Programmation Synchrone – Master 2 LC

Examen

14 décembre 2009

Le barème est indicatif. Durée 2 heures. Tous les documents sont permis.

Question 1 : *Styles de programmation* (6 points)

Un compteur de vitesse émet, à chaque arrivée du signal seconde, le nombre de signaux "centimètre parcouru" depuis la dernière occurrence du signal seconde.

1. Modéliser le compteur de vitesse en utilisant un langage synchrone à flot de données.
2. Démontrer que votre modèle est réactif et déterministe.
3. Modéliser le même système en utilisant les automates synchrones (SSM).

Pour chaque modèle, préciser l'interface (entrées et sorties) du modèle et donner au moins une trace d'exécution.

Question 2 : *Signaux échantillonnés* (6 points)

On suppose que l'horloge de base Lustre est une horloge "temps-réel" dont la période, exprimée en secondes, est une constante réelle connue qu'on note T.

Un flot de valeurs réelles est vu comme un signal numérique échantillonné sur la période T. Par exemple, sur la figure 1, le signal u est échantillonné par la séquence de valeurs 1.0, 2.0, 3.0, 2.5, 1.5, 0.5. ...

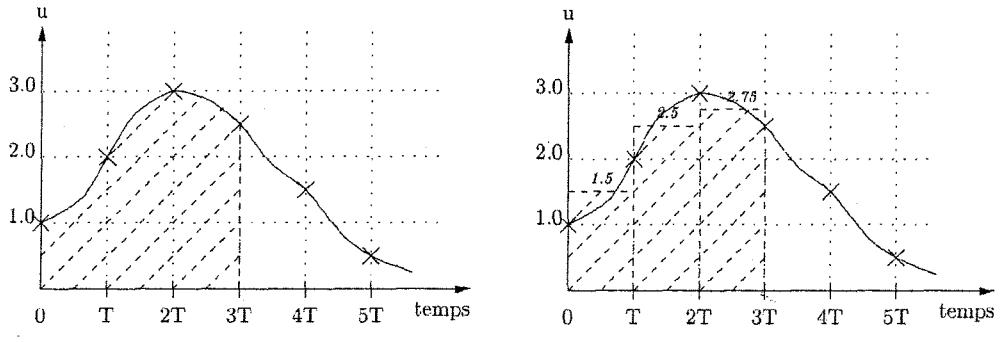

FIG. 1 – Echantillonnage d'un signal et (a) son intégrale exact sur $4T$ ou (b) l'approximation de l'intégrale sur $4T$.

Un outil important en traitement du signal est l'intégrateur : il prend en entrée un signal numérique (vu comme une fonction du temps) et produit en sortie l'intégrale par rapport au temps de ce signal.

Pour un signal continu, on rappelle que la valeur de l'intégrale à un instant t est l'aire comprise entre la courbe, les axes et la verticale de l'instant t ; par exemple, sur la figure 1 (a), la valeur de l'intégrale à l'instant $3T$ est donnée par la surface de la zone hachurée.

La manière la plus simple d'approcher l'intégrale avec des signaux échantillonnés consiste simplement à calculer de proche en proche des sommes de rectangles. Par exemple, sur la figure 1 (b), la somme des rectangles hachurés donne une approximation de l'intégrale à l'instant $3T$.

1. Ecrire un opérateur qui reçoit en entrée un signal u ayant des valeurs réelles et qui calcule à chaque instant une approximation de l'intégrale du signal échantillonné u. Par convention, l'intégrale à l'instant initial est 0.
2. Ecrire le même opérateur si l'échantillonnage du signal u est fait avec une période plus petite T' telle que $T = n \times T'$ et n est une constante donnée. Ainsi, l'opérateur reçoit à chaque instant (multiple de T) un tableau de n valeurs réelles et calcule une approximation (plus fine) de l'intégrale du signal échantillonné u.

Question 3 : *Contrôle d'accès par code secret* (8 points)

On se propose de programmer en SSM un module réactif issu d'un distributeur de billets. Le rôle de ce module, appelé ACCES, est de vérifier que la saisie du code secret se déroule "correctement".

Ce module reçoit en entrée le signal `Carte`, dont une occurrence signifie qu'un utilisateur vient d'insérer une carte dans la machine. Il produit en sortie le signal `Distribuer`, dont l'occurrence signifie que la saisie s'est déroulée correctement, et que l'utilisateur peut accéder à la phase de distribution des billets. Si la saisie ne se déroule pas correctement, ACCES émet le signal `Retenir` : la carte est alors "avalée" par la machine.

La spécification de ce comportement sera précisée et complétée au fur et à mesure des questions. Pour chaque modèle écrit, préciser les noms et les types des signaux (entrées, sorties ou locaux) utilisés.

1. Le module de contrôle ne s'occupe que de la gestion des événements. En particulier, la saisie et la vérification du code sont réalisées par un autre module, DECODEUR, que ACCES déclenche en émettant le signal `DebutCode`. Chaque fois qu'il est déclenché, le module DECODEUR attend que l'utilisateur ait composé et validé un code, puis le vérifie et envoie à ACCES soit le signal `CodeOk`, soit le signal `CodeFaux`. On suppose dans un premier temps que l'utilisateur n'a pas droit à l'erreur dans la saisie du code. Ecrire le modèle SSM du module ACCES qui, chaque fois qu'une carte est insérée dans la machine, déclenche le module DECODEUR puis attend *dans le futur strict* une occurrence du signal `CodeOk`, auquel cas il émet *immédiatement* `Distribuer`, ou une occurrence de `CodeFaux`, auquel cas il émet immédiatement `Retenir`.

2. Modifier le modèle précédent de manière à permettre à l'utilisateur trois essais pour composer le code correct. Plus précisément, si l'utilisateur se trompe à son premier essai, ACCES lui redonne une chance en relançant le module DECODEUR. Il fait de même si l'utilisateur se trompe à son deuxième essai. C'est seulement quand le troisième essai est infructueux que la carte est retenue.

3. Pour que l'utilisateur ne voie pas disparaître sa carte s'il a oublié son code secret, on rajoute une entrée `Annul` au programme ACCES. Avant de taper la troisième fois le code, l'utilisateur peut donc décider à tout moment (même s'il a déjà composé un mauvais code) d'annuler la transaction, auquel cas ACCES émet la (nouvelle) sortie `Restituer`. Compléter le modèle du module ACCES pour qu'il tienne compte de ces nouvelles spécifications.

4. Ecrire l'observateur pour la propriété suivante du module ACCESS : "Le module émet exactement un seul des signaux `Distribuer` et `Retenir` entre deux signaux `Carte`."