SystemVerilog pour la vérification

Introduction des structures de base

Yann Thoma

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

Octobre 2010

Introduction

- Ce qu'il faut savoir de SystemVerilog pour la vérification
  - Langage orienté objet
  - Notion de classe (monde logiciel)
    - Héritage
    - Création dynamique d'objets
  - Notion de module (monde matériel)
    - Hiérarchie de composants
    - Création statique
  - Notion d'interface (entre-deux mondes)
    - Facilite la mise au point de bancs de tests

Types

logique 4 états

Etats: 0, 1, X, Z

```
logic                // 1 bit
integer              // 32 bits signé
integer unsigned     // 32 bits non signé


logic [7:0]           a;  // 8 bits  non signé
logic signed [31:0] b;  // 32 bits signé
integer               s;  // 32 bits signé
integer unsigned      u;  // 32 bits non signé
```

Types

logique 2 états

Etats: 0, 1

```
bit       // 1 bit
byte      // 8 bits signé
shortint  // 16 bits signé
int       // 32 bits signé
longint   // 64 bits signé
```

- Avantages:
  - Peut être simulé plus rapidement
  - Utilise moins de mémoire
- Conversions possibles entre 2 et 4 états

Types

- Attention, le langage n'est pas très typé
- Les troncatures et extensions se font sans avertissement

```
logic [15:0] log;    // valeur initiale: 8'bx
int i;               // valeur initiale: 0

i   = log;           // i=0
i   = -1;            // i=-1
log = i;             // log=16'hffff
i   = log;           // i=65535
```

Types non entiers

```
time        // entier (au moins) 64 bits
shortreal   // float C
real        // double C
realtime    // comme real
```


Types énumérés

```
enum {rouge, vert, bleu} couleur;

couleur= vert;

typedef enum {rouge, vert, bleu} couleur_t;

couleur_t c;
c=vert;
```

Structures

Identique à C

```
struct {
    byte red;
    byte green;
    byte blue;
} pixel;

pixel.blue= 8'h02;

typedef struct {
    byte red;
    byte green;
    byte blue;
} pixel_t;

pixel_t pixel;
pixel.blue=8'h10;
```


Packages

Définition

```
package LeNotre_pkg;
  typedef struct {
    byte red;
    byte green;
    byte blue;
  } pixel_t;

  function logic IsDark(
           input pixel_t);
    ...

  pixel_t BasicColor;

endpackage
```

Utilisation

```
module NotreModule;
  import LeNotre_pkg::*;

  pixel_t unpixel;

  initial
    NotreModule::BasicColor=
        {8'h01,8'h10,8'h00};
    ...

endmodule
```

Types: tableaux

```
// Vecteur, tableau de bits
logic[7:0] unOctet;
unOctet[3] = 1'b0;

// Tableau de vecteurs
logic[7:0] tableaudOctets [0:31];
tableaudOctets[3][5]=1'b0;
tableaudOctets[4]=8'h7f;

typedef logic[7:0] octet_t;
```


Randomisation simple

- Fonction permettant de donner une valeur aléatoire à une variable:
- *randomize()*

Exemple

```
logic[7:0] var, var2;

assert(randomize(var));

assert(randomize(var) with {var > 8'h05;});

assert(randomize(var,var2) with {var2>var;var > 8'h05;});
```

Gestion du temps

```
    logic clk=0;
    logic [7:0] input0;
    logic output0;

    // génération de l'horloge. Période: 10 ns
    always #5 clk = ~clk;

    // Clocking block
    default clocking cb @(posedge clk);
        output clk, rst, input0;
        input output0;
    endclocking

task stimuli();
    input0<=0;
    ##1;        // Attente d'un cycle d'horloge
    input0<=1;
    ##10;       // Attente de 10 cycles d'horloge
    input0<=0;
    ...
endtask
```


Hiérarchie

- Contrairement à VHDL
- Possibilité d'accéder à n'importe quel signal de la hiérarchie
- *$root* désigne le top de la hiérarchie

Exemple

```
$root.BancdeTest.DUT.entreeA
```

Compilation

```
vlog -sv fichier1.sv fichier2.sv
```