TELECOM ParisTech

INSTITUT Mines-Télécom

ELECINF 102 : Processeurs et Architectures Numériques

TD Logique séquentielle

Yves Mathieu
yves.mathieu@telecom-paristech.fr

Plan

Rappel SystemVerilog

La bascule D avec enable

Parallélisation

Sérialisation

Détecteur de front montant

Filtrage à moyenne glissante

La bascule D

Toujours, à chaque front d'horloge, copier l'entrée sur la sortie.
Sinon, la sortie ne change pas.

La bascule D

Toujours, à chaque front d'horloge, copier l'entrée sur la sortie.
Sinon, la sortie ne change pas.(implicite)

En SystemVerilog

```
always @(posedge clk)
   q <= d;
```

La bascule D

La remise à zéro asynchrone

. . . sur niveau haut

```
always @(posedge clk or posedge reset)
   if(reset)
      q <= 1'b0;
   else
      q <= d;
```

La bascule D

La remise à zéro asynchrone

. . . sur niveau bas

```
always @(posedge clk or negedge nreset)
   if(!nreset)
      q <= 1'b0;
   else
      q <= d;
```

La bascule D

La remise à zéro synchrone

. . . sur niveau haut

```
always @(posedge clk)
   if(reset)
      q <= 1'b0;
   else
      q <= d;
```

La bascule D

La remise à zéro synchrone

. . . sur niveau bas

```
always @(posedge clk)
   if(!nreset)
      q <= 1'b0;
   else
      q <= d;
```

La bascule D

l'enable

sans reset

```
always @(posedge clk)
   if (en)
      q <= d;
```

La bascule D

l'enable

avec reset asynchrone

```
always @(posedge clk or negedge nreset)
   if(!nreset)
      q <= 1'b0;
   else if (en)
      q <= d;
```

La bascule D

l'enable

avec reset synchrone

```
always @(posedge clk)
   if(!nreset)
      q <= 1'b0;
   else if (en)
      q <= d;
```

Un registre

Un registre de 4 bits

```
logic[3:0] d;
...
logic[3:0] q;

always @(posedge clk)
   q <= d;
```

Un compteur

Un compteur modulo 16

```
logic[3:0] q;

always @(posedge clk)
   q <= q + 1;
```

Dont on ne connait pas l'état initial !

Un registre à décalage

Affectations simultanées

```
logic a, b, c, d;

always @(posedge clk)
begin
   b <= a;
   c <= b;
   d <= c;
end
```

```
logic a, b, c, d;

always @(posedge clk)
begin
   d <= c;
   c <= b;
   b <= a;
end
```

Dans un bloc, entre begin et end, les affectations sont faites simultanément.
L'ordre n'a donc pas d'importance et les deux codes sont équivalents.

- À **droite** d'une affectation <=, l'état **avant** le front.
- À **gauche** d'une affectation <=, l'état **après** le front.

Plan

Rappel SystemVerilog

La bascule D avec enable

Parallélisation

Sérialisation

Détecteur de front montant

Filtrage à moyenne glissante

La bascule D avec enable

- Nous voulons construire une bascule D munie d'un signal **reset** synchrone actif à l'état "haut"
- Cette bascule sera munie d'un signal supplémentaire **enable**
- Si le signal **enable** vaut **1**, la bascule fonctionne normalement
- Si le signal **enable** vaut **0**, la bascule est gelée, **Q** garde sa valeur même après un front montant de l'horloge.

1. Faites un shéma a base de portes simples et d'une bascule D.
2. Ecrivez le code SystemVerilog équivalent

Plan

Rappel SystemVerilog

La bascule D avec enable

Parallélisation

Sérialisation

Détecteur de front montant

Filtrage à moyenne glissante

Parallélisation

- Nous recevons une séquence de données **data_in** codées sur 1 bit.
- A chaque cycle d'horloge un signal **en_in** indique si le bit **data_in** est une donnée valide.
- Nous voulons transformer cette séquence en séquences de données **data_out** de 4bits, accompagnée d'un signal **en_out** indiquant si la donnée **data_out** est valide.

1. Réalisez un chronogramme montrant un exemple de transmission
2. Déterminez le ou les signaux suplémentaires nécessaires au fonctionnement du dispositif
3. Déterminez les éléments logiques nécessaires à la réalisation du dispositif
4. Faites un schéma
5. Ecrivez le code SystemVerilog équivalent

Plan

Rappel SystemVerilog

La bascule D avec enable

Parallélisation

Sérialisation

Détecteur de front montant

Filtrage à moyenne glissante

Sérialisation

- Réception d'une séquence de données de 4 bits **data_in**.
- A chaque cycle d'horloge un signal **en_in** indique si le bit **data_in** est une donnée valide.
- Transformer cette séquence en séquences de données **data_out** de 1bit, accompagnée d'un signal **en_out** indiquant si la donnée **data_out** est valide.

1. Choisissez les conditions fonctionnement (oubliées dans l'énoncé).
2. Réalisez un chronogramme montrant un exemple de transmission
3. Déterminez le ou les signaux suplémentaires nécessaires au fonctionnement du dispositif
4. Déterminez les éléments logiques nécessaires à la réalisation du dispositif
5. Faites un schéma
6. Ecrivez le code SystemVerilog équivalent

Plan

Rappel SystemVerilog

La bascule D avec enable

Parallélisation

Sérialisation

Détecteur de front montant

Filtrage à moyenne glissante

Détecteur de front montant

L'horloge, signal global prédéfini, est l'unique signal pouvant être utilisé comme entrée de synchronisation sur une bascule. A chaque étape, construire un schéma, puis un code SystemVerilog.

1. Comment peut on faire pour détecter d'un cycle à l'autre de l'horloge, un signal est passé de l'état 0 à l'état 1 ?
2. Comment garantir que le signal généré dure exactement un cycle ?
3. Comment modifier le montage pour détecter indifféremment un changement d'état de 0 à 1 ou de 1 à 0 ?
4. A quoi peut servir ce genre de dispositif ?

Plan

Rappel SystemVerilog

La bascule D avec enable

Parallélisation

Sérialisation

Détecteur de front montant

Filtrage à moyenne glissante

Filtrage à moyenne glissante

Nous recevons une séquence de données codées sur 8 bits. A chaque étape, construire un schéma, puis un code SystemVerilog.

1. Réalisez un filtre calculant la somme des 4 derniers échantillons reçus.
2. Réalisez un filtre calculant la moyenne des 4 derniers échantillons reçus.
3. Optimisez la structure pour limiter le nombre d'additionneurs nécessaires à moins de 3.