Programmation des systèmes réactifs
Lustre (Partie 1)

EPU élec - Option GSE

26 septembre 2007

Avertissement : Les questions avec le symbole ♠ sont plus difficiles et ne seront pas nécessairement résolues en séance.

1 Découverte de l'environnement LUSTRE

Travailler sous LINUX.

1.1 Configuration

Modifier les variables d'environnement PATH et MANPATH pour accéder aux exécutables et aux manuels de Lustre.

1.2 Lancement de l'environnement Lustre

Taper (ou récupérer) le programme counter.lus qui se trouve dans le sous répertoire src du répertoire TD1.

```
1   -- counter.lus
2   -- Charles André
3   -- September 15, 2004
4
5   -- program given in hand-outs
6   -- purpose: explore Lustre environment
7
8   node Counter (init,incr:int; reset:bool) returns (c:int);
9   let
10          c = init -> if reset then init else pre(c)+incr;
11  tel
12
13  -- to execute, type:
14  -- luciole counter.lus Counter
```

Lancer sa compilation, suivie de son exécution par la commande :
`luciole counter.lus Counter`

1.3 Exploration des commandes

La fenêtre (Figure 1) apparaît.

- Explorer les commandes du menu.
- Lancer sim2chro (x11) du menu Tools.
- Simuler interactivement le programme.
- Essayer les modes auto step/compose. Intérêt?
- Observer le fonctionnement dit real-time clock.

Figure 1: Fenêtre principale.

Une copie d'écran de la fenêtre de sim2chro est donnée dans la figure 2.

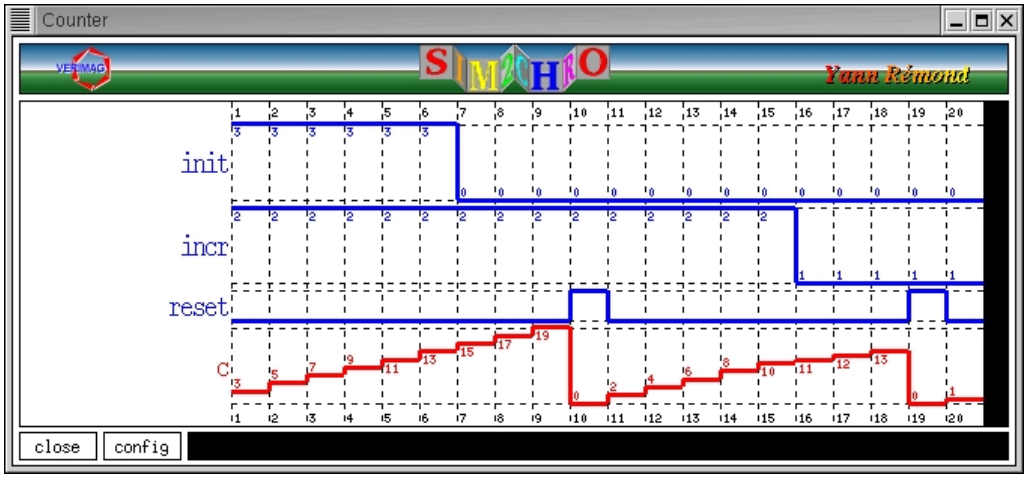

Figure 2: Exemple d'exécution de Counter.

1.4 Modification du programme

Écrire un programme Lustre qui prend en entrée un flot entier M et qui donne en sortie un flot entier C tel que

- Initialement C est à 0.
- A chaque réaction C est incrémenté de 1 modulo M.
- Chaque fois que M change, le compteur est ré-initialisé.

2 Introduction à la vérification

2.1 Bascule SR

Une bascule SR a deux entrées booléennes S (Set) et R (Reset) et une sortie booléenne Q qui reflète son état.

Le programme Lustre src/SR.lus donne deux programmations possibles pour la SR. Noter qu'on rajoute une entrée booléenne init qui permet de choisir l'état initial de la bascule.

```
1   -- SR.lus
2   -- Charles André
3   -- September 15, 2004
4   -- purpose: test of assertions
5
6   node SR1 (init,S,R:bool) returns (Q:bool);
7   let
8           Q = init ->
9                   if S then true else
10                          if R then false else
11                                  pre(Q);
12  tel
13
14  node SR2 (init,S,R:bool) returns (Q:bool);
15  let
16          Q = init ->
17                  if R then false else
18                          if S then true else
19                                  pre(Q);
20  tel
21
22  node verif (init,S,R:bool) returns (ok:bool);
23  var
24          q1, q2:bool;
25  let
26          q1 = SR1(init,S,R);
27          q2 = SR2(init,S,R);
28          ok = (q1 = q2);
29  tel
30
31  -- trial:
32  -- lesar SR.lus verif -v -diag
```

Comparer sur divers scénarii le fonctionnement des nœuds SR1 et SR2.

2.2 Test d'équivalence

Le nœud Lustre verif teste l'équivalence de SR1 et SR2. Pour cela , il faut appeler le script lesar qui compile le programme et appelle le model-checker :

```
lesar SR.lus verif -v -diag
```

La conclusion est négative : les nœuds ne sont pas équivalents. Analyser le contre-exemple donné par le logiciel.

2.3 Usage d'assertion

Ajouter l'hypothèse que S et R ne sont jamais simultanément à true. Vérifier que SR1 et SR2 sont alors équivalents. Expliquer pourquoi. (suggestion: comparer les tableau de Karnaugh correspondants).

2.4 Conditions initiales

A l'intérieur du nœud verif, on considère maintenant que les deux SR sont initialisées indépendamment par des flots booléens i1 et i2.

Écrire une assertion qui impose le même état initial aux deux SR.

2.5 Pré-condition de test ♠

Parfois une propriété n'est vraie qu'après un transitoire qui amène le système dans un état particulier.

Écrire un nœud de test qui vérifie la propriété suivante : "SR1 et SR2 sont équivalentes après le premier passage à true de S ou R, à partir du deuxième instant" (ceci toujours sous l'hypothèse que S et R sont exclusifs).

3 Génération d'horloges

On veut engendrer des horloges particulières à partir de l'horloge de base. En particulier on veut une horloge qui donne true toutes les 8 périodes de l'horloge de base. Programmer et visualiser une telle horloge (Le fichier src/clocks.jpg suggère un certain nombre de modules utiles.

Inventer des horloges plus complexes (plusieurs impulsions dans la période).