement de votre programme avec le script test. Celui-ci teste votre programme avec un ensemble de circuits fournis.

Lors de l'évaluation, une série plus importante de fichiers sera utilisée.

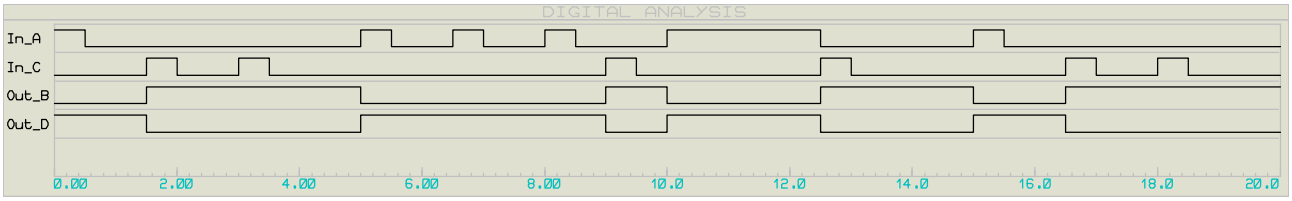
Le programme devra contenir une documentation sur son architecture et son fonctionnement interne.

8. Exemple

a) circuit_11 bascule RS



b) Résultat de la simulation Proteus



c) Netliste spice :

IN_A 1000 INPUT
IN_C 1003 INPUT
OUT_B 1001 OUTPUT
OUT_D 1002 OUTPUT
U16 1000 1002 1001 NOR
U17 1001 1003 1002 NOR

d) Contenu des fichiers de stimulus :

circuit_11_IN_A.stm : -_____-__-__-____-----_____-_____
circuit_11_IN_C.stm :____-__-_____-_____-_____-__-____

e) Fichiers résultats de la simulation :

circuit_11_OUT_B.prb :___-------_____--_____-----___-------
circuit_11_OUT_D.prb :---_____--------__-----_____---_____

f) Fichiers des sorties de références :

circuit_11_OUT_B_ref.prb :___-------_________--_____-----___-------
circuit_11_OUT_D_ref.prb :---_______--------__-----_____---_______

Le contenu de ces fichiers doit être identique à ceux générés par le simulateur. Une comparaison automatique peut être effectuée.

g) Le détail de la simulation obtenue :

Journal d'exécution	**Commentaires**
IN_A 1000 INPUT IN_C 1003 INPUT	Lecture de la Netlist
OUT_B 1001 OUTPUT OUT_D 1002 OUTPUT U16 1000 1002 1001 NOR U17 1001 1003 1002 NOR	Détail du circuit
Event(0,Net(1003,_U),_0) Event(0,Net(1000,_U),_1) Event(1000,Net(1003,_U),_0) Event(1000,Net(1000,_U),_0) Event(2000,Net(1003,_U),_0) Event(2000,Net(1000,_U),_0) Event(3000,Net(1000,_U),_0) Event(3000,Net(1003,_U),_1) Event(4000,Net(1000,_U),_0) Event(4000,Net(1003,_U),_0) Event(5000,Net(1003,_U),_0) Event(5000,Net(1000,_U),_0) Event(6000,Net(1003,_U),_1) ...	Liste des évènements d'entrée sortie
################ 0 ################	Première itération de la boucle de simulation
---------- Events 0---------- Event(0,Net(1003,_U),_0) Event(0,Net(1000,_U),_1)	Contenu de la pile de simulations. Ce sont les premiers éléments de la pile d'entrées sortie.
---------- Propag ---------- U16(NOR,_1_U=>_0)	Propagation de l'état des équipotentielles. Event (0) sur Net 1000 impose un niveau _1, donc la sortie de la Nor (Net 1001) passe à _0 à l'instant 1.
---------- Events 1---------- Event(1,Net(1001,_U),_0)	évènements générés par la propagation L'évènement a été créé par le pas de propagation précédent.
---------- Propag ---------- U17(NOR,_0_0=>_1)	Propagation de l'état des équipotentielles. Les nouvelles valeurs sont appliquées aux composants,
---------- Events 2---------- Event(2,Net(1002,_U),_1)	évènements générés pas la propagation. Cet évènement n'entraîne pas de nouvelles transitions, le pas de simulation est terminé.
################ 1000 ################	Les évènements suivants dans la pile d'entrée sortie sont placés dans la pile de simulation.
---------- Events 1000---------- Event(1000,Net(1003,_0),_0) Event(1000,Net(1000,_1),_0)	Contenu de la pile de simulations. Ce sont les éléments suivants dans la pile d'entrée sortie.
---------- Propag ----------	La propagation n'a généré aucun nouvel évènement. Le pas de simulation est terminé.

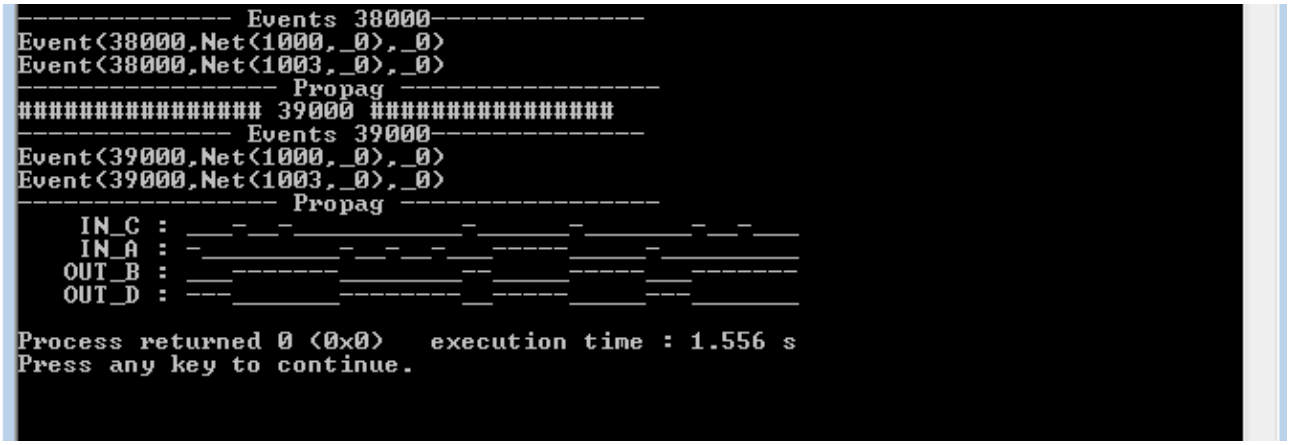
`################ 2000 ################`	Les évènements suivants dans la pile d'entrée sortie sont placés dans la pile de simulation.

`-------------- Events 2000--------------`
`Event(2000,Net(1003,_0),_0)`
`Event(2000,Net(1000,_0),_0)`
	Contenu de la pile de simulations. Ce sont les éléments suivants dans la pile d'entrée sortie.

`----------------- Propag -----------------`	La propagation n'a généré aucun nouvel évènement. Le pas de simulation est terminé.
`################ 3000 ################`	Les évènements suivants dans la pile d'entrée sortie sont placés dans la pile de simulation.

`-------------- Events 3000--------------`
`Event(3000,Net(1000,_0),_0)`
`Event(3000,Net(1003,_0),_1)`
	Contenu de la pile de simulations. Ce sont les éléments suivants dans la pile d'entrée sortie.

`----------------- Propag -----------------` `U17(NOR,_0_1=>_0)`	Propagation de l'état des équipotentielles.
`-------------- Events 3001--------------` `Event(3001,Net(1002,_1),_0)`	évènements générés pas la propagation
`----------------- Propag -----------------` `U16(NOR,_0_0=>_1)`	Propagation de l'état des équipotentielles.
`-------------- Events 3002--------------` `Event(3002,Net(1001,_0),_1)`	évènement générés pas la propagation
`----------------- Propag -----------------`	La propagation n'a généré aucun nouvel évènement. Le pas de simulation est terminé.
`################ 4000 ################` ……	



Exemple circuit NON :

Circuit_04.cir
IN_A 1000 INPUT
OUT_B 1001 OUTPUT
U1 1000 1001 NOT

circuit_04_IN_A.stm :___--_---_-_-_-__--_
circuit_04_OUT_B.prb :---__-___-_-_-_--_-

```
IN_A 1000 INPUT
OUT_B 1001 OUTPUT
U1 1000 1001 NOT

Event(0,Net(1000,_U),_0)
Event(1000,Net(1000,_U),_0)
Event(2000,Net(1000,_U),_0)
Event(3000,Net(1000,_U),_1)
Event(4000,Net(1000,_U),_1)
Event(5000,Net(1000,_U),_0)
Event(6000,Net(1000,_U),_1)
Event(7000,Net(1000,_U),_1)
Event(8000,Net(1000,_U),_1)
Event(9000,Net(1000,_U),_0)
Event(10000,Net(1000,_U),_1)
Event(11000,Net(1000,_U),_0)
Event(12000,Net(1000,_U),_1)
Event(13000,Net(1000,_U),_0)
Event(14000,Net(1000,_U),_1)
Event(15000,Net(1000,_U),_0)
Event(16000,Net(1000,_U),_0)
Event(17000,Net(1000,_U),_1)
Event(18000,Net(1000,_U),_1)
Event(19000,Net(1000,_U),_0)
################ 0 ################
------------- Events 0-------------
Event(0,Net(1000,_U),_0)
---------------- Propag ----------------
U1(NOT,_0=>_1)
------------- Events 1-------------
Event(1,Net(1001,_U),_1)
---------------- Propag ----------------
################ 1000 ################
------------- Events 1000-------------
Event(1000,Net(1000,_0),_0)
---------------- Propag ----------------
################ 2000 ################
------------- Events 2000-------------
Event(2000,Net(1000,_0),_0)
---------------- Propag ----------------
################ 3000 ################
------------- Events 3000-------------
Event(3000,Net(1000,_0),_1)
---------------- Propag ----------------
U1(NOT,_1=>_0)
------------- Events 3001-------------
Event(3001,Net(1001,_1),_0)
---------------- Propag ----------------
################ 4000 ################
------------- Events 4000-------------
Event(4000,Net(1000,_1),_1)
---------------- Propag ----------------
################ 5000 ################
------------- Events 5000-------------
Event(5000,Net(1000,_1),_0)
---------------- Propag ----------------
U1(NOT,_0=>_1)
------------- Events 5001-------------
Event(5001,Net(1001,_0),_1)
---------------- Propag ----------------
################ 6000 ################
------------- Events 6000-------------
Event(6000,Net(1000,_0),_1)
---------------- Propag ----------------
U1(NOT,_1=>_0)
------------- Events 6001-------------
Event(6001,Net(1001,_1),_0)
---------------- Propag ----------------
################ 7000 ################
------------- Events 7000-------------
Event(7000,Net(1000,_1),_1)
---------------- Propag ----------------
################ 8000 ################
------------- Events 8000-------------
Event(8000,Net(1000,_1),_1)
---------------- Propag ----------------
################ 9000 ################
------------- Events 9000-------------
Event(9000,Net(1000,_1),_0)
---------------- Propag ----------------
U1(NOT,_0=>_1)
------------- Events 9001-------------
Event(9001,Net(1001,_0),_1)
---------------- Propag ----------------
################ 10000 ################
------------- Events 10000-------------
Event(10000,Net(1000,_0),_1)
---------------- Propag ----------------
U1(NOT,_1=>_0)
------------- Events 10001-------------
Event(10001,Net(1001,_1),_0)
---------------- Propag ----------------
################ 11000 ################
------------- Events 11000-------------
Event(11000,Net(1000,_1),_0)
---------------- Propag ----------------
U1(NOT,_0=>_1)
------------- Events 11001-------------
Event(11001,Net(1001,_0),_1)
---------------- Propag ----------------
################ 12000 ################
------------- Events 12000-------------
Event(12000,Net(1000,_0),_1)
---------------- Propag ----------------
U1(NOT,_1=>_0)
------------- Events 12001-------------
Event(12001,Net(1001,_1),_0)
---------------- Propag ----------------
################ 13000 ################
------------- Events 13000-------------
Event(13000,Net(1000,_1),_0)
---------------- Propag ----------------
U1(NOT,_0=>_1)
------------- Events 13001-------------
Event(13001,Net(1001,_0),_1)
---------------- Propag ----------------
################ 14000 ################
------------- Events 14000-------------
Event(14000,Net(1000,_0),_1)
---------------- Propag ----------------
U1(NOT,_1=>_0)
------------- Events 14001-------------
Event(14001,Net(1001,_1),_0)
---------------- Propag ----------------
################ 15000 ################
------------- Events 15000-------------
Event(15000,Net(1000,_1),_0)
---------------- Propag ----------------
U1(NOT,_0=>_1)
------------- Events 15001-------------
Event(15001,Net(1001,_0),_1)
---------------- Propag ----------------
################ 16000 ################
------------- Events 16000-------------
Event(16000,Net(1000,_0),_0)
---------------- Propag ----------------
################ 17000 ################
------------- Events 17000-------------
Event(17000,Net(1000,_0),_1)
---------------- Propag ----------------
U1(NOT,_1=>_0)
------------- Events 17001-------------
Event(17001,Net(1001,_1),_0)
---------------- Propag ----------------
################ 18000 ################
------------- Events 18000-------------
Event(18000,Net(1000,_1),_1)
---------------- Propag ----------------
################ 19000 ################
------------- Events 19000-------------
Event(19000,Net(1000,_1),_0)
---------------- Propag ----------------
U1(NOT,_0=>_1)
------------- Events 19001-------------
Event(19001,Net(1001,_0),_1)
---------------- Propag ----------------
    IN_A : ___--_---_-_-_-__--_
   OUT_B : ---__-___-_-_-_--__-

Process returned 0 (0x0)   execution time : 0.254 s
Press any key to continue.
```