SystemVerilog pour la vérification

Assertions

YTA

Reconfigurable and Embedded Systems Institute
Haute Ecole d'Ingénierie et de Gestion du canton de Vaud

Novembre 2010

Plan

Assertions: Concept

- Les assertions permettent de spécifier le comportement du système
  - De manière interne
  - Au niveau des entrées/sorties
- Servent à
  - La vérification formelle
  - Vérifier le bon fonctionnement du DUT pendant la simulation

Types d'assertions

- Assertions combinatoires
  - Semblables aux assertions VHDL
- Assertions temporelles
  - Permettent de vérifier le comportement au cours du temps
  - Assertions de type: Si séquence A, alors on doit observer séquence B

Séquences

- Il est possible de définir des séquences

Exemple

```
// a suivi de b
sequence seq_a;
    a ##1 b
endsequence

// c suivi de d à 0
sequence seq_b;
    c ##1 !d;
endsequence
```


Propriétés

- Des propriétés peuvent être définies à partir des séquences

Exemple

```
property prop;
    seq_a |=> seq_b;
endproperty
```

- Des assertions peuvent être définies à partir des propriétés

Exemple

```
assert property (
    @(posedge clk)        // agit sur le flanc montant de l'horloge
    disable iff (rst==1) // annule l'assertion lors d'un reset
    prop;
);
```

Assertion

- L'assertion précédente peut être écrite sous forme plus compacte

Exemple

```
assert property (
    @(posedge clk)        // agit sur le flanc montant de l'horloge
    disable iff (rst==1) // annule l'assertion lors d'un reset
    a ##1 b |-> c ##1 !d;
);
```

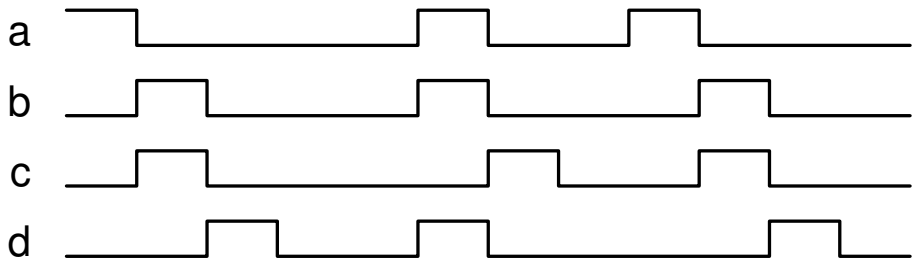
- L'utilisation de séquences est pertinente s'il y a réutilisation possible

Types d'implication

- Implication différée
  - |=>
  - L'évaluation de la séquence de droite commence après la fin de celle de gauche
- Implication directe
  - |->
  - L'évaluation de la séquence de droite commence au moment de la dernière phase de celle de droite

Opérateurs d'implication

Exemple

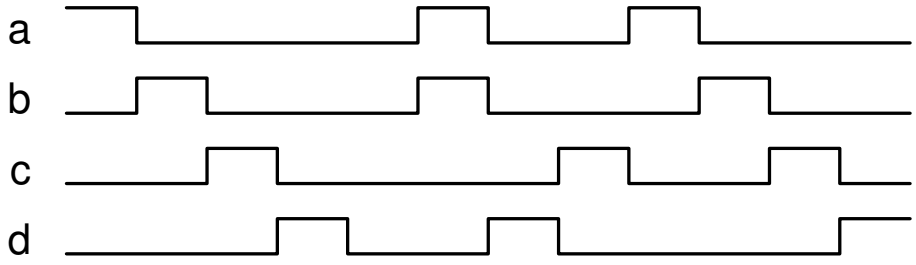


Assertions

```
property a ##1 b |-> c ##1 d;

property a ##1 b |=> c ##1 d;

property a & b |->d ##1 c;
```


Opérateurs d'implication

Exemple



Propriétés

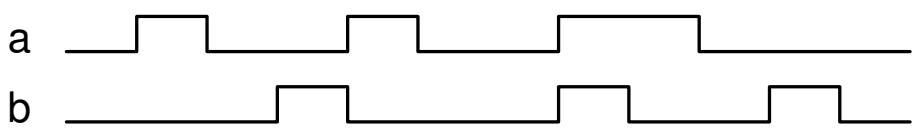
```
property a ##1 b |-> c ##1 d;

property a ##1 b |=> c ##1 d;
```

Délais

- Il est possible de spécifier un délai correspondant à un intervalle de temps
  - `a |-> ##[n:m] b`
    - b est vrai après un temps compris entre n et m
  - `a ##[n:$] b |=> c`
    - Lorsque *a* est observé et que *b* l'est après un temps d'au moins *n*, alors *c* doit l'être un temps après *b*

Délais

Exemple



Propriétés

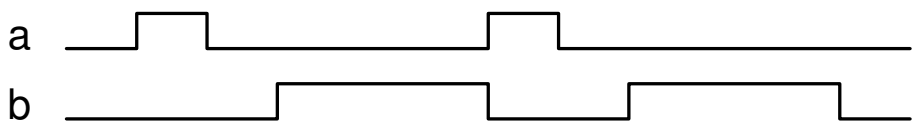
```
property a |-> ##2 b;

property a |-> ##[2:3] b;
```

Répétitions

- Il est possible de spécifier la répétition d'un événement
  - `a[*n] |-> b`
    - Si a est vrai pendant n cycles, alors b doit l'être lors du dernier de a
  - `a[*n:m] ##1 b |=> c`
    - Lorsque *a* est observé pendant *n* à *m* cycles et qu'ensuite *b* est vrai, alors au cycle suivant *c* doit être vrai

Répétitions

Exemple



Propriétés

```
property a |-> ##2 b;

property a |-> ##2 b[*3];
```

Le passé

- L'opérateur `$past` permet d'observer la valeur passée d'une variable
  - `a |-> $past(b)`
    - Si *a* est vrai, alors *b* a dû l'être au cycle précédent
  - `a |-> $past(b,n)`
    - Si *a* est vrai, alors *b* a dû l'être *n* cycles avant