ELECINF102
Processeurs et Architectures Numériques

Contrôle de connaissances

15 juin 2016 à 8h30

Document autorisé : une feuille recto-verso

Durée: 1h30 minutes

Ce contrôle comporte 3 parties **indépendantes** :

1. Générateur de séquence
2. Encodeur rotatif
3. Calcul flottant

Consignes importantes : Si des **schémas** sont réalisés, ils doivent être impérativement clairs 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) ne seront pas représentés dans ces schémas, mais l'état des bascules D à l'initialisation sera indiqué.

Si des **codes SystemVerilog** sont écrits, tous les signaux utilisés doivent être correctement déclarés, leur taille (nombre de bit) doit être définie. Les processus synchrones, ou combinatoires doivent être clairement distingués.

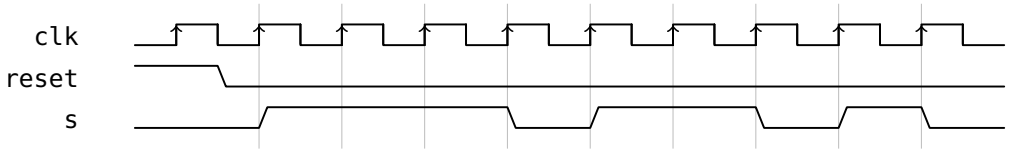
N'oubliez pas d'inscrire nom, prénom, et numéro de casier sur votre copie.

Bon courage !

1 Générateur de séquence

Nous voulons réaliser un générateur de séquence synchrone piloté par une signal d'initialisation `reset` et une horloge `clk` et produisant un signal de sortie `s` dont le comportement est le suivant :

- Le signal `reset` est actif à l'état haut
- Après initialisation le signal `s` suit la séquence présentée dans la figure suivante.
- A la fin de la séquence le signal `s` reste à `0` jusqu'à la prochaine initialisation.



Question 1 : Utilisation d'un **registre à décalage**
Proposez une structure de génération de la séquence basée exclusivement sur l'utilisation d'un **registre à décalage**. Vous pouvez utiliser un shéma ou un code SystemVerilog. Expliquez en quelques lignes votre "intention" avant de réaliser le shéma ou définir le code.

Question 2 : Utilisation d'un **unique compteur**
Proposez une structure de génération de la séquence basée exclusivement sur l'utilisation **d'un unique compteur**, et de logique combinatoire associée. Vous pouvez utiliser un shéma ou un code SystemVerilog. Expliquez en quelques lignes votre "intention" avant de réaliser le shéma ou définir le code.

Question 3 : Utilisation d'un **double compteur**
Le code suivant est proposé, expliquez son principe.

```
module generateur( input  logic reset ,
                   input  logic clk ,
                   output logic s      );

   logic [2:0] num_sequence ;
   logic [1:0] len_sequence ;

   always @(posedge clk or posedge reset)
       if(reset)
          num_sequence <= 4 ;
       else if((num_sequence != 0) && (len_sequence == 0))
          num_sequence <= num_sequence - 1 ;

   always @(posedge clk or posedge reset)
       if(reset)
          len_sequence <= 0 ;
       else
          if(len_sequence == 0) begin
             if(num_sequence != 0) len_sequence <= num_sequence - 1 ;
          end else
             len_sequence <= len_sequence - 1 ;

   always @(*) s <= (len_sequence != 0) ;

endmodule
```

Question 4 : Passage à l'échelle
On suppose maintenant que la séquence est 1023 cycles à 1, 1 cycle à 0, 1022 cycles à 1, 1 cycle à 0, 1021 cycles à 1 . . .Quelle est d'après vous la meilleurs solution ? (expliquez)

- Du point de vue de la compacité du code écrit.
- Du point de vue de la compacité du matériel (bascules D, comparateurs, logique...) nécessaire.

2 Encodeur rotatif

Un encodeur rotatif encode la rotation d'un bouton tournant (bouton de volume par exemple) sur deux signaux binaires A et B. On peut déduire de la séquence temporelle des valeurs de ces deux signaux le sens de la rotation du bouton. La figure 2 indique la succession des valeurs de A et B pour une rotation dans le sens horaire. La figure 3 indique la succession des valeurs de A et B pour une rotation dans le sens antihoraire. Les flèches (+1 et −1) indiquent quand la valeur contrôlée par le bouton doit être incrémentée ou décrémentée.

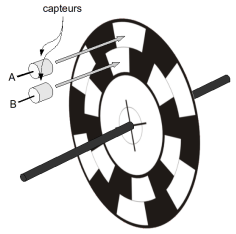


Figure 1 – *Encodeur rotatif*

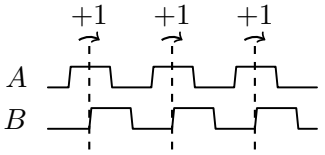


Figure 2 – *Sens horaire*

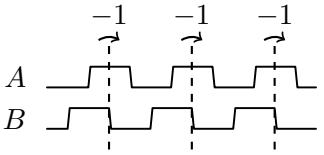


Figure 3 – *Sens antihoraire*

Rappel des conditions de résolution des questions :

- Le signal `clk` est la seule horloge autorisée pour la logique synchrone.
- `clk` est supposée être très rapide devant la vitesse d'évolution des signaux `A` et `B`.
- Le signal `reset` est actif à l'état haut.

Question 1 : **Travaux préliminaires**.

1.A Donnez sous forme d'un schéma *ou* de code SystemVerilog l'implémentation d'un détecteur de fronts montants. Outre le signal `clk`, ce module a comme entrée un signal binaire `D` et comme sortie un signal `UP` qui prend la valeur 1 pendant un cycle de l'horloge si un front montant sur `D` est détecté.

1.B Dans un deuxième schéma *ou* code SystemVerilog), modifiez le détecteur de fronts montants en ajoutant une entrée `EN` telle que `UP` ne signale un front montant que si l'entrée `EN` est active.

Question 2 Décodage des informations de l'encodeur rotatif

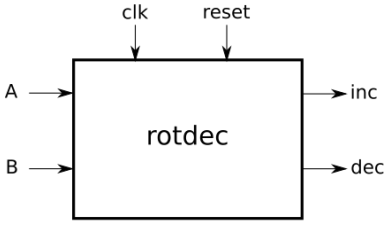


Figure 4 – *Interface du décodeur*

```
module rotdec(input logic  clk,
              input logic  reset,
              input logic  A,
              input logic  B,
              output logic inc,
              output logic dec);
  // ...
endmodule
```

Figure 5 – *Squelette de code*

Concevez un module (schéma *ou* SystemVerilog) qui décode les informations reçues d'un encodeur rotatif (voir figures 4 et 5) :

- Les entrées `A` et `B` proviennent de l'encodeur.
- La sortie `inc` passe à 1 pendant un cycle de `clk` si une rotation dans le sens horaire est détectée.
- La sortie `dec` passe à 1 pendant un cycle de `clk` si une rotation dans le sens antihoraire est détectée.

Question 3 Utilisation de l'encodeur

Concevez un module (schéma *ou* code SystemVerilog) `volume_ctrl` ayant les mêmes entrées que le module précédent (figure 4) et une seule sortie `vol` qui peut prendre une valeur entière comprise entre 0 et 100. `vol` représente, par exemple, la valeur du volume contrôlée par l'encodeur. Réutilisez le module `rotdec` pour construire ce nouveau module.

Il ne doit pas y avoir de dépassement, c'est-à-dire que `vol` ne peut pas être décrémenté en dessous de 0 ni dépasser 100.

3 Calcul flottant dans le nanoprocesseur

La gestion des nombres réels par les microprocesseurs est basée sur une approximation appelée "représentation flottante" dans laquelle un réel est défini par : $R = M * 2^E$ où `M` est une mantisse entière, et `E` est un exposant entier. Nous allons introduire des nombres flottants dans le nanoprocesseur. Comme notre nanoprocesseur ne traite que des mots de 8bits la représentation simpliste choisie est la suivante :

- Tous les nombres sont positifs.
- `E` est **signé** en complément à 2, codé sur **2 bits**, placé dans les 2 bits de **poids fort** du nombre flottant.
- `M` est **positive** codée sur **6 bits**, placée dans les 6 bits de **poids faible** du nombre flottant.
- On nommera `F8` cette représentation des nombres réels.

Notre objectif est de réaliser un opérateur **combinatoire** *fpadd* gérant l'**addition** de 2 nombres de type F8 dont le squelette de son code Verilog est fourni. Un dépassement de capacité, c'est à dire, un résultat d'addition ne pouvant pas être codé sur un nombre de type `F8`, sera signalé par la mise à 1 du signal `error`.

```
module fpadd(FA,FB,FS,error) ;
input  logic [7:0] FA,FB ;
output logic [7:0] FS ;
output logic error;
...
endmodule
```

Question 1 : **Question préliminaire**
Quelle est la valeur du nombre réel non nul le plus petit pouvant être représenté par un code `F8` ? Quelle est la valeur du numbre réel le plus grand pouvant être représenté par un code `F8` ?

Question 2 : **Addition de 2 nombres F8 de même exposant**.
On suppose que les exposants de `FA` et de `FB` sont identiques.

- **2.A** Quelles opérations doit on effectuer pour additionner les deux nombres `FA` et `FB` ?
- **2.B** Que faire pour créer un nombre de type F8 en cas de dépassement de capacité sur la mantisse ?
- **2.C** L'exposant ne peut être compris qu'entre -2 et 1, dans quel cas doit on signaler une erreur ?
- **2.D** En déduire un code SystemVerilog ou un schéma pour générer les signaux `FS` et `error`

Question 3 : **Recadrage de nombres de type F8**.
Pour additionner des nombres d'exposants différents, nous créons un nouveau module *fprec* permettant de préparer des données pour le module *fpadd*. Ce module **combinatoire** prends en entrée 2 nombres `FA` et `FB` quelconques, et génère en sortie deux nombres `FSA` et `FSB` ayant un exposant identique.

```
module fprec(FA,FB,FSA,FSB) ;
input  logic [7:0] FA,FB ;
output logic [7:0] FSA, FSB ;
...
endmodule
```

- Nous voulons réaliser les calculs en minimisant les possibilités de générer des erreurs de dépassement.
- **3.A** Quelle transformations peut on effectuer sur FA et FB pour obtenir deux nombres ayant le même exposant ?
- **3.B** Quelle opération sur les exposants permet de déterminer la transformation à réaliser ?
- **3.C** En déduire un code SystemVerilog ou un schéma pour générer les signaux `FSA` et `FSB`