ARCHITECTURE LOGIQUE PROGRAMMABLE
Exposé 04

Cne. Brinsi Riadh

Académie militaire F.J

br.riadh@gmail.com

December 1, 2015

Modèles Génériques

Génération de circuits paramétrables

Le langage permet de paramétrer les modèles pour les rendre plus génériques grâce à :

- Les paramètres génériques
- Les objets d'interface non contraints
- L'instruction concurrente generate

syntaxe:

```
entity entity_name is
Generic (" parameters list ");
Port ( " ports list " );
end entity;
```

Exemple:

```
entity Nbit_adder is
generic( SIZE : positive :=8;);
port (
a,b : in unsigned( SIZE-1 downto 0);
ci : in std_logic;
sum : out unsigned( SIZE-1 downto 0);
co : out std_logic);
end entity;
```

instanciation:

```
inst: Nbit_adder generic map (16) port map(A, B, C, sum, C);
```

Modèles Génériques

Instruction generate

Elle permet de dupliquer de manière concurrente des instructions de manière itérative ou conditionnelle Les étiquettes des generate sont obligatoires La structure de contrôle itérative :

```
Etiquette : for ... in ... to ... generate
                    ...
            end generate
```

Les itérateurs dans la boucle ne sont pas à déclarer la structure conditionnelle :

```
Etiquette : if ... generate
            end generate ;
```

Exercice Adder n bits

Modèles Génériques

Instruction séquentielle: Loop

Forme générale :

```
for i in val_deb to val_fin loop... end loop;

while condition loop....end loop;
```

Modèles Génériques

Instruction séquentielle: Loop

Forme générale :

```
for i in val_deb to val_fin loop... end loop;
```

```
library ieee;
use ieee.std_logic_1164. all;
entity oddParityLoop is
   generic (width : integer : = 8);
   port (ad : in std_logic_vector(width-1 downto 0);
   oddParity : out std_logic);
   end oddParityLoop ;
architecture synt of oddParityLoop;
begin
   process (ad)
   variable loopXor : std_logic;
   begin
      loopXor := ' 0 ;' ;
      for i in 0 to width-1 loop
         loopXor := loopXor xor ad(i);
      end loop;
      oddParity  <= loopXor;
   end process;
end synt;
```

Modèles Génériques

Instruction séquentielle: Loop

- Un package sert à réutiliser un ensemble de déclarations de composants
  - Un package commence par le mot clé package et se termine par le mot clé end
  - Il contient les définitions de composants et de signaux (et plus...)
- Ce package pourrait être sauvegardé dans la bibliothèque WORK et appelé par

```
use WORK.LOGIC_OPS.all
```

```
package LOGIC_OPS is
   component AND2_OP
      port (A, B : in BIT; z : out BIT);
   end component;
   component OR32_OP
      port (A, B, C : in BIT; z : out BIT);
   end component;
   component NOT_OP
      port (A, : in BIT; A_BAR : out BIT);
   end component;
end LOGIC_OPS;
```

Exercice Carry Sselect Adder