Professeur : Etienne Messerli
Assistant : Anthony Convers

Laboratoire 05 – Conception d'une interface fiable

Objectifs du laboratoire

Ce laboratoire a pour but de concevoir une interface fiable sur le bus Avalon connecté sur le système à processeur HPS (hard processor system) avec l'objectif de mettre en évidence les différences de comportement entre le logiciel (séquentiel) et le matériel (concurrent). Vous disposerez d'un générateur de 4 nombres dont l'équation « na+nb+nc=nd » doit toujours être valide ! La première partie montrera qu'une lecture séquentielle de ces 4 nombres peut impliquer une incohérence entre les valeurs lues (na+nb+nc ≠ nd). Dans une seconde partie, vous devrez concevoir une interface permettant de garantir une acquisition correcte des 4 nombres. L'objectif est d'acquérir ces valeurs ayant toute la même référence de temps (en anglais : timestamp). Il s'agira de prendre une "photo instantanée" des 4 nombres, même si le temps de transfert sera différent pour chacune d'entre eux. La vérification est obtenue en calculant la somme « na+nb+nc » qui doit être égale à nd.

Spécifications

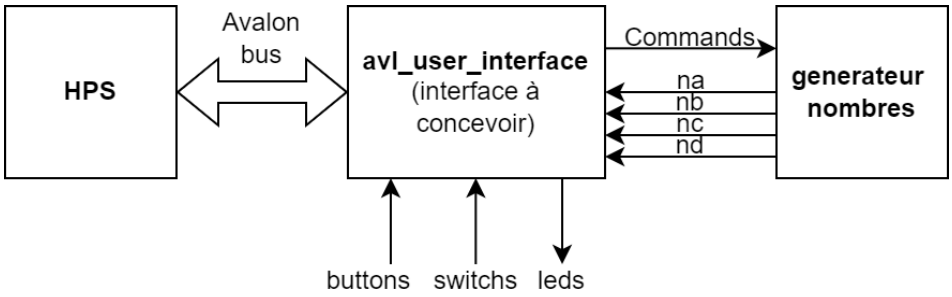
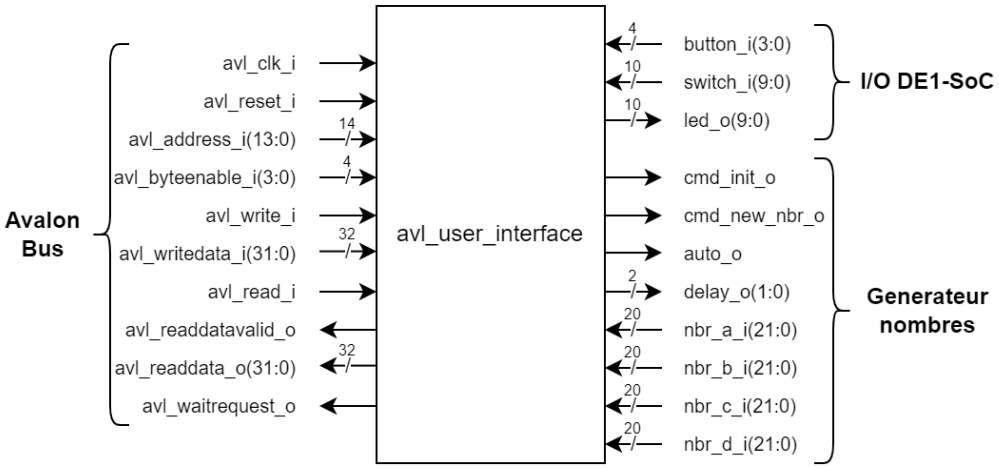


Schéma du system

Le composant VHDL "avl_user_interface.vhd" sera utilisée pour implémenter l'interface connecté sur le bus Avalon afin de communiquer avec le générateur de nombres et les différents périphériques.

Voici le symbole du composant avl_user_interface :



Symbole du composant avl_user_interface

L'interface développée doit permettre d'accéder, à travers le bus Avalon, aux périphériques suivants :

- Interface constant ID sur 32 bits (disponible à l'offset 0)
- 4 Boutons (Key) DE1-SoC
- 10 Interrupteurs (Switch) DE1-SoC
- 10 Leds DE1-SoC
- Générateur de nombres (Composant dans la partie FPGA de la DE1-SoC). Détails donnés ci-après.

Générateur de nombres

Ce composant est un générateur de 4 nombres aléatoires qui répondent en tout temps à l'équation suivante :

- nbr_a + nbr_b + nbr_c = nbr_d

Chaque nombre est sur 22 bits avec une représentation non-signée.

Le générateur de nombre répond à différentes commandes qui permettent : réinitialiser les 4 nombres, générer des nombres en mode automatique ou manuel, sélectionner la fréquence de génération des nombres. Le tableau ci-dessous décrit les signaux d'entrées du composant qui correspond aux différentes commandes.

Noms des signaux	**Description**
cmd_init_i	Lorsque le signal est actif ('1'), les 4 nombres sont initialisés à zero. Commande prioritaire.
auto_i	Sélection du mode pour la génération des nombres : - '0' : mode manuel - '1' : mode automatique
cmd_new_nbr_i	Un front montant sur ce signal génère un nouvel ensemble de 4 nombres, fonctionne seulement dans le mode manuel.
delay_i (1..0)	Sélection de la fréquence de génération des nombres, fonctionne seulement dans le mode automatique : - "00" : 1 Hz - "01" : 1 KHz - "10" : 100 KHz - "11" : 1 MHz

Plan d'adressage

Voici le plan d'adressage qui est spécifié pour le bus AXI lightweight HPS-to-FPGA du projet fourni.

Offset on bus AXI lightweight HPS-to-FPGA (relative to BA_LW_AXI)	**Fonctionnalités**
0x00_0000 – 0x00_0003	Constante ID 32 bits (Read only)
0x00_0004 - 0x00_FFFF	reserved
0x01_0000 - 0x01_FFFF	Zone disponible pour votre interface
0x02_0000 - 0x1F_FFFF	not used

Voici le plan d'adressage proposé pour votre interface.

Adresse (offset)	**Read**	**Write**
0x00	[31..0] Constante ID interface	not used
0x04	[31..4] "0..0" ; [3..0] buttons	not used
0x08	[31..10] "0..0" ; [9..0] switchs	not used
0x0C	[31..10] "0..0" ; [9..0] leds	[31..10] reserved ; [9..0] leds
0x10	[31..2] "0..0" ; [1-0] status	[31..5] reserved ; [4] new_nbr [3..1] reserved ; [0] init_nbr
0x14	[31..5] "0..0" ; [4] mode_gen [3-2] "0..0" ; [1-0] delay_gen	[31..5] reserved ; [4] mode_gen [3-2] reserved ; [1-0] delay_gen
0x18	available for news functionality	available for news functionality
0x1C	available for news functionality	available for news functionality
0x20	[31..24] "0..0" [23-22] "00" code nbr_a [21..0] valeur nbr_a	not used
0x24	[31..24] "0..0" [23-22] "01" code nbr_b [21..0] valeur nbr_b	not used
0x28	[31..24] "0..0" [23-22] "10" code nbr_c [21..0] valeur nbr_c	not used
0x2C	[31..24] "0..0" [23-22] "11" code nbr_d [21..0] valeur nbr_d	not used
0x30 … 0x3C	reserved	reserved
0x40 … 0xFFFC	not used	not used

Explications sur le plan d'adressage :

- **status** : Deux bits de statut, placés aux bits [1..0] (Ils seront utilisés lors de la seconde partie du laboratoire) :
  - status[1] : indique si l'interface dispose d'un système de capture des 4 nombres (photo instantanée). Si status[1] = '1' système capture disponible.
  - status[0] : indique qu'une photo a été prise lorsque le système de capture est actif. Ce bit est valide uniquement si status[1] = '1'.
- **init_nbr** : Commande pour initialiser les 4 nombres à zéro (prioritaire).
- **new_nbr** : Commande pour générer un nouvel ensemble de 4 nombres (mode manuel)
- **mode_gen** :Sélection du mode automatique ou manuel pour la génération des nombres.
- **delay_gen** : Sélection de la fréquence de génération des nombres (mode automatique).
- **nbr_x** : lecture du nombre avec son code correspondant :
  - code [23-22] : identifiant du nombre sur 2 bits.
  - valeur [21..0] : valeur sur 22 bits en représentation non-signée.

Spécifications du programme

Le but est de contrôler le générateur de nombres selon l'états des boutons et interrupteurs de la DE1-SoC, et de lire les valeurs des 4 nombres. Il faudra également vérifier la cohérence des 4 nombres lues. La spécification du fonctionnement est la suivante :

Au démarrage, le programme doit remplir les conditions suivantes :

- Les 10 leds DE1-SoC sont éteintes.
- Afficher la constante ID du bus AXI lightweight HPS-to-FPGA au format hexadécimal dans la console de ARM-DS.
- Afficher la constante ID de votre interface sur le bus Avalon au format hexadécimal dans la console de ARM-DS.

Ensuite pendant l'exécution du programme, à tout instant les actions suivantes doivent être respectées :

- Copie de la valeur des 10 interrupteurs (SW) sur les 10 leds de la DE1-SoC.
- Une pression sur KEY0 permet de l'initialisation des 4 nombres à zéro.
- Une pression sur KEY1 permet de générer un nouvel ensemble de 4 nombres, cela fonctionne seulement lorsque le mode manuel est sélectionné.
- Tant que KEY2 est actif, lecture successive des 4 nombres, puis calcul de la somme des nombres pour vérifier la cohérence, et affichage dans la console :
  - Si les 4 nombres sont cohérents, donc correct, affichage du message :
    - OK : status: X , somme: X, nbr_a: X, nbr_b: X, nbr_c: X, nbr_d: X,
  - Si les 4 nombres ne sont pas cohérents, donc incorrect, incrémentation d'un compteur du nombre d'erreur cumulée et affichage du message :
    - ER : status: X , somme: X, nbr_a: X, nbr_b: X, nbr_c: X, nbr_d: X,
    - ER : nombre d'erreur cumulée : X
- L'état de SW9-8 permet de sélectionner la fréquence de génération des nombres :
  - SW9-8 = 00 : 1 Hz

  - SW9-8 = 01 : 1 KHz
  - SW9-8 = 10 : 100 KHz
  - SW9-8 = 11 : 1 MHz
- L'état de SW7 permet de sélectionner le mode de génération des nombres :
  - SW7 = 0 : mode manuel
  - SW7 = 1 : mode automatique
- L'état de SW0 permet de sélectionner une acquisition fiable (implémenter seulement dans la partie 2) :
  - SW0 = 0 : acquisition non fiable
  - SW0 = 1 : acquisition fiable garantie

À rendre

Ce laboratoire est évalué. Il y a un rapport à rédiger à l'issu de ce laboratoire contenant les explications sur les différentes étapes de la réalisation de votre système. Vous devez rendre une archive avec les sources du projet pour Quartus et le programme C. Utiliser le Makefile à la racine du projet pour générer votre archive à rendre en tapant « make zip » dans un terminal.

Les fichiers sont à rendre sur Cyberlearn à la date indiquée.

Travail demandé

1ère partie :

L'objectif de cette 1ère partie est de démontrer que la lecture séquentielle des 4 nombres peut impliquer une incohérence entre ceux-ci.

1) Concevoir votre interface en VHDL dans le composant "avl_user_interface" afin de répondre au plan d'adressage proposé. Celui-ci doit permettre de répondre à toutes les spécifications décrites précédemment.
   Dans cette première partie, les bits de status[1-0] seront mis à "00" (fixe).
2) Simuler votre interface à l'aide de la console TCL/TK. Vérifier les bons accès et commandes avec le générateur de nombres.
3) Puis, tester votre interface sur la carte DE1-SoC. Vous utiliserez l'accès mémoire pour vérifier le fonctionnement de celle-ci.
4) Ecrire des fonctions C pour les accès aux différents périphériques/fonctionnalités. Puis écrire le programme principal suivant les spécifications.
5) Tester vos fonctions et le programme sur la carte DE1-SoC.
   Vous devez voir que les 4 nombres lus ne sont pas toujours cohérents selon la formule. Vous devez expliquer pourquoi ?
6) Faire valider votre première partie par le professeur ou l'assistant.
   Sauvegarder une copie de votre travail (.hdl + .c) correspondant à la partie 1.

2ième partie :

La seconde partie de la manipulation consiste à modifier l'interface afin d'ajouter la possibilité de réaliser une acquisition correcte (fiable) des 4 nombres. Cette fonctionnalité sera active suivant l'état du switch SW0. Votre solution doit permettre d'avoir les 2 modes d'acquisition disponible. Ensuite, si KEY2 est actif, votre système doit lire de façon répété les nombres selon le mode d'acquisition choisi par SW0.
Dans cette partie les 2 bits de status[1-0] seront utilisés.
Les fonctionnalités créées en 1ère partie doivent être conservées.

7) Concevoir une solution permettant de réaliser une « photo » des 4 nombres sur commande **du HPS**. Expliquer votre solution et la répartition du système entre le logiciel et le matériel. Donner les adaptations nécessaires au plan d'adressage.
8) Réaliser les modifications nécessaires dans votre interface et réaliser si nécessaire une simulation ou un test avec l'accès mémoire de ARM DS.
9) Modifier votre programme selon votre solution conçue au point 7.
10) Tester votre solution et démontrer, pour n'importe quelle fréquence de génération, que la lecture des 4 nombres est cohérente.
11) Faire valider votre deuxième partie par le professeur ou l'assistant.
    Vous devez proposer une suite de tests permettant de démontrer que votre solution est fiable.