

Programmation des systèmes réactifs

Lustre (Partie 2)

ESINSA - Option GSE

28 septembre 2004

1 Les opérateurs \rightarrow et **pre** de LUSTRE

1.1 Comportement de $n = 0 \rightarrow 1 \rightarrow 2$;

Quel est le comportement intuitif de la variable n ?

Écrire un programme Lustre permettant d'observer le comportement effectif de n .

1.2 Vérification de l'associativité de l'opérateur \rightarrow

Cas booléen

Écrire un programme Lustre permettant de vérifier (à l'aide de Lesar) que, si X, Y, Z sont des flots booléens on a la propriété (associativité) :

$$(X \rightarrow Y) \rightarrow Z = X \rightarrow (Y \rightarrow Z)$$

C'est à dire que l'on peut écrire $X \rightarrow Y \rightarrow Z$ sans ambiguïté.

Cas entier

Modifier le programme précédent pour travailler sur les flots entiers. Quel est le diagnostic de la vérification ? Pourquoi ?

Cas général ♠

En utilisant les équations de récurrence définissant l'opérateur \rightarrow établir la propriété d'associativité dans le cas général.

Avertissement : pour la suite retenir la propriété $X \rightarrow Y \rightarrow Z \equiv X \rightarrow Z$.

1.3 Usage combiné de \rightarrow et **pre**

Exemple

Programmer, simuler et justifier le comportement des flots suivants :

1. $X = A \rightarrow \text{pre}(X)$;
2. $X = A \rightarrow \text{pre}(B \rightarrow \text{pre}(X))$;

Les sources sont dans `src/ExWithPre.lus`.

Suite de Fibonacci ♠

Programmer un générateur de la suite de Fibonacci¹, définie par la récurrence

$$F_n = F_{n-1} + F_{n-2}$$

$$F_1 = 1$$

$$F_2 = 1$$

Nœuds T et F

On définit deux nœuds Lustre très utiles pour la programmation :

```

1  -- TF.lus
2  -- Charles André
3  -- September 17, 2004
4
5  -- Purpose: facilities for sequential intializations
6  --          let X = (x1,x2,...,xn,...) and Y = (y1,y2,...,yn,...)
7
8  -- Y = T(X) is such that y1 = true; yk = xk-1
9  node T(X:bool) returns (Y:bool);
10 let
11     Y = true -> pre(X);
12 tel
13
14 -- Y = F(X) is such that y1 = false; yk = xk-1
15 node F(X:bool) returns (Y:bool);
16 let
17     Y = false -> pre(X);
18 tel
19
20
21 node TF(X:bool) returns (Y1,Y2:bool);
22 let
23     Y1 = T(X);
24     Y2 = F(X);
25 tel
26
27 -- luciole TF.lus TF
```

Le fichier correspondant est src/TF.lus. Observer et justifier leur comportement.

Application: générateur d'horloge

En utilisant les nœuds T et F, construire un programme Lustre qui engendre la séquence `false;(false>true)*`.

2 Systèmes à temps discret

2.1 Réponse d'un système du premier ordre

Soit un système du premier ordre de constante de temps τ .

¹Son vrai nom est Leonardo Pisano.

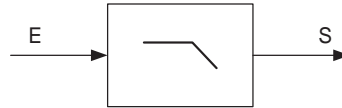


Figure 1: Système du premier ordre.

La réponse temporelle de ce système (lorsque E reste constant pendant Δt) est

$$S(t + \Delta t) = S(t) + (E(t) - S(t)) * e^{-\Delta t/\tau}$$

On pose $K = e^{-\Delta t/\tau}$. Écrire et simuler un programme Lustre qui calcule la sortie de ce système. Le prototype du nœud doit être

```
node FirstOrder(E,K:real) returns (S:real);
```

2.2 PWM

On veut réaliser en Lustre une modulation de largeur d'impulsions (PWM). Le signal d'entrée est X qui prend ses valeurs dans $[0.0..XM]$. La sortie du modulateur P est booléenne. Il y a N instants par périodes.

La figure 2 suggère une façon de construire le signal de sortie en utilisant un compteur entier cpt .

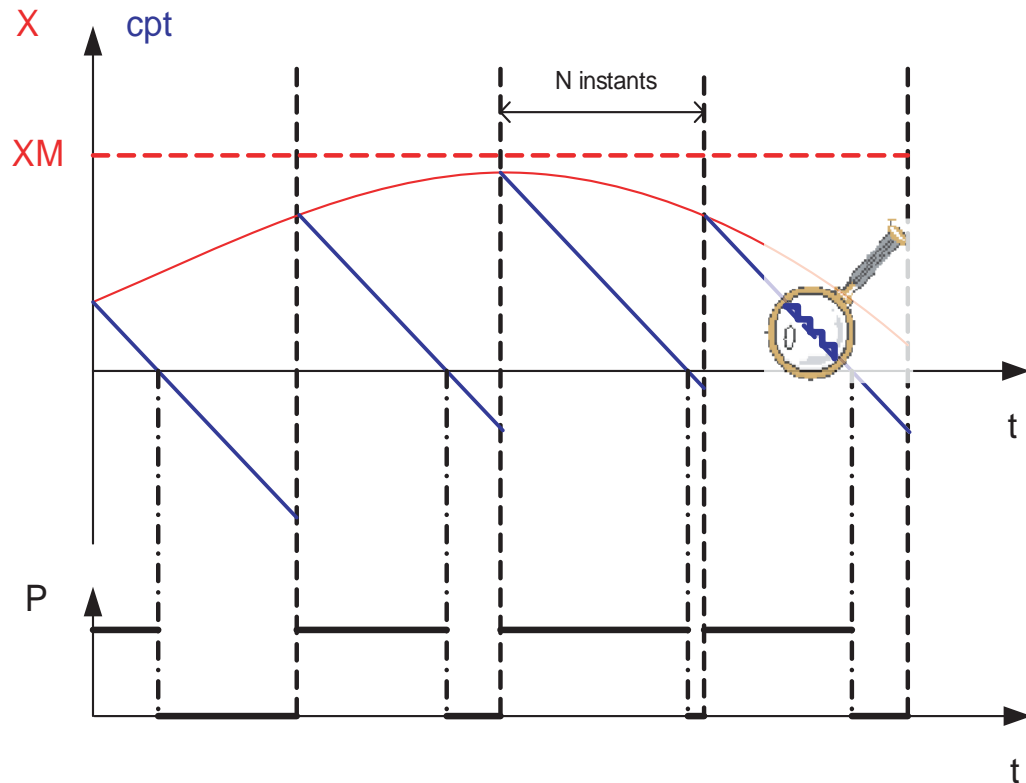


Figure 2: Système du premier ordre.

Programmer le nœud `node PWM(X:real) returns (P:bool);`.

Suggestions : utiliser les fonctions `int` et `real` pour passer des entiers aux réels et réciproquement. Utiliser également des constantes.

2.3 Génération d'un signal sinusoïdal

Un programme peut engendrer un signal sinusoïdal par consultation de table ou par calcul d'une équation de récurrence. Nous choisissons cette deuxième solution qui est mise en œuvre par un filtre transversal (IIR).

Soit F_s la fréquence d'échantillonnage, F la fréquence désirée (bien sûr F est au moins deux fois plus petite que F_s).

On pose $\theta = 2\pi \frac{F}{F_s}$.

L'équation de récurrence $y(n) = A * y(n-1) - y(n-2)$ produit les valeurs de $\sin(n\theta)$, si on prend $A = 2 \cos(\theta)$ et les valeurs initiales $y(0) = 0.0$ et $y(1) = \sin(\theta)$. On pose $C = \sin(\theta)$.

Programmer node SINUS(A,C:real) returns (S:real);

Suggestion : écrire un petit programme en C, Matlab, ... qui prend F_s et F et retourne les valeurs de A et C .