

TME4 -- Modélisation et Simulation VHDL



Contents

Objectifs	1
Exercices	1
Exercice 1 -- Additionneur 4 bits	1
Question 1	1
Question 2	2
Question 3	2
Exercice 2 -- ALU pour l'ARM	2
Question 1	2
Question 2	2
Exercice 3 -- Shifter pour l'ARM	3
Question 1	3

Objectifs

- Un premier objectif est de décrire (en VHDL) et simuler un petit circuit dont le schéma est fourni.
- Dans un second temps, vous allez décrire et simuler deux blocks : l'ALU et le décaleur qui seront utilisés dans le processeur que vous allez réaliser.

Pour simuler vos circuits, vous allez utiliser l'outil **ghdl** et le visualiseur de chronogrames **gtkwave**. Ces deux outils sont sous licence GPL et peuvent donc être utilisés librement par vous.

Exercices

Exercice 1 -- Additionneur 4 bits

Nous vous demandons de décrire en VHDL un petit circuit réalisant l'addition de ses deux entrées chacune codée sur 4 bits. Le résultat produit sera également codé sur 4 bits.

Question 1

Écrivez le fichier `.|VHDL|` correspondant à ce circuit en partant des informations fournies en cours. Cette description doit être de type *zéro-délai* (pas de clause *After*).

Vous allez pouvoir vérifier la syntaxe et analyser votre fichier grâce à la commande suivante :

```
etudiant@pc:~> ghdl -a -v adder.vhdl
```

Si l'analyse de votre circuit a réussi vous devez avoir obtenu un fichier `adder.o` correspondant au résultat de la compilation.

Question 2

Une fois la description comportementale compilée avec succès (sans aucune erreur), pour valider votre description vous devez écrire un autre fichier VHDL pour tester votre circuit. Dans ce fichier que l'on appelle couramment *test bench*, vous allez d'une part instancier votre circuit à tester et d'autre part écrire un *process* dont le rôle va être de fournir des valeurs pour les entrées du circuit à tester et éventuellement vérifier la valeur des sorties.

Compilez votre fichier *test bench* comme vous l'avez déjà fait avec votre circuit *adder.vhdl* puis réalisez l'élaboration de votre projet.

Commencez par quelques test simples de votre circuit :

```
etudiant@pc:~> ghdl -a -v adder_tb.vhdl
etudiant@pc:~> ghdl -e -v adder_tb
```

Simulez votre circuit :

```
etudiant@pc:~> ghdl -r adder_tb --vcd=adder.vcd
```

Visualisez avec **gtkwave** le résultat de votre simulation.

Question 3

Modifiez votre fichier *test bench* pour effectuer un test exhaustif de votre circuit, rendez-le auto-testant (ne plus visualiser les résultats avec **gtkwave**).

Exercice 2 -- ALU pour l'ARM

Comme vous allez le voir dans les semaines à venir on peut simplifier cette ALU en restreignant ses fonctionnalités à 4 opérations de base : ADD avec retenue, AND, OR et XOR.

Cette ALU reçoit ses deux opérandes codés sur 32 bits et fournit un résultat sur 32 bits. Elle reçoit également en entrée une retenue (*cin*) et fournit en sortie les 4 flags : Z, N, V et Cout. Cette ALU est exclusivement combinatoire, voici son interface :

```
entity Alu is
  port (
    op1  : in  Std_Logic_Vector(31 downto 0);
    op2  : in  Std_Logic_Vector(31 downto 0);
    cin  : in  Std_Logic;
    cmd  : in  Std_Logic_Vector(1 downto 0);
    res  : out Std_Logic_Vector(31 downto 0);
    cout : out Std_Logic;
    z    : out Std_Logic;
    n    : out Std_Logic;
    v    : out Std_Logic;
    vdd  : in  bit;
    vss  : in  bit );
end Alu;
```

Les entrées *cmd* permettent de spécifier l'opération à effectuer, une seule d'entre elles doit être égale à 1, les autres doivent être égales à 0.

Question 1

Écrivez le modèle *.vhdl* de l'ALU.

Question 2

Écrivez un *test bench* permettant de tester cette ALU. Vous pourrez utiliser un générateur pseudo-aléatoire pour générer des valeurs quelconques sur 32 bits.

Exercice 3 -- Shifter pour l'ARM

Pour notre processeur ARM nous allons avoir besoin d'un composant dédié à la réalisation des décalages et rotations. Ce Shifter est exclusivement combinatoire, voici son interface :

```
entity Shifter is
  port (
    shift_lsl : in Std_Logic;
    shift_lsr : in Std_Logic;
    shift_asr : in Std_Logic;
    shift_ror : in Std_Logic;
    shift_rrx : in Std_Logic;
    shift_val : in Std_Logic_Vector(4 downto 0);
    din       : in Std_Logic_Vector(31 downto 0);
    cin      : in Std_Logic;
    dout     : out Std_Logic_Vector(31 downto 0);
    cout     : out Std_Logic;
    -- global interface
    vdd      : in bit;
    vss      : in bit );
end Shifter;
```

Question 1

Écrivez son modèle en `.vhd1` et le `test bench` permettant de le tester.