Unité : CSN & SysLog2

Processus et instructions séquentielles

HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD

Etienne Messerli

Octobre 2021

Déroulement concurrent et séquentiel

- Dans un langage informatique classique :
  - les instructions ont un déroulement séquentiel
- Dans un circuit :
  - toutes les portes fonctionnent simultanément
  - tous les signaux évoluent de manière concurrente
- Le langage VHDL dispose d'instructions concurrentes pour décrire le comportement des circuits réels (matériel)

Processus concurrents ...

- Toutes les portes d'un circuit sous tension fonctionnent continuellement →

  une description VHDL est constitué d'un ensemble de processus concurrents

- L'ordre dans lequel les processus apparaissent dans la description est indifférent

```
D <= not C;
C <= A and B;
```

est identique à

```
C <= A and B;
D <= not C;
```

…processus concurrents …

- Des processus simples peuvent être déclarés de façon implicite, en les décrivant avec une affectation simple, conditionnelle ou sélectionnée

```
-- une affectation simple,
-- hors d'une déclaration de process
A <= B or C ;
```

revient au même que :

```
process (B,C) begin
   A <= B or C ;
end process;
```

…processus concurrents …

```
A <= '1' when B='1' else
     '1' when C='1' else '0';
-- ATTENTION : l'affectation conditionnelle avec when else
-- est une instruction concurrente
-- (utilisation interdite dans un process)
```

revient au même que :

```
process (B,C)
begin                              -- ATTENTION :
  if B='1' then                    -- la structure conditionnelle
    A <= '1';                      -- if then else est une
  elsif C='1' then                 -- instruction séquentielle
    A <= '1';                      -- doit être utilisée dans
  else                             -- uniquement dans un process
    A <= '0';
  end if;
end process;
```

…processus concurrents

Comme une porte logique, une instruction concurrente (processus auto déclaré) :

- **s'active** lorsqu'un des signaux d'entrées (droite de l'expression) **change** de valeur,
- et **se rendort** lorsque la sortie (bit ou vecteur) a atteint un **état stable**

L'instruction *process* ...

- Le langage VHDL dispose d'une instruction *process* dont la syntaxe est :

```
process (Liste_De_Sensibilité)
  --zone de déclaration
begin

  --Zone pour instructions
  --                séquentielles ....

end process;
```

… instruction *process* …

un *process*

- **s'active** lorsqu'un des signaux de sa liste de sensibilité (entrées) **change**,
- et **se rendort** lorsque toutes les instructions séquentielles ont été évaluées **une seule fois** (*end process*)

…instruction *process* …

- L'exécution des instructions à l'intérieur d'un *process* est séquentielle
- Le temps simulé est stoppé durant l'exécution d'un *process*, donc
- les affectations des signaux ne prennent effet qu'après **l'endormissement** du process (fin de l'exécution de l'algorithme séquentiel)

…instruction *process* …

- Une description avec un *process* sert à décrire les relations entre les entrées et les sorties sous une forme informatique:
  algorithme à exécution séquentielle
- Les descriptions de forme informatique requièrent souvent l'utilisation de variables (au sens informatique)
  => fait pas partie matière cours CSN/SysLog2
  (cela sera traité dans le cours à choix CSF)

Syntaxe de l'instruction processus

```
[Label:] process (Liste_De_Sensibilité)
                 -- ↑ Signaux_activant_le_processus
  --zone de déclaration
begin

  --Zone pour instructions
  --              séquentielles ....

end process;
```

Elément puissant du langage VHDL

Processus, zone de déclaration

- Déclaration de variables et de constantes
- Pour utilisateurs expérimentés :
  - déclarations de types et de sous-types
  - déclaration de variables
  - déclaration de fonctions et de procédures
- Déclaration de signaux ou de composants interdite dans un processus

Processus et liste de sensibilité …

Avec liste de sensibilité :

- Utilisé pour les descriptions synthétisables
- Utilisé dans le cadre des unités CSN et SysLog2

Sans liste de sensibilité :

- Utilisé pour fichiers de simulation & spécification
- Sera traité dans l'unité à choix CSF

Processus et liste de sensibilité …

- Avec liste de sensibilité :
  - utilisé pour les descriptions **synthétisables**
  - processus activé **uniquement** lorsqu'un signal de la liste de sensibilité change
  - instruction *wait* interdite
  - processus endormi à la fin (*end process*)
  - permet de décrire avec un algorithme :
    - système combinatoire ou
    - système séquentiel ou
    - *modèle, spécification.*

Les instructions séquentielles

Les instructions séquentielles **doivent être** placées **à l'intérieur** d'un processus.

Liste des instructions séquentielles:

- Affectation d'un signal `y <= a and c;`
- Affectation de variable `var1 := d and x;`
- Instruction de test **`if ... then else`**
- Instruction de choix **`case ... is when`**

Syntaxe de l'instr. d'affectation d'un signal

Affectation d'un signal :

```
signal <= a fonct_logique b;
```

Le signal sera affecté à **l'endormissement** du processus (**`end process ou wait ...`**)

Remarque :

L'affectation avec condition (`..<= ..when..`) n'est pas utilisable à l'intérieur d'un *process*. C'est une instruction concurrente

Syntaxe de l'instr. d'affectation d'une variable

- Affectation d'une variable :

```
variable1 := x fonct_logique z;
```

- Exemple :

```
Temp := var0 or signal_a or var2;
```

La variable est actualisée **immédiatement**

L'utilisation des variables est possible seulement à l'intérieur de l'instruction process (zone de description séquentielle)

Exemple de processus I

Description d'un système combinatoire avec un processus

```
process (A, B, C)
begin
   T <= A and W;
   V <= C and T;
   W <= C or  B;
end process;
```

Complétez le chronogramme ci-après

Chronogramme I

```
process (A, B, C)
begin
   T <= A and W;
   V <= C and T;
   W <= C or  B;
end process;
```

Entrées :

A

B

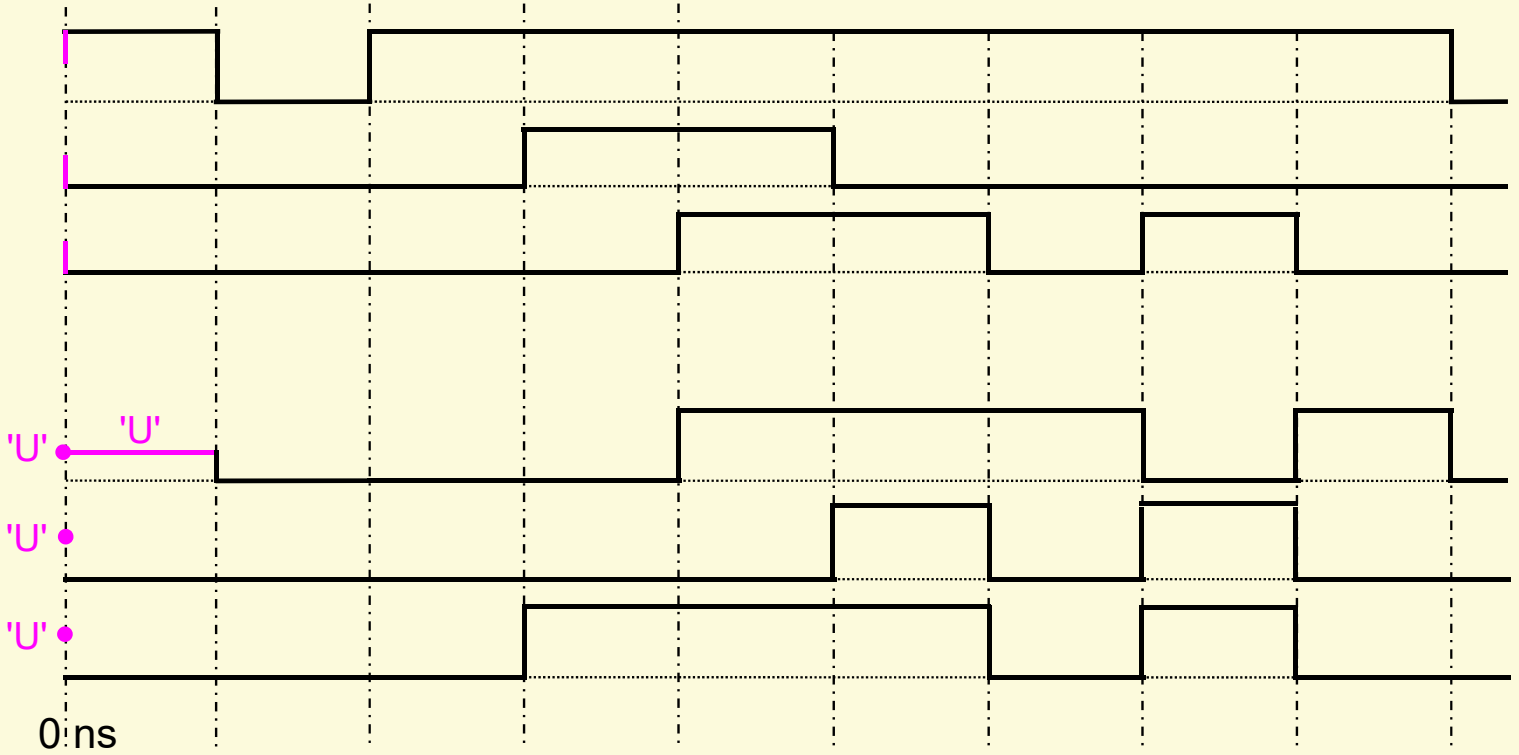
C

Sorties :

T

V

W



Remarque sur processus I

Cette description est non synthétisable

```
process (A, B, C)
begin
   T <= A and W;
   V <= C and T;
   W <= C or  B;
end process;
```

Pour quelle raison (erreur) cette description est non synthétisable ?

Exemple de processus II

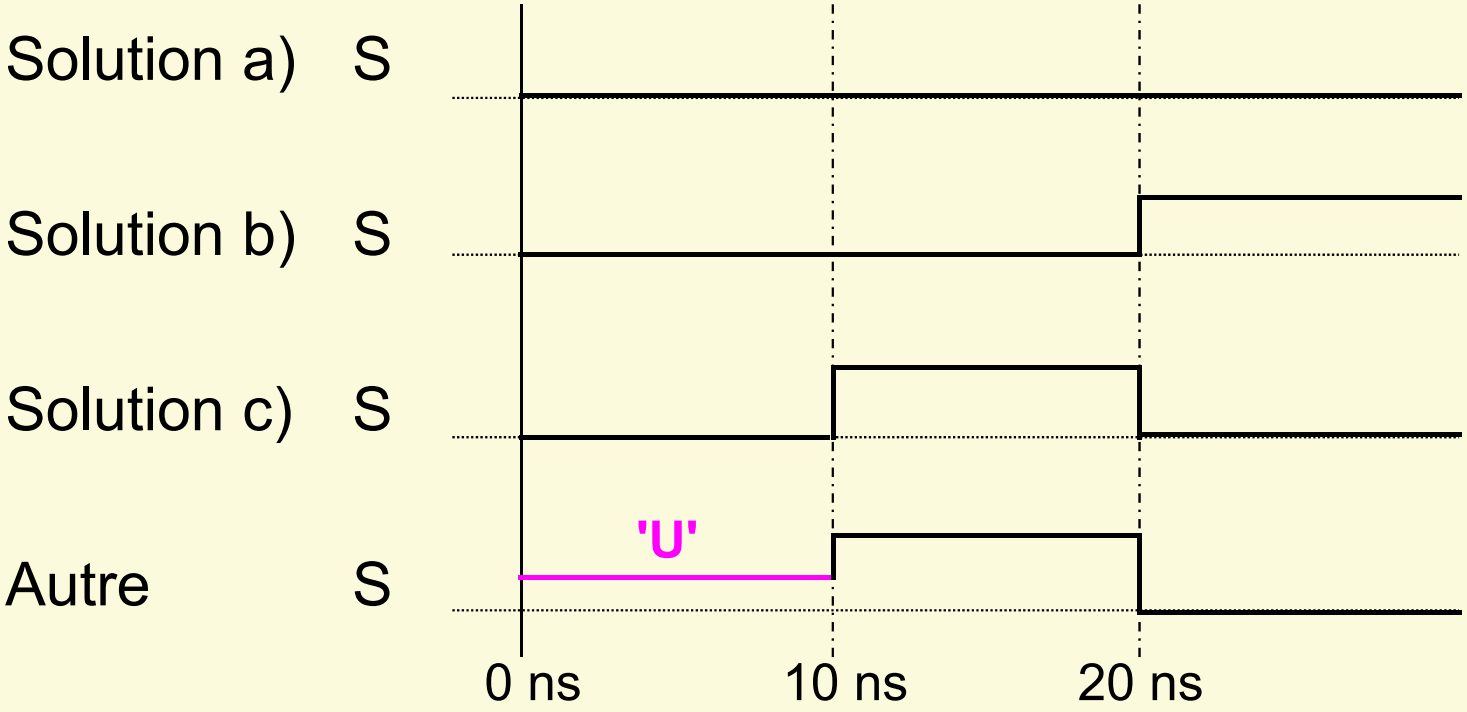
Comportement séquentielle à l'intérieur d'un process

```
process
begin
   S <= '0';
   S <= '1' after 20 ns;
   S <= '1' after 10 ns, '0' after 20 ns;
   wait ;
end process;
```

Complétez le chronogramme ci-après

Quel chronogramme est correct ?

```
process
begin
 S <= '0';
 S <= '1' after 20 ns;
 S <= '1' after 10 ns,
        '0' after 20 ns;
 wait ;
end process;
```



Syntaxe de l'instruction conditionnelle

```
process ( ..... )
begin
  if Condition1 then
     --ZoneInstructions_1
  elsif Condition2 then
     --ZoneInstructions_2

    ...

  else
     --ZoneInstructions_n
  end if;
end process;
```

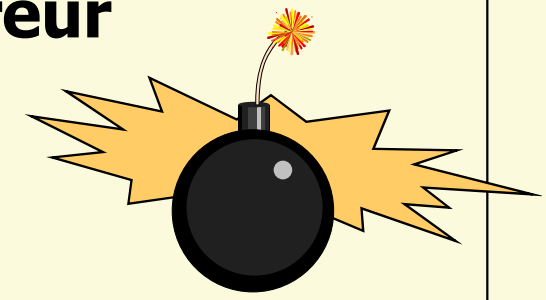
Exemple d'instruction conditionnelle

Démultiplexeur

```
process (Sel, Enable)
begin
    Sorties <= "0000"; --recommandé
  if Enable = '1' then
    if Sel = "00" then
      Sorties <= "0001";
    elsif Sel = "01" then
      Sorties <= "0010";
    elsif Sel = "10" then
      Sorties <= "0100";
    else
      Sorties <= "1000";
    end if;
  end if;
end process;
```

```
else
  Sorties <= "0000";
```

Exemple avec erreur



Syntaxe de l'instruction de choix

```
process ( ..... )
begin
  case Expression is
     when Valeur1    => --ZoneInstructions_1;
     when Valeur2    => --ZoneInstructions_2;
        ...
        ...
     when others     => --ZoneInstructions_n;
  end case;
end process;
```

Exemple d'instruction de choix

Décodeur d'adresse

```
process(adresse)
begin
   CS_A <= '0'; --Valeur par defaut
   CS_B <= '0'; --Valeur par defaut
   case Adresse is
      when "000"|"001"|"010" => CS_A <= '1';
      when "011"|"100"       =>  null;
      when "101"|"110"|"111" => CS_B <= '1';
      when others => CS_A <= 'X';CS_B <= 'X';
                        --cas pour simulation
   end case;
end process;
```

Description SLC avec un process

IMPORTANT (SLC : Système Logique Combinatoire)

- Indiqués d'utiliser toutes les entrées avec la syntaxe :

```
process(all) -- VHDL 2008
```

- Toutes les sorties du SLC doivent être initialisées **au début** du processus.
  ou
  *Toutes les sorties du SLC doivent êtres affectées dans **toutes** les branches des instructions séquentielles.*

- **Interdit** d'utiliser, dans le *process*, les sorties générées par celui-ci ! (Si nécessaire utiliser une variable, voir unité CSF)

Exercice décodeur d'adresse

Donner la description VHDL, avec un process, du décodeur d'adresse ci-dessous.

Adresse	cs_rom	cs_ram	cs_flash	cs_io	
0x0000 - 0x0FFF	1	0	0	0	ROM
0x1000 - 0x4FFF	0	0	0	0	zone libre
0x5000 - 0x7FFF	0	1	0	0	RAM
0x8000 - 0x9FFF	0	0	1	0	FLASH
0xA000 - 0xEFFF	0	0	0	0	zone libre
0xF000 - 0xFFFF	0	0	0	1	voir détails

Une archive avec un chablon et un testbench est disponible sur Cyberlearn

Puis vous devez rendre votre fichier *.vhd sur Cyberlearn

Exercice décodeur d'adresse

Détail pour la zone des IOs.

Adresse	cs_io	CS spécifiques actifs
0xF000 - 0xF00F	1	cs_leds actif
0xF010 - 0xF01F	1	cs_switch actif
0xF020 - 0xF03F	1	cs_matrice_led actif
0xF040 - 0xF09F	1	zone libre
0xF0A0 - 0xF0BF	1	cs_capt_analog actif
0xF0C0 - 0xF0DF	1	cs_cmd_moteur actif
0xF0E0 - 0xF0FF	1	zone libre
0xF100 - 0xFFFF	1	zone libre

Questions ?