Cours VHDL

A noté :

* Domaine concurrent -> espace entre « begin » et « end » de l’architecture
  + Traitements parallèles
* Tableau de vérité : sorties en fonction des différentes combinaisons d’entrée
* Additionner deux STD\_LOGIC\_VECTOR -> convertir les deux vector en signer, puis convertir l’addition en tant que vector
* Constante de type vector entre double cote
* (others => ‘0’) renvoie un vecteur de 0
* « & » correspond a la concaténation

Les signaux :

* Initialisation d’un signal :
  + Signal nomDuSignal : typeDeSignal := valeurInitiale ;
* STD\_LOGIC\_VECTOR : Tableau de STD logic
  + Initialisation : STD\_LOGIC\_VECTOR(3 downto 0) -> initialiser de ma décroissante
  + STD\_LOGIC\_VECTOR(1 to 4) -> initialiser de manière croissante
* Type INTEGER : ce sont des entiers
  + Initialisation : nomVariable : integer range 0 to 20 -> entier limité à l’intervalle 0 ;20

Process :

* Initialisation d’un process : nom\_du\_process : process(liste\_de\_sensibilité)
  + Avec liste de sensibilité = liste des signaux dont le changement d’état lance le calcul du process
  + Si liste vide, le process est exécuté sans condition
* Une image contenant texte

  Description générée automatiquementLes process s’exécutent de façon parallèle

Figure 1 : Structure d'un process

* Initialisation de variable : variable nom\_variable : type\_variable := valeur\_initiale ;

Structure conditionnelle :

* With entrees select Sortie <= ‘1’ when « 11 » ‘0’ when others ; -> Traduction : quand entrees est égale a 11 sortie prendre la valeur 1, quand entrees est égale a n’importe quoi d’autre sortie prendre 0
* Structure if : If … then … elsif … then … else … end if ;
* Structure case :Une image contenant texte

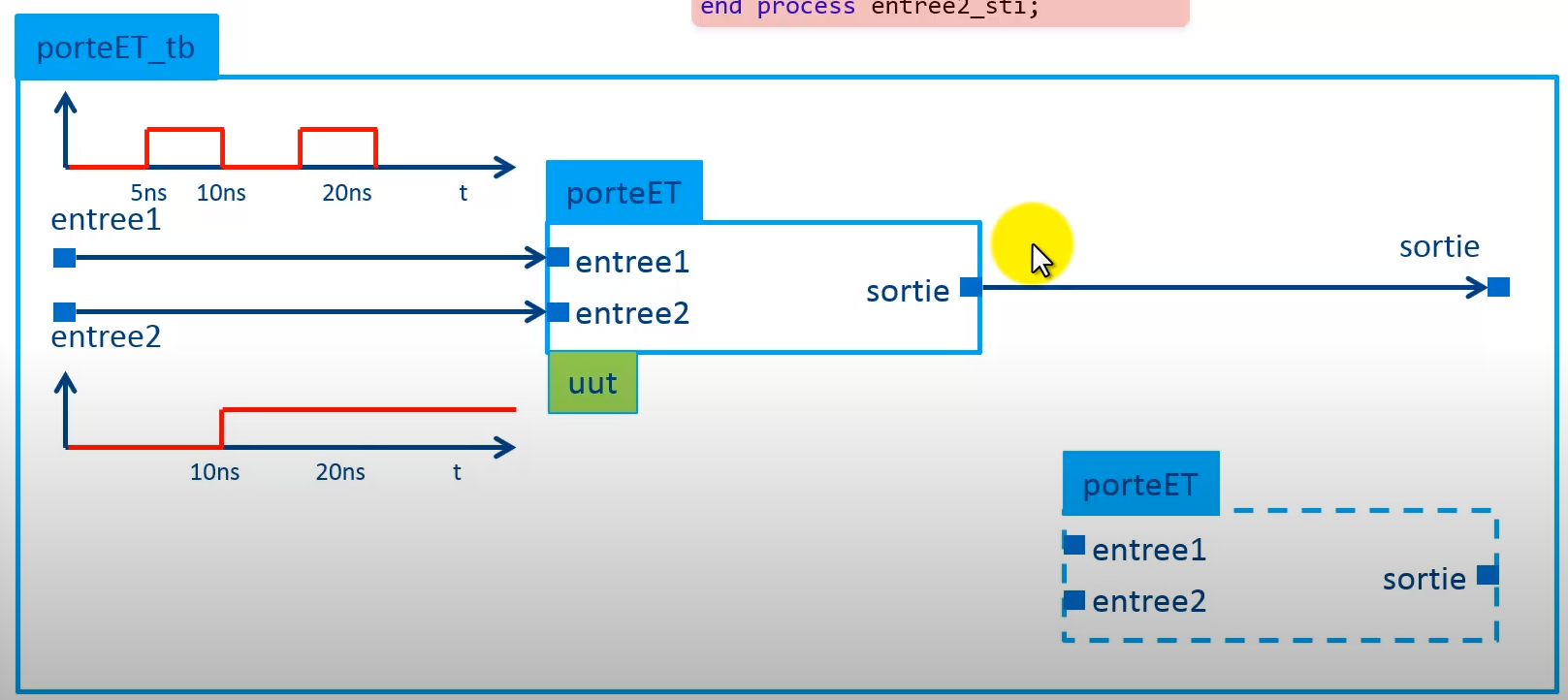
  Description générée automatiquement
* Boucle for: Une image contenant texte

  Description générée automatiquement
* Boucle while : Une image contenant texte

  Description générée automatiquement
* When … else … : 

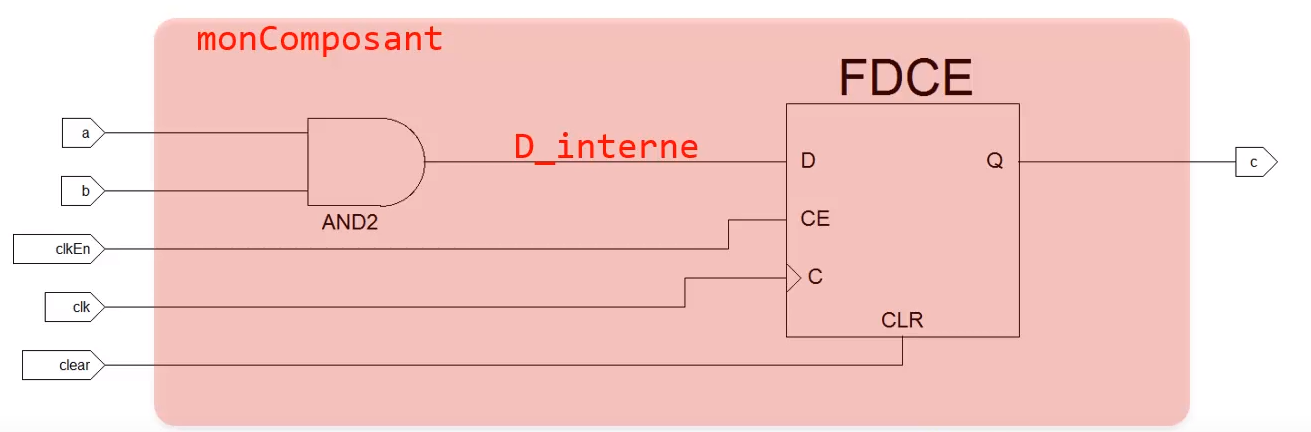
Les testbench :

* Voici une représentation graphique du testbench écrit sur le .txt. Les « entree1 », « entree2 » et « sortie » du carré « porteET » du schéma sont nommé « entreeC1 » « entreeC2 » » et « SortieC » sur le .txt.



* Les différentes étapes du testBench :
  + Initialisation des signaux et composant dans l’architecture et avant le « begin »
  + Mapping des signaux internes à l’architecture de test avec les signaux du composant
  + Création des différents process exécuter en simultané

Description structurelle :

* Schéma du bloc que nous avons d’écrit.

Conversion :

* TO\_SIGNED : permet de convertir un interger en signer
* STD\_LOGIC\_VECTOR : permet de convertir un signer en std logic vector
* TO\_INTEGER : permet de convertir un signer ou un non signer en interger
* SIGNED : permet de convertir un std logic vector en signer
* UNSIGNED : permet de convertir un std logic vector en non signer

Les attributs :

* « ‘event » : retourne true si le signal a changé de valeur en entrant dans le process
* « ‘last\_value » : retourne la valeur du signal avant le dernier évènement.
* Utiliser afin de déterminer un front montant dans une clock
* Exemple : clk’event and clk’last\_value = ‘0’ and clk = ‘1’ -> vérifie si la dernière valeur est égale a ‘0’ et si la valeur actuelle est égale a ‘1’
* Pour détecter les fronts montant d’horloge il y a la macro : rising\_edge(clk) ;