TP - Le Langage Lustre (avec correction)

Thao Dang

Exo 1

Ecrire un programme Lustre coresspondant au système suivant :

$$Y_1 = X_1/2 \tag{1}$$
$$Y_n = (X_n + X_{n-1})/2 \tag{2}$$

On peut compiler le programme avec **lus2ec** pour ensuite générer le code C. Le compilateur **lus2ec** produit un program qui contient le code du noeud principal et les appels de noeuds sont "inlined" et les variables (arrays, tuples) sont "étendues" en variables atomiques (bool, int, real ou types externes). On peut compiler le programme avec **lus2c** pour générer directment le code C.

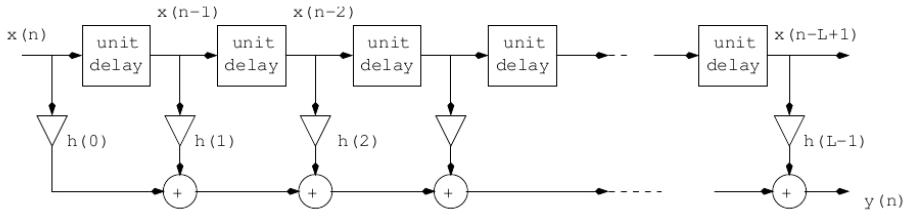
Solution :

```
node convolution(X:real) returns (Y:real);
let Y = (X + (0 -> pre X)) / 2.0;
tel;
```

Exo 2

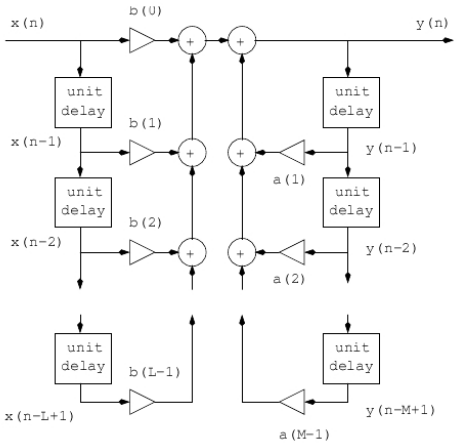
Filtres linéaires
– FIR (Finite Impulse Response)

$$y(n) = \sum_{m=0}^{L-1} x(n-m)b(m)$$



– IIR (Infinite Impulse Response) ou filtre récursif

$$y(n) = \sum_{m=0}^{L-1} x(n-m)b(m) + \sum_{m=1}^{M-1} y(n-m)a(m)$$



Exemple : $y(n) = x(n) + 0.9y(n-1)$. Ecrire un programme Lustre pour ce filtre.

Solution :

```
node filtre(x:real} return (y:real);

let
  y = x + (0.0 -> 0.9 * pre(y));
tel;
```

Exo 3. Pendule inversé

Les équations d'un pendule inversé sont

$$l\frac{d^2\theta}{dt^2} = sin(\theta)(\frac{d^2y_0}{dt^2} + g) - cos(\theta)\frac{d^2x_0}{dt^2} \quad (3)$$

$$x = x_0 + lsin(\theta) \quad (4)$$

$$y = y_0 + lcos(\theta) \quad (5)$$

Ecrire un programme Lustre pour ce pendule.

Solution : D'abord, on écrit un intégrateur de type Euler en avant pour $\frac{dx}{dt} = f(x)$ avec h comme le pas de temps

$$x(k+1) = x(k) + f(x(k))h$$

```
node integr(dt, x':real) returns (x:real);
  let x = 0.0 -> dt * x' + pre x;
  tel;

node deriv(dt, x:real) returns (dx:real);
  let dx = 0.0 -> (x - (pre x))/ dt;
  tel;
```

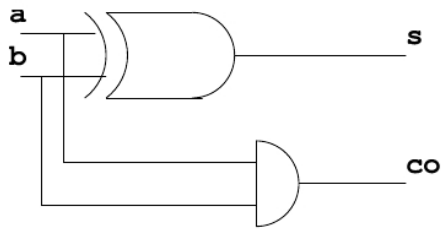
Module principal

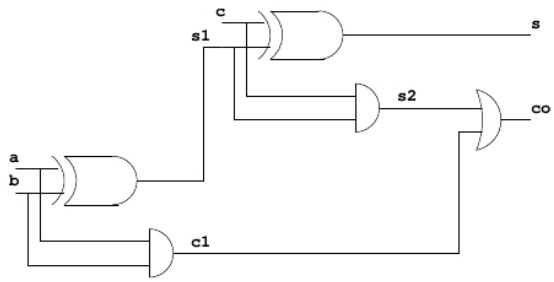
```
node pendulum(x0'', y0'':real) returns (theta:real);
 let theta =
        integr (dt, integr (dt, (sin thetap) * (y0'' + g)
                        - (cos thetap) * x0'') / l);
     thetap = 0.0 -> pre theta;
 tel;
```

Les constantes : dt = 0.001 (pas de temps), l = 10.0 (longueur), g = 9.81 (acceleration).

Exo 4. Vérification

On va essayer d'utiliser l'outil LESAR pour vérifier un programme LUSTRE qui correspond à un additioneur 1-bit. Taper la commande suivante : **"lesar prog.lus equivalence"** (si les programmes sont écrits dans dans un fichier nommé prog.lus).






```
node half_add(a,b:bool)
returns (s, co:bool);
  let s = a xor b;
      co = a and b;
  tel;

node full_add_h(a,b,c:bool)
returns (s, co:bool);
  var s1,c1,c2:bool;
  let
    (s1, c1) = half_add(a,b);
    (s, c2) = half_add(c, s1);
    co = c1 or c2;
  tel;


node full_add(a, b, c:bool) returns (s, co:bool);
  let
    s = (a xor b) xor c;
```

```
    co = (a and b) or (b and c) or (a and c);
  tel;


node equivalence(a,b,c:bool) returns (ok:bool);
  var o1, c1, o2, c2: bool;
  let
     (o1, c1) = full_add(a,b,c);
     (o2, c2) = full_add_h(a,b,c);
     ok = (o1 = o2) and (c1 = c2);
  tel;
```