IFT6223
Systèmes reconfigurables

Règles d'or

Jean Pierre DAVID

david@iro.umontreal.ca

Université de Montréal

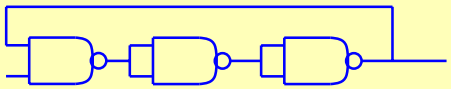
Quelques règles pour éviter les erreurs communes

1. Utiliser une seule horloge (… si possible !)
   a) Pas de boucle combinatoire (direct – induite)
   b) Pas de bascule induite par un « mauvais » VHDL
   c) Un seul process par signal
   d) Pas de glitch sur les signaux asynchrônes
   e) Toujours synchroniser les entrées

2. La conception multi-horloge
   a) Pas de glitch sur l'horloge
   b) Un et seul signal de référence stable avant tous les autres
   c) Synchroniser toutes les entrées sur l'horloge locale sauf si on peut garantir une dépendance de phase entre les horloges
   d) Utiliser des signaux globaux pour le routage des horloges pour éviter le « clock skew »

Pas de boucle combinatoire

1. Un cas trivial



2. Un cas **un peu** moins trivial

Ready_Out<=Ready_IN and Local_Ready;

Pas de bascule induite par un « mauvais » VHDL

```
Process (a,b)
Begin
If (a xor b) then c<=d;
Else c<=a;
End process;
```

Si d varie, le process n'est pas activé
donc c ne varie pas même si (a xor b) est vrai;

Dans les process combinatoires :

- Tous les signaux dont la valeur est utilisée doivent être repris dans la liste de sensibilité.
- Tout appel du process doit conduire à une assignation de tous les signaux qui dépendent du process.

Un seul process par signal

```
Process (clk)
Begin
If (clk'event and clk='1') then
  a<=a+1;
  b<=b+a;
End if;
End process;
```

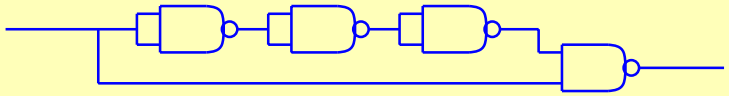
```
Process (clk)
Begin
If (clk'event and clk='1') then
a<=a+4;
b<=b-a;
End if;
End Process;
```

Si deux process affectent des valeurs différentes à un signal, il y un conflit qui sera traité différemment en fonction du compilateur. Exemple : le compilateur affecte le ou logique de deux signaux !!!

Il faut donc toujours s'assurer qu'on affecte une valeur à un signal dans un et un seul process. Exception : les tri-state. C'est au concepteur alors de garantir qu'à aucun moment il n'y aura de conflit cad qu'il y aura un et un seul process qui affectera un état différent de 'Z' au signal concerné.

Pas de glitch sur les signaux asynchrônes

Le « glitch » est un état instable d'un signal juste après que les signaux dont il dépend ont étés modifiés :



A priori, toute fonction logique combinatoire synthétisée dans un FPGA est susceptible de provoquer des glitchs :

- Toutes les entrées ne changent pas d'état exactement au même moment.
- La cascade de cellules logiques peut induire des états intermédiaires instables.

Les signaux de commandes asynchrônes (clrn et prn des bascules) doivent correspondre à la sortie d'un registre pour garantir l'absence de glitch.

Synchroniser les entrées

Tous les signaux qui entrent dans le FPGA peuvent à priori changer d'état à n'importe quel moment

Or, au moment d'un flanc montant de l'horloge, il faut que le système soit dans un état stable sinon le nouvel état pourrait correspondre à un état transitoire virtuel insensé.

Exemple : un compteur avec une entrée « Enable » qui changerait juste avant le flanc montant de l'horloge

Il est donc indispensable que toutes les entrées soient synchronisées sur la même horloge que le circuit (en passant par des registres) de sorte qu'elles ne changent d'état que quand les autres signaux changent d'état et qu'elles soient stables quand les autres signaux sont stables

Pas de glitch sur les horloges

Il peut arriver que l'on soit forcé de travailler avec plusieurs horloges. Certaines horloges seront éventuellement générées en interne. Il faut à tout prix éviter que ces horloges puissent présenter des glitchs.

A priori, une horloge ne peut donc être qu'un signal externe dépourvu de glitch ou un signal interne synchrône, cad qui est connecté à la sortie d'un registre.

Un concepteur averti peut néanmoins garantir que certaines fonctions combinatoires ne peuvent pas générer de glitch mais cela demande une parfaite connaissance de la technologie et de la manière dont elle est utilisée par l'outil de synthèse.

Multi-horloge : un signal stable de référence

Lorsque l'on veut transmettre une information entre deux circuits travaillant avec des horloges différentes, on n'est jamais certain que les données ne vont pas changer juste au « mauvais moment » pour l'autre partie du circuit.

Exemple : une horloge temps réel qui passe de 7H59 à 8H00. Il se pourrait que le système passe par l'état intermédiaire 7H41 … ou 8H50 …

Pour contourner le problème, on peut utiliser un signal de validité de la donnée. Celui-ci est valide un peu après que la donnée a changé et invalide un peu avant que la donnée ne change de nouveau. Lorsqu'on échantillonne l'entrée et que le bit de validité est actif, on est certain que les autres bits sont corrects également.

Multi-horloge : synchroniser … resynchroniser !!!

Chaque partie de circuit qui travaille avec une horloge ne peut voir ses entrées changer à n'importe quel moment.

Comme dans le cas des systèmes à horloge unique, il est nécessaire de synchroniser les entrées du sous-circuit sur son horloge locale.

Cela entraine un nombre considérable de registres supplémentaires puisque chaque donnée échangée demande un ou deux registres selon que le signal est unidirectionnel ou bidirectionnel. Et il faut en plus utiliser un bit de validité.

Erreur commune : « Même si ma donnée n'est pas stable maintenant, le système se ratrappera au coup d'horloge suivant » … rien n'est moins sûr car il pourrait entrer dans un état qui n'est pas défini et y rester ou faire n'importe quoi !

Horloges : utiliser les signaux globaux

Le routage à l'intérieur du FPGA engendre des délais qui ne sont négligeable. Il se pourrait qu'un flanc montant arrive avec un certain retard sur un registre B par rapport à un autre registre A qui dépend de la même horloge. La sortie de A risque donc de varier avant que le flanc montant de l'horloge n'arrive sur B. Si B dépend de A, sont entrée ne sera plus stable et on risque de nouveau d'arriver dans un état instable

Si possible utiliser des signaux globaux pour toutes les horloges car ces signaux garantissent une simultanéité parfaite.

Sinon, regrouper physiquement dans le FPGA les cellules logiques qui dépendent d'une même horloge.