TP - Le Langage Lustre

Thao Dang

8 mars 2010

Exo 1

Ecrire un programme Lustre coresspondant au système suivant :

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

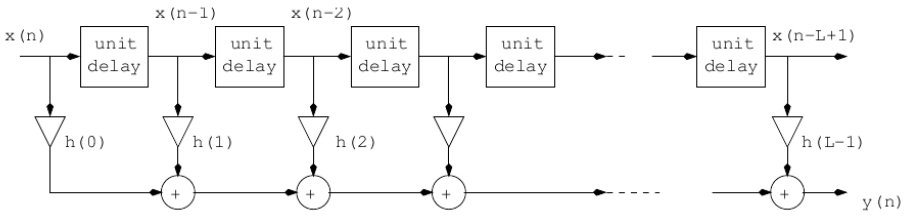
On compile le programme avec **lus2ec** pour générer le code C.

Exo 2

Filtres linéaires

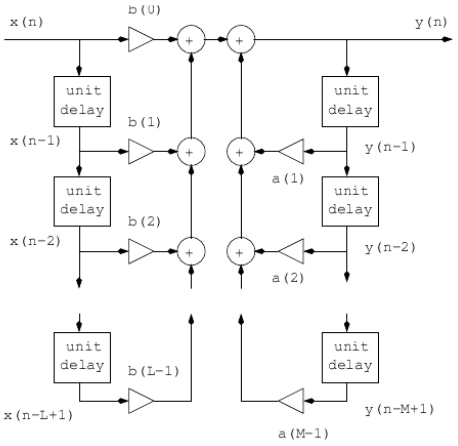
– FIR (Finite Impulse Response)

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



– 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.

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.

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;
```