ARCHITECTURE LOGIQUE PROGRAMMABLE
Exposé 03

Cne. Brinsi Riadh

Académie militaire F.J

br.riadh@gmail.com

24 novembre 2015

Signal (1/2)

- Un signal représente une équipotentielle
- Il doit être déclaré avant utilisation
- Il peut être déclaré :
  - dans un package, il est alors global
  - dans une entity, il est alors commun à toutes les architectures de l'entité
  - dans l'architecture, il est alors local

```
package Pkg_essai is
signal clock: std_logic;
signal reset: std_logic;
end Pkg_essai;
```

```
entity essai is port(
    a: in std_logic;
    b: in std_logic;
    s: out std_logic);
    signal reset: std_logic;
end essai;
```

```
architecture arch of essai is
  signal reset: std_logic;
begin
end behavioral;
```

Signal (2/2)

- L'affectation se fait avec l'opérateur « <= »
- Accès à des sous-éléments avec l'opérateur alias

```
alias lsb :std_logic_vector(7 downto 0)is add_bus(7 downto 0) ;
```

- Initialisation rapide

```
Toto <=(others=> '0'); est équivalent à Toto <= "00000...0";
```

Variable

- Une variable doit être déclarée avant utilisation
- Elle ne peut être déclarée que dans un « process »
- L'affectation se fait avec l'opérateur « := »

```
architecture arch of essai is
 begin
   P1:process
      variable a: integer;
   begin
      a:=5;
   end process ;
 end arch;
```

Constant

- Une constante doit être déclarée avant utilisation
- Elle peut être déclarée :
  - dans un package, elle est alors globale
  - dans une entity, elle est alors commune à toutes les architectures de l'entité
  - dans l'architecture, elle est alors locale
- L'affectation se fait avec ' « := »

```
architecture arch of essai is
 constant VCC:std_logic:='1';
 constant Gnd:std_logic:='0';
 begin
end arch;
```

Assignations concurrentes

- Toutes les déclarations sont exécutées simultanément et en permanence
- L'ordre des déclarations dans le code source n'a pas d'influence
- Les déclarations possibles sont :
  - Assignation continue :<=
  - Instantiation d'un composant :port map
  - Assignation conditionnelle :when ... else
  - Assignation sélective :with ... select ... when ... when
  - Appel d'un process
  - Instruction generate
  - Appel d'une fonction

Assignations séquentielles

- Ce mode concerne uniquement les function, procedure et process
- Les process manipulent les variable et signal
- Au sein de ces descriptions, les déclarations sont exécutées de manière séquentielle, l'une après l'autre
- L'ordre des déclarations est donc important
- Les déclarations possibles sont :
  - Assignation continue :<=(signal) et :=(variable)
  - Assignation conditionnelle :if ... then ... elsif ... then ... else ... end if ;
  - Assignation sélective :case...is...when...=>...when...=>...end case ;
  - Boucles :for ... in ... loop ... end loop ;
  - Boucles :while ... loop ... end loop ;
  - Instructions next et exit

process synchronisé

Mise en œuvre de process synchronisé sur l'horloge

- If clk'event and clk='1' then ou if rising_edge(clk)
- If clk'event and clk='0' then ou if fallin_edge(clk)

Assignations conditionnelles

Assignation concurrente

- Forme générale :
  signal<=valeur when condition else autre valeur when autre condition ... ;
- Une seule cible peut être assignée

```
s<=a when (sel="00") else
   b when (sel="01") else
   c;
```

- Mémorisation implicite lorsque toutes les conditions ne sont pas listées

Assignations sélectives

Assignation concurrente

- Forme générale : With sélecteur select signal <= valeur when val_sel, vvaleur when val_sel2 ... ;
- Possibilité de regrouper plusieurs valeurs du sélecteur pour une même assignation "|"

```
with selecteur select
      s <= a when "00",
           b when "01"|"11",
           c when "10",
           d when others;
```

- Mémorisation implicite lorsque toutes les conditions ne sont pas listées

Exercices

Exercice 03-1 : Mux4to1 utilisant if ...
Exercice 03-2 : Mux4to1 utilisant case ...
Comparer le rapport synthèse de chaque description VHDL

Exercices

Exercice 03-3 : Additionneur 8 bits

Exercice 03-4 : Multiplieur 8 bits