Exercice 1

1. **Logiciel critique**: logiciel qui ne doit jamais tomber en panne, puisque sa défaillance mènerait à une perte de vies humaines ou pertes économiques insupportables.
2. La **validation par revue de code**: le programme est lu (ligne par ligne) par un collègue informaticien qui cherche systématiquement les erreurs.
3. **Hypothèse synchrone** : lecture des entrées (de capteurs), calculs et écriture des résultats (vers les actionneurs) sont suffisamment rapides pour se faire entre deux ticks d'horloge sans jamais déborder.
4. **Logiciel de tems réel** doit répondre à chaque requête d'environnement avec un délai prédéfini. Les retards ne sont pas acceptés.
5. **Causalité**: dans notre contexte signifie que chaque cycle dans le diagramme de flots contient un retard (PRE ou FBY). Ça permet de faire tous les calculs dans l'ordre., aucune variable ne dépend pas d'elle-même.
6. **Cycle de développement en Y**: on modélise le système et le logiciel, on valide le modèle en s'assurant qu'il est fidèle à la spécification, et on génère automatiquement le code. Ainsi les niveaux bas du cycle en V sont remplacés par une génération de code.

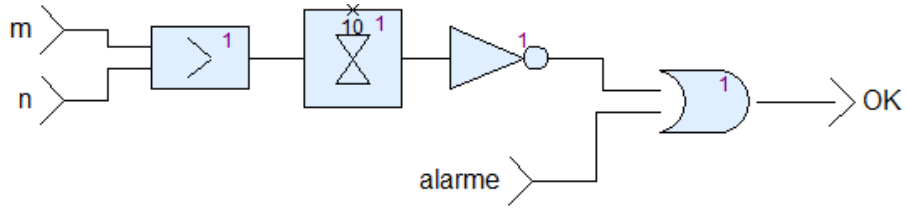
Exercice 2

1. Y=12,48, 240, 960, 2880, 5760,…
2. Le nœud B souffre d'un problème de causalité: il contient un cycle sans retard.

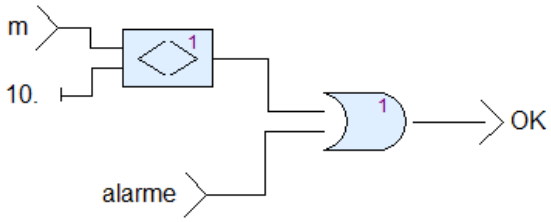
Exercice 3

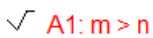
On se rappelle que $a \Longrightarrow b$ se récrit comme $\neg a \vee b$. Pour chacun des deux cas on fait un observateur de propriété, le connecte à l'opérateur toto et lance l'outil de preuve de SCADE.

1. L'observateur est comme ceci :

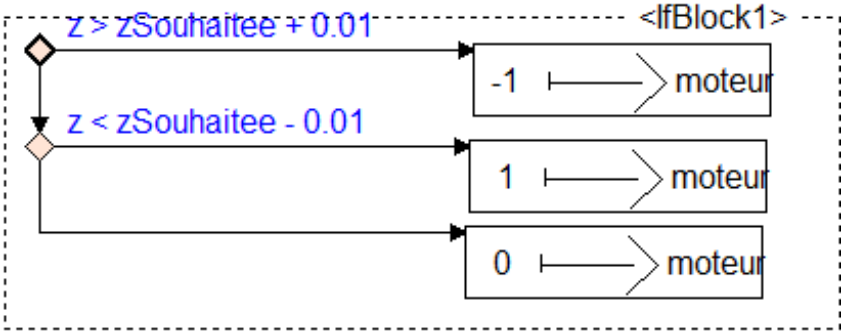


2. L'observateur est comme cela (l'hypothèse $m > n$ devient un **assume)**



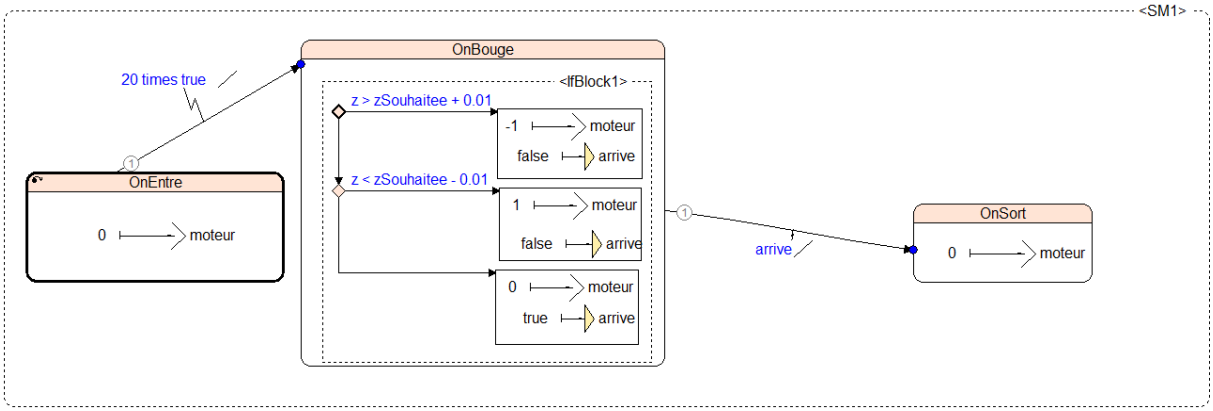


Exercice 4.1



(on a ajouté une petite tolérance par rapport au niveau souhaité)

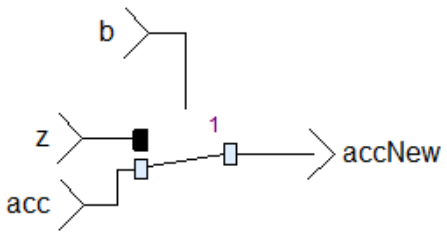
Exercice 4.2



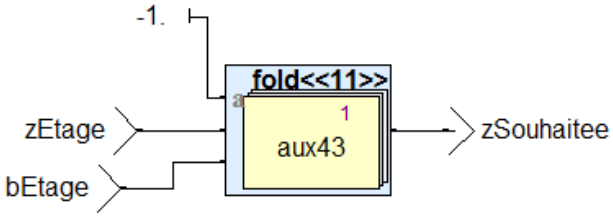
(on reste dans le troisième état pour une éternité)

Exercice 4.3

On fait un opérateur auxiliaire **aux43** comme ceci.



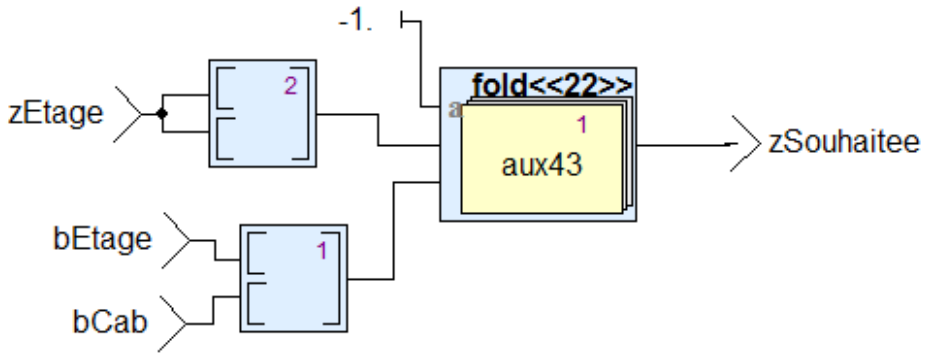
Et puis on fait le décodeur **decode43** comme cela :



(si aucun bouton n'est appuyé, l'opérateur renvoie -1, on l'utilisera en 4.5)

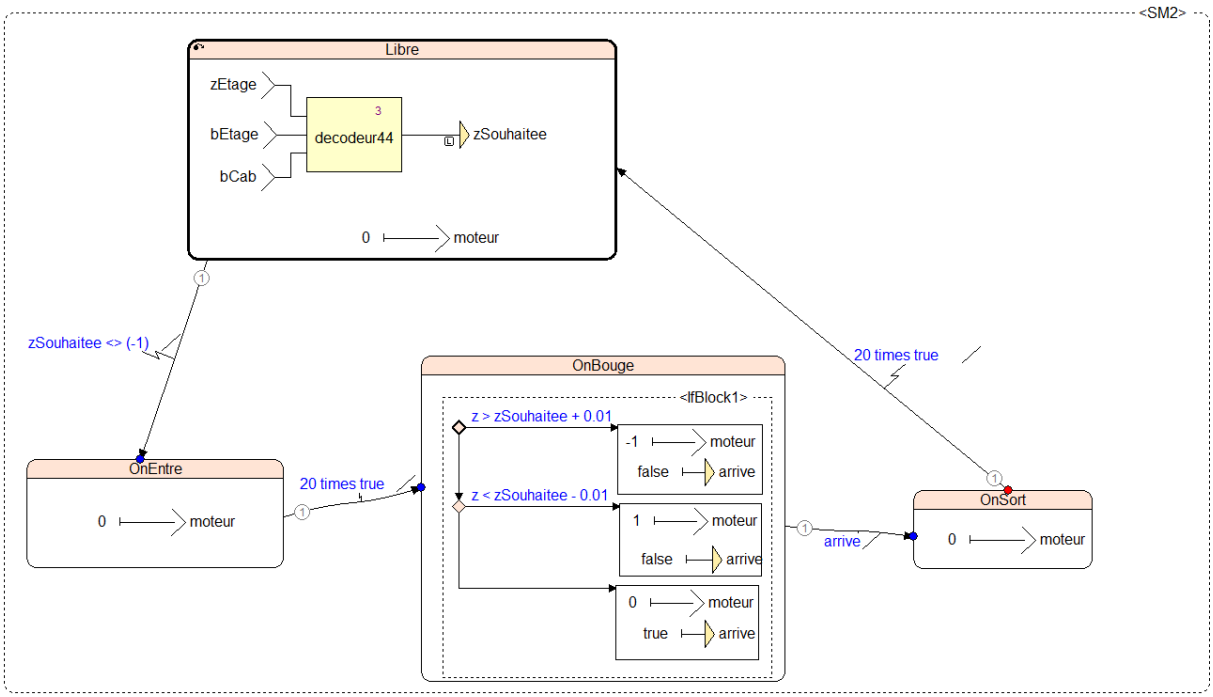
Exercice 4.4

On fait comme dans l'exercice précédent mais avec des tableaux 2 fois plus longs, qui incluent tous les boutons (et les niveaux correspondants), c'est l'opérateur **decode44**



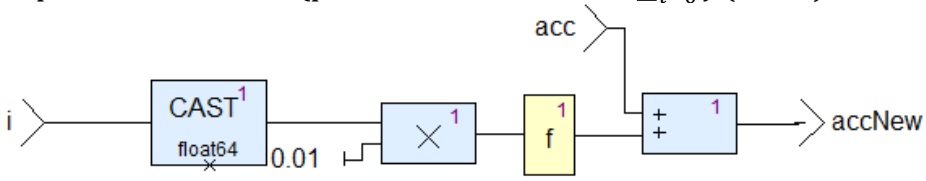
Exercice 4.5

On augmente un peu l'automate 4.2 en ajoutant l'état Libre

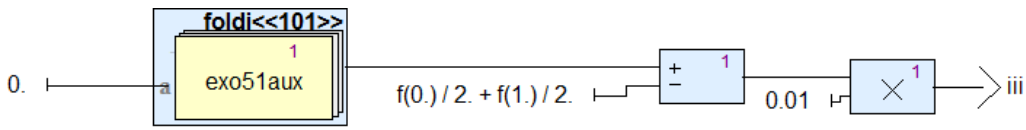


Exercice 5.1

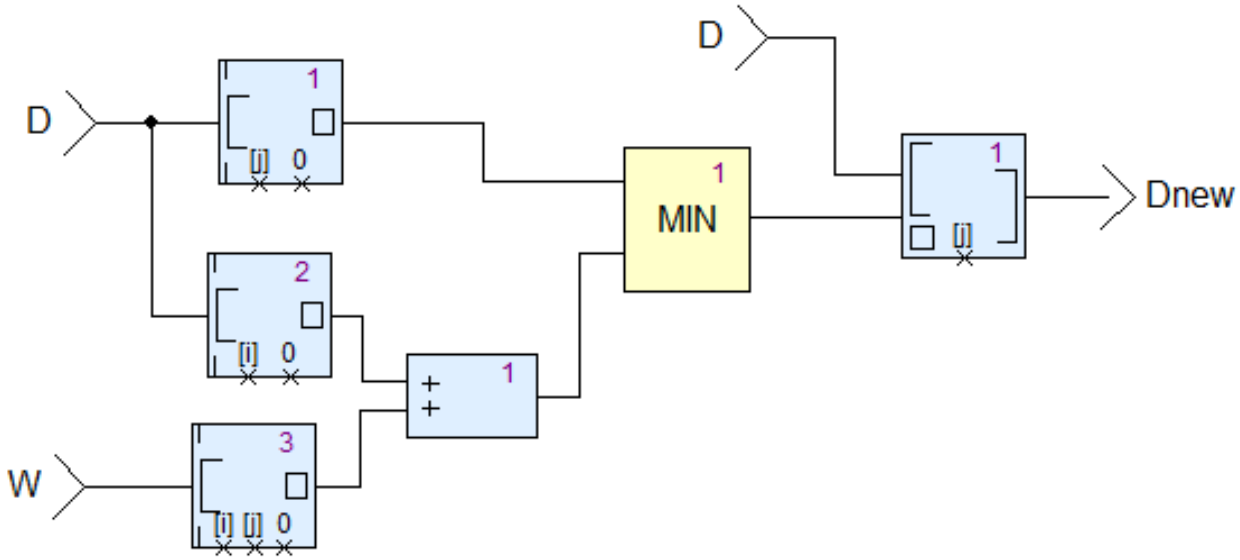
L'opérateur auxiliaire (pour calculer la somme $\sum_{i=0}^{100} f(0.01\ i)$ avec un foldi.)



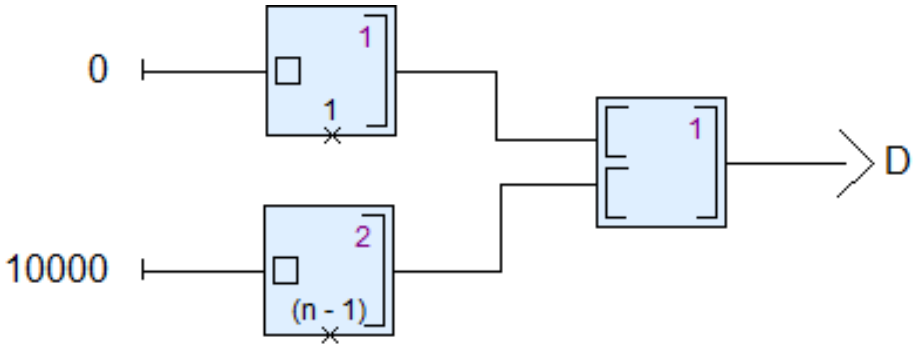
Et puis l'opérateur principal



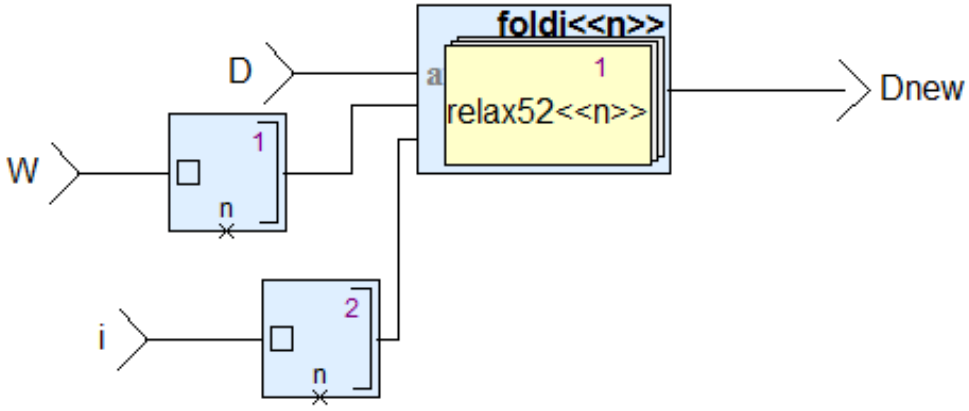
Exercice 5.2
L'opérateur générique **relax52** (avec i,j :int et les tableaux D, Dnew : int^n et W :int^n^n)



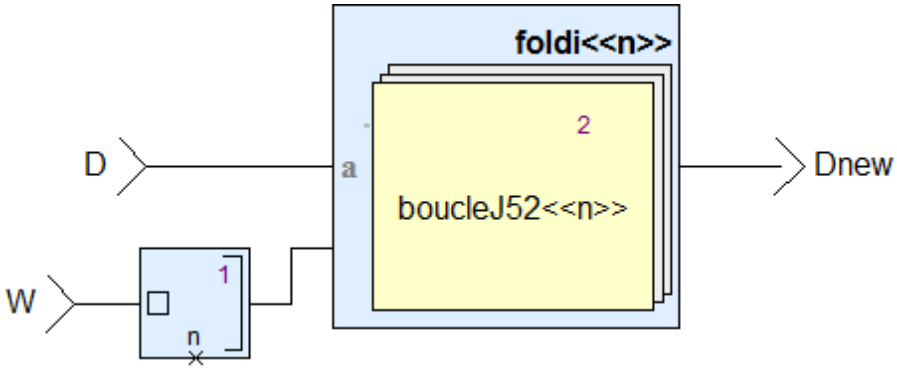
L'opérateur **init52** (au lieu d'∞ on a pris 10000)



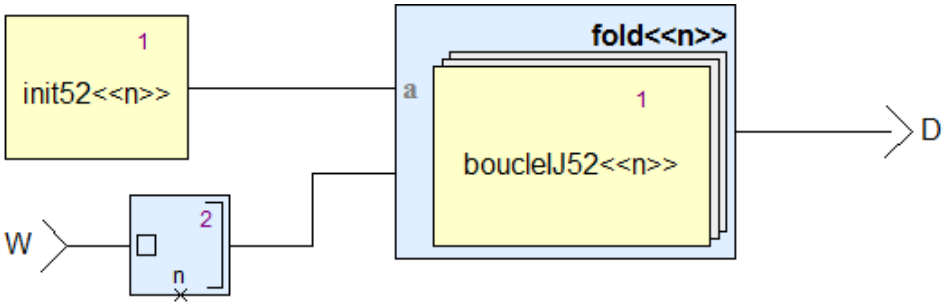
La boucle intérieure de Bellman-Ford : opérateur **boucleJ52**



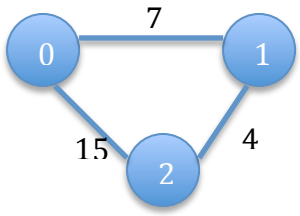
La boucle suivante : opérateur **boucleIJ52**



et finalement l'opérateur principal **BF52**:



Je l'ai testé sur un petit graphe de 3 sommets



avec l'opérateur de test suivant :

Comme attendu, j'ai obtenu le résultat D=[0,7,11]