TP1 : Modélisation VHDL Data-Flow

Objectifs

Le but de cette séance est de modéliser en VHDL "data-flow", puis de simuler d'un petit circuit très simple : un additionneur accumulateur que nous appellerons *addaccu*. Vous trouverez la DATA SHEET de ce circuit en annexe.

L'outil de la chaîne de CAO *ALLIANCE* utilisé aujourd'hui est le compilateur/simulateur VHDL **asimut**.

Vous pouvez obtenir des informations détaillées sur les outils de la chaîne *ALLIANCE* en tapant la commande :

```
> man asimut
```

Pour utiliser cette commande, il faut définir le chemin d'accès aux fichiers *man* dans la variable d'environnement UNIX MANPATH. En bash il faut lancer la commande:

```
> export MANPATH=$MANPATH:/asim/alliance/man
```

Le déroulement de ce premier TP comporte trois étapes: Modélisation comportementale du circuit *adaccu*, description des stimuli en entrée et des résultats attendus en sortie, et simulation effective avec le simulateur **asimut**.

A) Modélisation zero-delay du circuit addaccu

Une description comportementale "data-flow" est une description qui ne contient que des assignations concurrentes. Ces descriptions peuvent contenir des informations temporelles :

```
c <= a AND b  AFTER 300ps;
```

Puisque la sémantique du langage VHDL repose sur les *événements*, l'assignation ci-dessus signifie qu'un événement (c'est à dire un changement de valeur à une date t) sur le signal a ou sur le signal b peut entraîner un événement sur le signal c à la date t + 300 ps. La construction AFTER peut être omise, ce qui correspond à un temps de propagation nul. Les deux écritures ci-desous sont donc équivalentes :

```
x <= NOT y;
x <= NOT y AFTER 0 ps;
```

Une description comportementale qui ne contient pas de constructions AFTER est une description dite "ZERO-DELAY": on considère que tous les temps de propation sont négligeables par rapport au temps de cycle. C'est donc une description où on cherche à valider la fonctionnalité logique, mais où on ne cherche pas à simuler les performances temporelles.

Ecrire une description comportementale "zéro-delay" pour le composant *addaccu*. Le fichier comportera l'extension ".vbe" qui est l'extension usuelle pour indiquer un fichier VHDL comportemental de type data-flow (.vbe pour Vhdl

BEhaviour).

Vous pourrez vérifier que votre fichier *addaccu.vbe* est syntaxiquement correct en utilisant l'outil **asimut** pour compiler ce fichier sans lancer la simulation.

```
>asimut - b -c addaccu
```

- l'option -b indique qu'il s'agit d'une description purement comportementale (extension .vbe), car **asimut** accepte également en entrée des descriptions structurelle (dans ce ca le fichier possède l'extension .vst).
- l'option -c permet de compiler sans simuler.

Comme tous les outils de la chaîne ALLIANCE*, **asimut** utilise des variables d'environnement UNIX. La signification des 8 variables d'environnement utilisées par **asimut** est à découvrir dans la page man de l'outil **asimut**. Si l'on ne désire pas utiliser les valeurs par défaut, ces variables d'environnement peuvent être modifiées en utilisant les commandes suivantes:*

```
>export VH_MAXERR=10
>export VH_BEHSFX=vbe
>export VH_PATSFX=pat
>export VH_DLYSFX=dly
>export MBK_WORK_LIB=.
>export MBK_CATA_LIB=.
>export MBK_CATA_NAME=CATAL
>export MBK_IN_LO=vst
```

B) Définition des stimuli

Lorsque la description comportementale du composant *addaccu* compile sans erreurs, il faut écrire le fichier décrivant les stimuli qui vont être appliqués sur les entrées du circuit. Ce fichier doit également contenir les valeurs attendues sur les sorties du circuit.

Le format du fichier de stimuli accepté par *asimut* est défini dans la page man du format .pat :

```
>man pat
```

Ce fichier *stimuli.pat* doit en particulier décrire le signal d'horloge. On choisira un signal périodique de période 10 ns, et de rapport cyclique 50% (cela signifie que l'état bas et l'état haut ont des durées égales).

Les valeurs de tous les signaux d'entrée doivent être définies depuis le temps initial t = 0ns. Vous utiliserez la construction *spy* pour définir les différents signaux que vous souhaitez observer: on peut demander au simulateur de tracer les signaux de sortie, mais également les signaux internes présents dans la description VHDL. On commencera par un scénario simple comportant une dizaine de cycles.

Attention : Comme les registres internes au composant *addaccu* sont des bascules à échantillonnage sur front montant, les signaux d'entrée ne doivent pas changer de valeur lorsque le signal d'horloge passe de 0 à 1.

Vous pouvez vérifier le chronogramme défini par votre fichier *stimuli.pat*, en utilisant l'outil de visualisation graphique **xpat**:

```
>xpat -l stimuli
```

C) Simulation logique (zero-delay)

Vous pouvez maintenant lancer la simulation sous **asimut**, en applicant les stimuli définis dans le fichier *stimuli.pat* sur la description comportementale définie dans le fichier *addaccu.vbe*:

```
>asimut -b addaccu stimuli result
```

Le dernier argument sur la ligne de commande est le nom du fichier (de format .pat), contenant - en plus des stimuli sur les entrées - les valeurs des signaux calculées par le simulateur.

D) Simulation logico-temporelle

Pour terminer ce TP, vous allez maintenant complêter la description comportementale du composant *addaccu*, en attachant des temps de propagation aux différents opérateurs combinatoires. Plus précisément, vous allez utiliser la construction AFTER pour préciser les valeurs des temps de propagation de l'additionneur et du multiplexeur.

Relancer la simulation pour les trois cas suivants :

	cas 1	cas 2	cas 3
additionneur	1.0 ns	3.0 ns	5.0 ns
multiplexeur	0.2ns	0.5 ns	1.0 ns

Comment interprêtez-vous ces résultats ?

Compte-Rendu

Vous rédigerez un compte-rendu d'une page maximum pour ce TP, **mais sans inclure les fichiers sources**.