Introduction au VHDL

Very High Speed Integrated Circuit Hardware Description Language

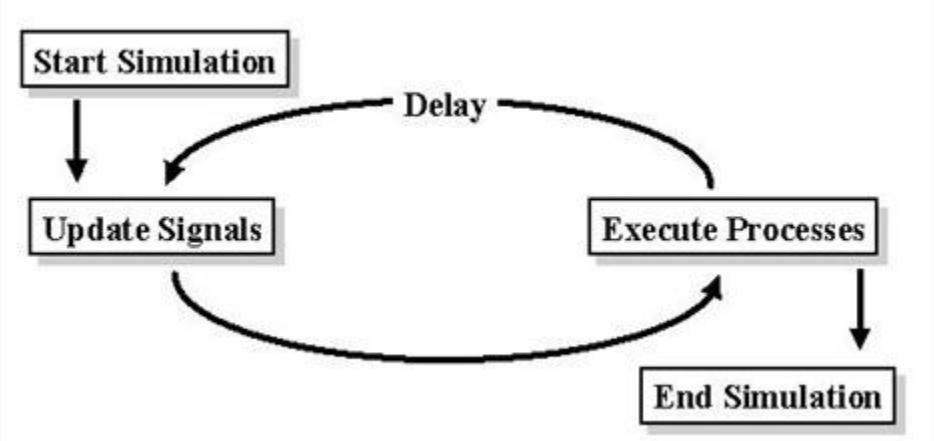
Par Philippe Gorley

Pourquoi cette présentation?

Beaucoup de trouble vis-à-vis le VHDL

Les étudiants aimaient l’idée

Simulation


Start Simulation
Delay
Update Signals
Execute Processes
End Simulation

Syntaxe

- **Identifiants**
- Types
- Opérateurs
- Entité
- Architecture
- “Component”
- Structure
- Signal et variables
- Séquentiel vs concurrent

- Mots clés réservés (signal, architecture, process, loop, etc.)
- Insensible à la casse
- Identifiant valide
  - Séquence de lettres, chiffres, soulignés
  - Commençant par une lettre
  - Se terminant par une lettre ou un chiffre
- Séparateurs
  - Espace blanc (espace, tab, retour chariot)
- Délimiteurs
  - Simple: “ & # ‘ ( ) * + , - . / : ; < = > |
  - Composé: => <= := >= <> -- /= **
- Commentaires introduits par --
- Littéraux
  - Caractère et chaîne de caractères: ‘’ vs “”
  - Chaînes de bits: “0101_1001”, X“BEEF”, O“7701”
  - Nombres: 63, 3e4
- Variable (variable ma_variable : time := 1.0 s;)
  - Restraints à process, procedure, function
- Signal (signal mon_signal : std_logic;)
- Constante (constant ma_const : integer := 2;)

Syntaxe

- Identifiants
- **Types**
- Opérateurs
- Entité
- Architecture
- “Component”
- Structure
- Signal et variables
- Séquentiel vs concurrent

- Fortement typé
- Scalaire
  - Énumération (type lum_trafic is (ROUGE, JAUNE, VERT);)
  - Entiers et sous types
  - Réel et sous types
  - Physique (unités)
- Tableau
  - type lum_intersection is array (0 to 5) of lum_trafic;
  - Affectation par sous plage
    - a1(3 downto 1) <= a2(2 downto 0);
  - Indexation par parenthèses
  - 2 dimensions (tableau de tableaux ou multidimensionnel)

Syntaxe

- Identifiants
- Types
- **Opérateurs**
- Entité
- Architecture
- “Component”
- Structure
- Signal et variables
- Séquentiel vs concurrent

- Plus haute précédance en premier, de gauche à droite
  - ** abs not (‘abs’ et ‘not’ sont unaires)
  - * / mod rem
  - + - (unaires)
  - + - & (binaires, concaténation)
  - sll srl sla sra rol ror
    - sll/srl: remplissage avec ‘0’
    - sla/sra: remplissage avec lsb ou msb
  - = /= < <= > >=
  - and or nand nor xor xnor
  - <= := (affectation de signal et variable, respectivement)
- ‘and’ et ‘or’ sont associatifs, ‘nand’ et ‘nor’ ne le sont pas

Syntaxe

- Identifiants
- Types
- Opérateurs
- **Entité**
- **Architecture**
- **“Component”**
- Structure
- Signal et variables
- Séquentiel vs concurrent

- Libraries (ieee) et packages (std_logic_1164)
- Entité
  - Décrit les ports I/O de la composante
  - Similaire au .h en C
- Architecture
  - Décrit le fonctionnement de l’entité
  - Plusieurs architectures par entité possibles
    - Notion plus avancée
- Component
  - Permet d’utiliser une entité ailleurs
  - Doit être identique à la déclaration de son entité
  - “Port map” par position, index ou nom

Syntaxe

- Identifiants
- Types
- Opérateurs
- Entité
- Architecture
- “Component”
- **Structure**
- Signal et variables
- Séquentiel vs concurrent

- RTL (Register Transfer Level)
  - Décrit le circuit tel quel
- Comportemental (Behavioural)
  - Décrit la fonctionnalité du circuit
  - Semblable à la programmation procédurale
- Structurel
  - N’utilise que des composants existantes
  - “port map”
- NB: Pas mutuellement exclusifs, plusieurs autres

Syntaxe

- Identifiants
- Types
- Opérateurs
- Entité
- Architecture
- “Component”
- Structure
- **Signal et variables**
- Séquentiel vs concurrent

Signal

- Ce que vous voulez utiliser pour ne pas vous casser la tête
- Visible à travers l’architecture
- Déclaration
  - signal *mon_signal* : *type* [:= *valeur*];
- Assignation
  - *mon_signal* <= *expression* [after *expression_temps*];
  - Planifie l’assignation: non immédiat, délai intégré

Variable

- Local au bloc
- Seulement pour les “process”, “procedure” et “function”
- Déclaration
  - variable *ma_variable* : *type* [:= *valeur*];
- Assignation
  - *ma_variable* := *valeur*;
  - Immédiate
- Valeur intermédiaire

Syntaxe

- Identifiants
- Types
- Opérateurs
- Entité
- Architecture
- “Component”
- Structure
- Signal et variables
- **Séquentiel vs concurrent**

Séquentiel (exécuté une ligne à la fois)

- wait
- if then..elsif then..else..end if
- case..when
- next/exit (dans une boucle)
- null
- Séquentiel si dans un process

Concurrent (exécuté en parallèle)

- Instantiation des “component”
- block

Merci

Prochain atelier vers la mi-novembre
Suivez “ÉLE - Page Étudiante - AÉÉTS” sur Facebook pour plus de détails
gorley.philippe@gmail.com
https://github.com/philippegorley/vhdl_workshop