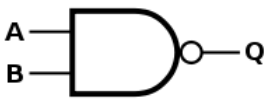
TD Concepts objet et Java

2016-17

Exercice 4

Les portes logiques sont des circuits électroniques qui possèdent des entrées et des sorties sur lesquelles on place et récupère des valeurs de bits. Les portes logiques [1] ET-NON (*nand*) possèdent deux entrées A et B. La sortie Q est au niveau 0 (ou *false*) si toutes les entrées sont au niveau 1 (ou *true*). Une seule entrée au niveau 0 suffit pour que la sortie soit à 1. Les portes logiques OU-NON (*nor*), qui possèdent également deux entrées A et B, ont leur sortie est au niveau 1 si aucune des entrées n'est au niveau 0. Une seule entrée au niveau 1 suffit pour que la sortie soit à 0.

Les symboles de ces deux portes sont représentés ci-dessous :

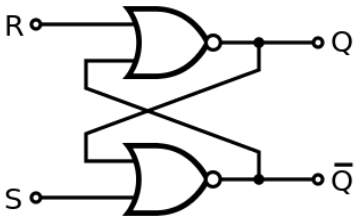


porte *nand*

porte *nor*

Les circuits séquentiels sont obtenus par combinaison de portes logiques, avec certaines sorties qui sont rebouclées en entrée. Contrairement aux portes logiques, l'état des sorties des circuits séquentiels dépend non seulement de l'état des entrées mais aussi de l'état antérieur des sorties : ce sont des circuits dotés de mémoire.

Les bascules RS, représentées ci-dessous, sont des *bistables* qui peuvent prendre deux états. Ces bascules sont asynchrones. Elles possèdent deux entrées nommées R et S et deux sorties complémentaires nommées Q et $\bar{Q}$. L'entrée S (*set*) met la bascule au travail et la sortie Q à la valeur 1. L'entrée R (*reset*) remet la bascule au repos et la sortie Q à la valeur 0.



Bascule RS avec portes NOR

1. Voir par exemple : `https://fr.wikibooks.org/wiki/Fonctionnement_d\%27un_ordinateur/Les_circuits_combinatoires`

La bascule RS ci-dessus est réalisée avec des portes *nor*. On envoie séparément et alternativement les signaux sur S et R. Si $S = R = 1$, alors l'état est indéterminé. La table de vérité en fonction de l'état précédent Q_n est la suivante :

Q_n	S	R	Q_{n+1}
1	1	0	1
1	0	1	0
1	1	1	?
1	0	0	1
0	1	0	1
0	0	1	0
0	0	0	0
0	1	1	?

1. On souhaite modéliser des portes *nand* et *nor* par des classes en factorisant au mieux le code. Toute modification des entrées doit entraîner la mise à jour de la sortie. Écrivez les classes nécessaires en les dotant de méthodes utiles pour l'affichage de l'état de leurs objets.
2. Mettez en œuvre le framework de test `JUnit` dans l'IDE Eclipse et écrivez des tests pour vos deux types de portes.
3. Écrivez à présent une classe modélisant une bascule RS composée de 2 portes *nor*, et implémentant les méthodes :
   — `getS`, `getR`, `getQ`, `getNonQ` pour obtenir l'état des valeurs correspondantes ;
   — les méthodes `setS` et `setR` pour changer l'entrée de la bascule et calculer un nouvel état ;
   — une méthode pour afficher l'état de la bascule.
4. Testez également cette nouvelle classe au moyen de tests `JUnit`.
5. On souhaite à présent ajouter aux portes logiques un nombre de *cycles* maximum au bout duquel la porte est à changer. On comptera comme cycle toute opération de lecture ou d'écriture sur la porte logique. Ajoutez le code nécessaire pour prendre en compte ce nombre de cycles maximum, puis mettez en œuvre la notion d'*exception* pour prendre en compte des erreurs au niveau des portes logiques (ainsi que de leurs sous-types éventuels) ainsi que des erreurs au niveau d'une bascule RS.
6. Prévoyez une stratégie de récupération du type d'erreur au niveau de la classe représentant la bascule RS réalisant le remplacement d'une éventuelle porte logique défaillante. Implémentez également la possibilité de l'ajout d'un nombre de cycles maximaux à une bascule particulière.