Laboratoire #1 : Circuits séquentiels

INF3500 - Conception et réalisation de systèmes numériques
Hiver 2019

Objectifs

Ce laboratoire a les objectifs suivants :

— vous familiariser avec les outils utilisés lors des laboratoires du cours INF3500 ;
— vous faire implémenter un circuit séquentiel ; et
— vous faire utiliser le concept de machine à état dans un circuit séquentiel.

Préparation du laboratoire

Avant d'arriver au laboratoire, suivez les étapes suivantes :

1. revoir la matière des cours des semaines 4 et 3 ; et,
2. lire le guide pratique d'utilisation de Vivado ; et,
3. lire la FAQ du cours.

Le guide pratique d'utilisation de Vivado est un tutoriel présentant les étapes de simulation, de synthèse, d'implémentation et de la programmation du FPGA dans l'environnement de développement Xilinx Vivado.

Contexte : circuits séquentiels

Les architectures utilisant des circuits séquentiels sont très répandues dans le domaine des circuits numérique. Par exemple, les circuits séquentiels sont utilisés dans les processeurs à usage général, les processeurs graphiques, les modules spécialisés dans le cryptominage, les unités de calculs dans un système de contrôle d'un drone, etc.

De tels circuits numériques sont organisés en blocs de logique combinatoire séparés par de la logique séquentielle.

Par exemple, les machines à état, qui sont utilisées dans de nombreux circuits numériques, sont implémentées avec de la logique séquentielle, utilisées pour enregistrer l'état actuel dans une mémoire (une *bascule*), et avec de la logique combinatoire pour calculer la valeur de l'état futur.

Partie A : Implémentation de la boucle principal de l'algorithme SHA-256

Dès que vous avez complète dans TP1 les fonctions de compression de l'algorithme SHA256, vous devez maintenant les utiliser pour implementer la boucle principale de l'algorithme. Le pseudo-code

suivant décrit l'algorithme. Vous devez implementer une *machine à états* en VHDL synthétisable pour générer le circuit équivalent. Les variables internes ainsi que l'entree sont exprimés sur 32 bits. La sortie est exprimée sur 256 bits.

```
const K = 0x428a2f98
pendant toujours
       si debut_systeme
              H_0^(0) = 0x6a09e667
              H_1^(0) = 0xbb67ae85
              H_2^(0) = 0x3c6ef372
              H_3^(0) = 0xa54ff53a
              H_4^(0) = 0x510e527f
              H_5^(0) = 0x9b05688c
              H_6^(0) = 0x1f83d9ab
              H_7^(0) = 0x5be0cd19
              W = 0x00000000
       sinon
              si nouvelle_entree
                     // Initialise les variables
                     a = H_0^(t-1)
                     b = H_1^(t-1)
                     c = H_2^(t-1)
                     d = H_3^(t-1)
                     e = H_4^(t-1)
                     f = H_5^(t-1)
                     g = H_6^(t-1)
                     h = H_7^(t-1)
                     W = entree
                     sortie_valid = 0

                     // Boucle principal
                     T_1 = h + Σ_1^{256}(e) + Ch(e,f,g) + K + W
                     T_2 = Σ_0^{256}(a) + Maj(a,b,c)
                     h = g
                     g = f
                     f = e
                     e = d + T_1
                     d = c
                     c = b
                     b = a
                     a = T_1 + T_2

                     // Mettre a jour le hash
                     H_0^(t) = a + H_0^(t-1)
                     H_1^(t) = b + H_1^(t-1)
                     H_2^(t) = c + H_2^(t-1)
                     H_3^(t) = d + H_3^(t-1)
                     H_4^(t) = e + H_4^(t-1)
                     H_5^(t) = f + H_5^(t-1)
                     H_6^(t) = g + H_6^(t-1)
                     H_7^(t) = h + H_7^(t-1)


                     // Sortie
                     sortie_valid = 0
                     sortie = H_0^(t)H_1^(t)H_2^(t)H_3^(t)H_4^(t)H_5^(t)H_6^(t)H_7^(t)

              fin si
       fin si
fin pendant
```

Les tâches à réaliser pour ce laboratoire sont présentées ci-dessous :

1. Proposez un circuit séquentiel afin d'implementer l'algorithme.
2. réalisez la description de votre module en VHDL synthétisable.

3. Validez avec les charges de TP le diagramme d'etats de votre machine à états.

Points à considérer :

1. attention à la logique de reset.
2. Rappelez-vous que dans un circuit sequential la valeur dans des bascules sont mises à jour au cycle suivant.
3. attention au dependence de données.

À remettre : le code de votre module (ne pas ajouter les codes complets dans le rapport) et le diagramme d'états (validé) de votre machine à états.

Partie B : Simulation

Veuillez générer un banc d'essai par modèle de référence disponible (référence TP1) a l'aide d'un programme logiciel (langage de votre préférence) dont vous devez reproduire le pseudo-code donné. Utilisez des énoncés "assert" pour valider le bon fonctionnement de votre circuit.

À remettre : le code votre banc d'essai ainsi que le code utilise pour générer le modèle de référence (ne pas ajouter les codes complets dans le rapport).

Partie C : Implémentation

Faites l'implémentation et la démonstration sur la carte Nexys 4 en respectant les spécifications ci-dessous :

1. Utiliser un style de description de type structurale.
2. Créez un module "top" contenant votre module.
3. Configurez votre module "top" avec une entrée encodée sur 16 bits provenant des commutateurs de la carte.
4. Votre module reçoit un signal 'entree' dont les 16 bits de poids fort sont égaux à 0, et les 16 bits de poids faible proviennent des commutateurs de la carte.
5. Reliez les sorties (16 bits de poids faible) aux DELs.

Veuillez utiliser le fichier de contrainte présenté `top.xdc`, dont le contenu est présenté dans la figure ci-dessous, afin de sélectionner les PINs physiques des commutateurs de la carte comme entrée de votre module "top".

```
# top.xdc
# pour carte Digilent Nexys 4 DDR
## Clock signal
set_property -dict { PACKAGE_PIN E3 IOSTANDARD LVCMOS33 } [get_ports { CLK100MHZ }];
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports {CLK100MHZ}];

##Buttons
set_property -dict { PACKAGE_PIN N17 IOSTANDARD LVCMOS33 } [get_ports { reset }];
set_property -dict { PACKAGE_PIN P17 IOSTANDARD LVCMOS33 } [get_ports { entree_valide }];


## Detecteurs = Switches
## Entrees

set_property -dict { PACKAGE_PIN J15 IOSTANDARD LVCMOS33 } [get_ports { entree[0] }];
set_property -dict { PACKAGE_PIN L16 IOSTANDARD LVCMOS33 } [get_ports { entree[1] }];
set_property -dict { PACKAGE_PIN M13 IOSTANDARD LVCMOS33 } [get_ports { entree[2] }];
set_property -dict { PACKAGE_PIN R15 IOSTANDARD LVCMOS33 } [get_ports { entree[3] }];
set_property -dict { PACKAGE_PIN R17 IOSTANDARD LVCMOS33 } [get_ports { entree[4] }];
set_property -dict { PACKAGE_PIN T18 IOSTANDARD LVCMOS33 } [get_ports { entree[5] }];
set_property -dict { PACKAGE_PIN U18 IOSTANDARD LVCMOS33 } [get_ports { entree[6] }];
set_property -dict { PACKAGE_PIN R13 IOSTANDARD LVCMOS33 } [get_ports { entree[7] }];
```

```
set_property -dict { PACKAGE_PIN T8 IOSTANDARD LVCMOS18 } [get_ports { entree[8] }];
set_property -dict { PACKAGE_PIN U8 IOSTANDARD LVCMOS18 } [get_ports { entree[9] }];
set_property -dict { PACKAGE_PIN R16 IOSTANDARD LVCMOS33 } [get_ports { entree[10] }];
set_property -dict { PACKAGE_PIN T13 IOSTANDARD LVCMOS33 } [get_ports { entree[11] }];
set_property -dict { PACKAGE_PIN H6 IOSTANDARD LVCMOS33 } [get_ports { entree[12] }];
set_property -dict { PACKAGE_PIN U12 IOSTANDARD LVCMOS33 } [get_ports { entree[13] }];
set_property -dict { PACKAGE_PIN U11 IOSTANDARD LVCMOS33 } [get_ports { entree[14] }];
set_property -dict { PACKAGE_PIN V10 IOSTANDARD LVCMOS33 } [get_ports { entree[15] }];

## Sorties

set_property -dict { PACKAGE_PIN H17 IOSTANDARD LVCMOS33 } [get_ports { sortie[0] }];
set_property -dict { PACKAGE_PIN K15 IOSTANDARD LVCMOS33 } [get_ports { sortie[1] }];
set_property -dict { PACKAGE_PIN J13 IOSTANDARD LVCMOS33 } [get_ports { sortie[2] }];
set_property -dict { PACKAGE_PIN N14 IOSTANDARD LVCMOS33 } [get_ports { sortie[3] }];
set_property -dict { PACKAGE_PIN R18 IOSTANDARD LVCMOS33 } [get_ports { sortie[4] }];
set_property -dict { PACKAGE_PIN V17 IOSTANDARD LVCMOS33 } [get_ports { sortie[5] }];
set_property -dict { PACKAGE_PIN U17 IOSTANDARD LVCMOS33 } [get_ports { sortie[6] }];
set_property -dict { PACKAGE_PIN U16 IOSTANDARD LVCMOS33 } [get_ports { sortie[7] }];
set_property -dict { PACKAGE_PIN V16 IOSTANDARD LVCMOS33 } [get_ports { sortie[8] }];
set_property -dict { PACKAGE_PIN T15 IOSTANDARD LVCMOS33 } [get_ports { sortie[9] }];
set_property -dict { PACKAGE_PIN U14 IOSTANDARD LVCMOS33 } [get_ports { sortie[10] }];
set_property -dict { PACKAGE_PIN T16 IOSTANDARD LVCMOS33 } [get_ports { sortie[11] }];
set_property -dict { PACKAGE_PIN V15 IOSTANDARD LVCMOS33 } [get_ports { sortie[12] }];
set_property -dict { PACKAGE_PIN V14 IOSTANDARD LVCMOS33 } [get_ports { sortie[13] }];
set_property -dict { PACKAGE_PIN V12 IOSTANDARD LVCMOS33 } [get_ports { sortie[14] }];
set_property -dict { PACKAGE_PIN V11 IOSTANDARD LVCMOS33 } [get_ports { sortie[15] }];
```

Partie D : Analyse et Discussion

Veuillez remplir le tableau suivant et discuter les résultats.

LUT	Bascules	Fréquence [MHz]	Latence [ns]	Débit [operations/s]

Rapport

Rédigez votre rapport selon les directives demandées ici.

Barème

Critère	points
Partie A : conception du module	8
Partie B : banc d'essai et simulation	4
Partie C : implémentation	4
Partie D : analyse et Discussion	2
Rapport : présentation et qualité du français	2
Total	20