Laboratoire #1 : Circuits combinatoires

INF3500 - Conception et réalisation de systèmes numériques
Hiver 2019

Objectifs

Ce laboratoire a les objectifs suivants :

— approfondir l'apprentissage du langage VHDL ;
— pratiquer la description d'un circuit combinatoire ;
— pratiquer la description structurale et par flot de données des modules
— pratiquer la description et l'utilisation d'un banc d'essai pour un circuit combinatoire ;
— implémenter un circuit combinatoire et vérifier le fonctionnement sur FPGA ;

Préparation du laboratoire

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

1. revoir la matière des cours des semaines 1, 2 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

Les cryptomonaies, telles que Bitcoin, sont des technologies qui agitent souvent les médias, notamment en raison de leur volatilité. D'un point de vue technique, Bitcoin est une technologie qui utilise des algorithmes empruntés à la cryptographie, notamment afin de valider les transactions.

Dans ce laboratoire on s'intéresse à la fonction de hachage SHA-256, utilisées notamment lors de la validation des transactions Bitcoin. SHA-256 est une fonction de hachage de la famille des SHA-2 ("Secure Hash Algorithm 2"), générant une "emprunte", ou "haché", de 256 bits. Ainsi, en appliquant l'algorithme SHA-256 sur un mot encodé sur une longueur $l \leq 2^{64}$ bits, une empreinte de 256 bits est générée. La spécification complète de l'algorithme SHA-256 est disponible ici. La lecture de ce document n'est pas requise pour ce laboratoire, mais contient l'ensemble des définitions de l'algorithme.

L'algorithme SHA-256 calcule l'empreinte d'un mot en plusieurs itérations. À chacune des itérations, des fonctions dites de "compressions" sont appliquées sur une fraction du mot initial. Dans le cadre de ce laboratoire, vous allez implémenter les fonctions de compressions uniquement. L'implémentation complète de l'algorithme SHA-256 sera l'objet d'un autre laboratoire.

Partie A : Implémentation des fonctions de compression de l'algorithme SHA-256

Les six fonctions de compression utilisées dans l'algorithme SHA-256 sont décrites dans les équations 1-6, et appliquées sur des mots x, y, z encodés sur 32 bits.

$$Ch(x,y,z) = (x \wedge y) \oplus (\neg x \wedge z) \tag{1}$$

$$Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \tag{2}$$

$$\Sigma_0^{\{256\}}(x) = ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x) \tag{3}$$

$$\Sigma_1^{\{256\}}(x) = ROTR^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x) \tag{4}$$

$$\sigma_0^{\{256\}}(x) = ROTR^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x) \tag{5}$$

$$\sigma_1^{\{256\}}(x) = ROTR^6(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x) \tag{6}$$

Les opérateurs utilisés dans les fonctions de compression sont des **opérateurs binaires** définis dans le tableau 1.

FIGURE 1 – Opérateurs binaires

Symbole	Opérateur
$\wedge$	ET
$\vee$	OU
$\neg$	NON
$\oplus$	XOR

Par ailleurs les fonctions rotation à droite $ROTR^n(x)$ et décalage à droite $SHR^n(x)$ sont présentées dans les équations 7 et 8.

$$ROTR^n(x) = (x << (32-n)) \vee (x >> n) \tag{7}$$

$$SHR^n(x) = (x >> n) \tag{8}$$

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

1. Proposez un circuit combinatoire par fonctions de compression en utilisant un style de description par flot de données.
2. Réalisez la description de votre module en VHDL synthétisable.
3. Réalisez un banc d'essai et simulez vos modules dans Vivado. Des valeurs de référence à évaluer ainsi que le résultat attendu vous sont fournies.
4. À remettre : les codes de vos modules et de votre banc d'essai (ne pas ajouter les codes complets dans le rapport).

Partie B : Simulation

Veuillez compléter le banc d'essai par modèle de référence disponible ici afin de tester votre circuit contre des résultats déjà validés par logiciel. Utilisez des énoncés "assert" pour valider le bon fonctionnement de votre circuit.

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 les 6 fonctions de compression cascadé, en respectant l'ordre $\sigma_1^{\{256\}} -> \sigma_0^{\{256\}} -> \Sigma_1^{\{256\}} -> \Sigma_0^{\{256\}} -> Maj -> Ch$.
3. Lorsque plusieurs entrées sont requises à une fonction, réutilisez plusieurs fois la sortie de la fonction précédente.
4. Configurez votre module "top" avec une entrée encodée sur 16 bits provenant des commutateurs de la carte.
5. Le module $\sigma_1^{\{256\}}$ reçoit un signal 'x' dont les 16 bits de poids fort sont égaux à 0, et les 16 bits de poids faible proviennent des commutateurs de la carte.
6. Reliez les sorties 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
## 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] }];
```

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
Discussion	2
Rapport : présentation et qualité du français	2
Total	20