# ELECINF102
# Processeurs et Architectures Numériques

Contrôle de connaissances

12 juin 2015 à 8h30

Document autorisé : une feuille recto-verso

Durée: 1h30 minutes

Ce contrôle comporte 3 parties **indépendantes** :

1. Produit scalaire
2. Transmission de données par modulation de position d'impulsion
3. Amélioration du nanoprocesseur

**Consignes importantes** : Si des **schémas** sont demandés dans les différents exercices, ils doivent être impérativement clairs, lisibles et sans ambiguïté. Les dimensions des bus doivent être indiquées. Si nécessaire le sens des signaux doit être précisé. Pour la logique synchrone, les signaux d'horloge et d'initialisation asynchrone (`reset_n`) ne seront pas représentés dans ces schémas.

**BONUS :** Les questions notées **BONUS** sont peut être d'un niveau de difficulté un peu plus élevé que les autres questions. Pensez à faire les autres exercices avant de passer trop de temps sur ces questions.

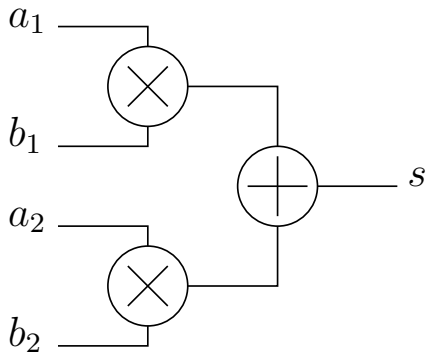
N'oubliez pas d'inscrire nom, prénom, et numéro de casier sur votre copie.

Bon courage !

# 1 Produit Scalaire

On veut effectuer le produit scalaire de 2 vecteurs $\vec{A}$ et $\vec{B}$ ayant $N$ composantes $a_i$ et $b_i$ respectivement. Le produit scalaire $\vec{A} \cdot \vec{B}$ est $s = \sum_{i=1}^{N} a_i \cdot b_i$

Nous ne considérerons que le cas d'un circuit électronique pour $N = 2$. La figure suivante montre l'utilisation de 2 multiplieurs et d'un additionneur, les entrées du calcul étant les composantes des vecteurs.




**Question 1** : Les entrées sont des nombres positifs codés sur 4 bits en virgule fixe : 2 bits pour la partie entière et 2 bits pour la partie fractionnaire. Les calculs se font avec la précision maximale.

Quel est le nombre de bits des parties entières et fractionnaires de la sortie $s$ ?

**Question 2** : Le calcul se fait d'une façon synchrone, les entrées sont issues de registres et la sortie $s$ est échantillonnée dans un registre. Tous les registres ont la même horloge $H$ dont la période est $T_H$. Si le multiplieur à un temps de propagation maximum de $5ns$ et l'additionneur $3ns$ et qu'on néglige les temps de propagation et de pré-positionnement (*set-up*) des registres, quelle est la fréquence maximum du calcul ?
**Nous rappelons qu'un registre est un ensemble de bascules D mises en parallèle et partageant la même horloge.*

**Question 3** : Proposez une méthode pour augmenter le débit de calcul et calculez la fréquence maximale de fonctionnement dans ce cas.

**Question 4 BONUS** : Proposez une méthode pour diminuer le nombre de multiplieurs (faites un shéma). Déterminez la fréquence maximale de fonctionnement de ce nouveau circuit. A quelle fréquence peut on envoyer des nouvelles composantes à cette structure de calcul ?

# 2 Transmission de données par modulation de position d'impulsion

**Les questions 1, 2 et 3 peuvent être traitées indépendamment.**

Nous désirons transférer des données codées sur 4 bit en utilisant une transmission série.

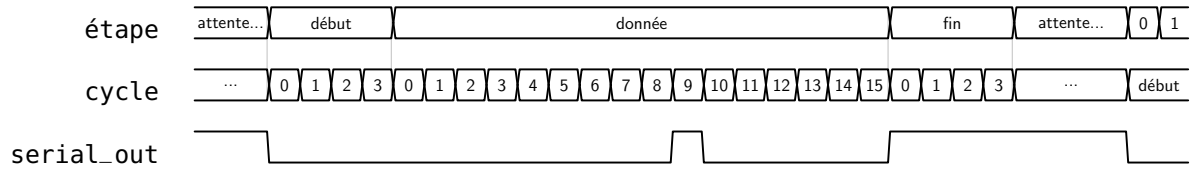
La sortie `serial_out` du dispositif proposé génère de manière synchrone des séquences de bits régulières composées :

- D'un indicateur de début de transmission composé d'une séquence de 4 bits consécutifs à 0.
- D'une séquence de 16 bits dont le seul bit à 1 est celui dont la position dans la séquence correspond à la valeur de la donnée à transmettre.
- D'un indicateur de fin de transmission composé d'une séquence de 4 bits consécutifs à 1.

Enfin :

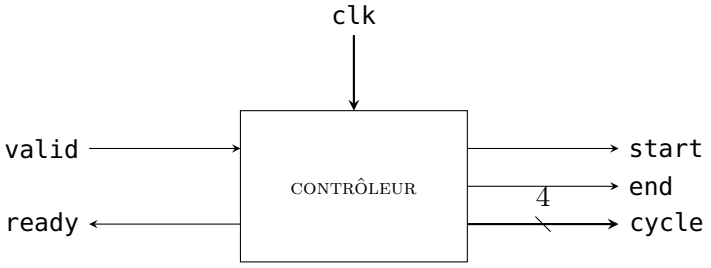
- La sortie du dispositif est bloquée à 1 quand il n'y a pas de transmission.

Le chronogramme suivant montre, par exemple, la transmission de la donnée `9`, suivie du début de la transmission d'une nouvelle donnée.




## 2.1 Un contrôleur de transmission

Ce sous-module gère l'interface de contrôle du dispositif et la durée de la séquence de transmission.




- `reset` : Entrée d'initialisation active à l'état haut.
- `clk` : Entrée d'horloge.
- `valid` : Entrée, permettant à l'utilisateur de valider la présence d'une donnée à transmettre.
- `ready` : Sortie permettant d'indiquer que le dispositif est prêt à recevoir des données.
- `start` : Sortie indiquant que le dispositif est en train de générer la séquence de début.
- `end` : Sortie indiquant que le dispositif est en train de générer la séquence de fin.
- `cycle` : Sortie sur 4 bits indiquant le numéro de la période courante pendant la séquence de transmission de la donnée.

Le fonctionnement du contrôleur est le suivant :

- Au repos, le signal `ready` doit être à 1, les signaux `start` et `end` doivent être à 0.
- Si le signal `valid` passe à 1 (pendant un cycle) alors le contrôleur génère la séquence de transmission :

  **4 périodes** pour l'indicateur du début de transmission (`start=1`)

  **16 périodes** pour la transmission de la donnée

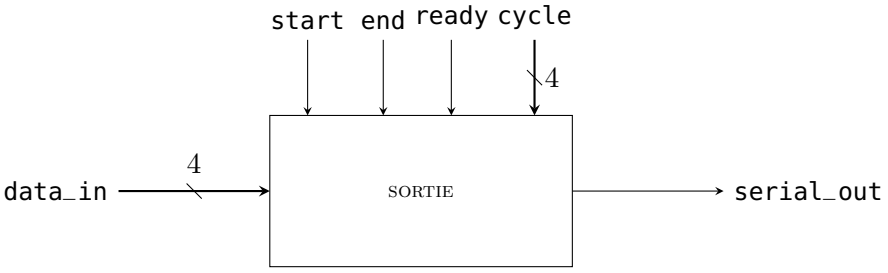
  **4 périodes** pour l'indicateur de fin de transmission (`end=1`)
- Durant toute la séquence de transmission (24 périodes) , le signal `ready` est maintenu à 0 et le signal de commande `valid` est ignoré.

- Le signal `ready` repasse à 1 à la fin de la transmission.
- Enfin, la valeur du signal `cycle` n'a de sens que pendant les 16 périodes effectives de la transmission de la donnée.

**Question 1** : Écrivez le code **SystemVerilog** du module permettant de générer cette séquence.

### 2.2 Génération de la sortie

Un module "sortie" est consacré à la génération de la sortie `serial_out`. La figure suivante montre l'interface de ce sous-module :




La donnée entrante `data_in` est :

- Présente au moment de la validation par le signal `valid`
- Maintenue par l'utilisateur pendant toute la durée de la transmission.

**Question 2** : En remarquant que la sortie `serial_out` peut être générée combinatoirement, écrivez le code **SystemVerilog** du module "sortie".

### 2.3 Adaptation de fréquence

Dans la pratique, la fréquence de transmission n'est pas forcément la même que celle de l'horloge du système. Nous supposons que cette fréquence de transmission est N fois plus lente que la fréquence d'horloge avec N constante entière codée sur 8 bits.

Pour gérer cette adaptation nous créons tout d'abord un générateur de tops, consistant à générer un signal nommé `top`, valant 1 pendant 1 cycle de l'horloge principale tous les N cycles.

**Question 3** : Ecrivez le code **SystemVerilog** d'un tel générateur de `top`.

**Question 4 BONUS** : Proposez une adaptation du module "controle" pour tenir compte de l'utilisation de `top`.

# 3 Amélioration du nanoprocesseur

Tel que proposé en cours et en TP, le nano-processeur manque de fonctionnalités essentielles. Il est, par exemple, difficile de structurer le code en fonctions et sous-programmes.

Nous désirons ajouter deux instructions au nanoprocesseur :

- `JSR` : pour "Jump to SubRoutine". Son argument est l'adresse en mémoire du début du sous-programme appelé.
- `RTS` : pour "ReTurn from Subroutine". Son argument est ignoré
- Ces instructions sont des instructions de saut au même titre que les instructions JMP, JNZ et JNC.

Nous rappelons que le microprocesseur est piloté par un automate en 3 cycles nommés `IF`, `AF` et `EX` pour "Instruction Fetch", "Address Fetch" et "Execute".

Vous trouverez en fin de sujet, un schéma du nano-processeur ainsi que les codes du compteur de programme et du contrôleur de base.

## 3.1 Une approche simpliste

Nous limitons l'appel de sous-programme à un seul niveau (seul le programme principal peut appeler un sous-programme).

L'exécution de `JSR` doit réaliser les actions suivantes :

- Sauvegarder dans le nano-processeur l'adresse de retour, c'est à dire l'adresse de l'instruction suivant l'instruction courante.
- Forcer le nanoprocesseur à se brancher à l'adresse du sous-programme comme dans le cas d'un JMP.

L'exécution de `RTS` doit réaliser les actions suivantes :

- Forcer le nanoprocesseur à se brancher à l'adresse d'instruction précédemment sauvegardée par l'instruction `JSR`.

**Question 1** : Proposez une modification du schéma, et une modification des codes permettant d'implémenter les instructions JSR et RTS.

## 3.2 Une approche plus générale (BONUS)

Pour pouvoir généraliser les appels de sous-programmes (appels imbriqués), la méthode précédente nécessite de multiplier les registres dans le micro-processeur. Une méthode plus générique consiste à sauver l'adresse de retour de sous-programme dans la mémoire externe, et de ne conserver en interne que la position dans la mémoire de cette adresse de retour.

Pour cela nous ajoutons au nanoprocesseur un registre spécifique appelé "pointeur de pile" ou `SP` pour "stack pointer". Nous utiliserons les adresses "hautes" de la mémoire pour stocker les adresses de retour.

La gestion de `SP` en relation avec `JSR` et `RTS` est la suivante :

- A l'initialisation `SP` est forcé à la valeur maximale des adresses en mémoire : 255.
- A chaque exécution de `JSR` :
  - L'adresse de retour de sous programme est stockée en mémoire à l'adresse pointée par `SP`
  - `SP` est décrémenté de 1 (pour anticiper le stockage d'une éventuelle nouvelle adresse de retour...)
- A chaque exécution de RTS,
  - Le nanoprocesseur récupère la donnée pointée par `SP+1` : l'adresse de retour.
  - `SP` est incrémenté de 1 (pour revenir dans l'état avant l'appel du sous-programme)

Nous ne chercherons pas à gérer les cas limites (trop d'appels imbriqués pour la taille de la mémoire, `RTS` sans `JSR` préalable...).

**Question 2** : Proposez une modification du schéma, et une modification du code permettant d'implémenter les instructions JSR et RTS.

**Code du PC "de base"** :

```
...
always @(posedge clk or negedge reset_n)
  if(!reset_n)
    PC <= 0 ;
  else
    if(Load_PC)
       PC <= Q ;
    else
       PC <= PC+Inc_PC ;
...
```

**Code du Contrôleur "de base"** : Seuls les codes utiles pour les questions posées sont indiqués.

```
...
always @(*)
begin
   Inc_PC    <= (Etat == IF ) || (Etat == AF) ;
   Load_PC   <= (Etat == AF ) && (( I == JMP || ( I==JNC && !C) || ( I==JNZ && !Z)) ;
   Load_Add  <= (Etat == AF ) ;
   Sel_Add   <= (Etat == EX ) ;
   Load_I    <= (Etat == IF ) ;
   Load_AZC  <= ...
   WRITE     <= (Etat == EX ) && (I ==  STA) ;
end
...
```

**Schéma du nanoprocesseur** :

Vous pouvez, si vous le voulez, inclure ce schéma "modifié par vos soins" dans votre copie. N'oubliez pas d'indiquer vos noms et prénoms sur le shéma.

- **Nom :**
- **Prénom :**
- **Casier :**