Laboratoire #4 : Circuits Arithmétiques

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 arithmétique ; et
— vous faire vous familiariser avec la notion d'optimisation matériel.

Préparation du laboratoire

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

1. revoir la matière des cours des semaines 6 a 10 ; 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 arithmetiques

Les architectures utilisant des circuits arithmétiques sont très utilisées dans le domaine des circuits numériques. Par exemple, les circuits arithmétiques sont utilisés dans les ALUs des processeurs à usage général, les processeurs graphiques, les processeurs de traitement de signal, les modules spécialisés dans le cryptominage, les unités de calculs dans un système de contrôle d'un drone, etc.

Partie A : Implémentation de l'algorithme SHA-256

Ce laboratoire est la continuité du laboratoire #2, dans lequel vous avez implémenté la boucle principale de l'algorithme SHA256. Dans ce laboratoire, vous allez implementer la partie complémentaire de l'algorithme SHA256. Le pseudo-code suivant décrit l'algorithme SHA256 complet. Vous devez implementer une *machine à états* en VHDL synthétisable pour générer le circuit équivalent. L'entrée est encodée sur 512 bits. Les variables internes sont encodée sur 32 bits. La sortie est encodée sur 256 bits.

```
const K =

       { 0x428a2f98, 0x71374491, 0xb5c0fbcf, 0xe9b5dba5, 0x3956c25b, 0x59f111f1, 0x923f82a4, 0xab1c5ed5,
       0xd807aa98, 0x12835b01, 0x243185be, 0x550c7dc3, 0x72be5d74, 0x80deb1fe, 0x9bdc06a7, 0xc19bf174,
       0xe49b69c1, 0xefbe4786, 0x0fc19dc6, 0x240ca1cc, 0x2de92c6f, 0x4a7484aa, 0x5cb0a9dc, 0x76f988da,
       0x983e5152, 0xa831c66d, 0xb00327c8, 0xbf597fc7, 0xc6e00bf3, 0xd5a79147, 0x06ca6351, 0x14292967,
       0x27b70a85, 0x2e1b2138, 0x4d2c6dfc, 0x53380d13, 0x650a7354, 0x766a0abb, 0x81c2c92e, 0x92722c85,
       0xa2bfe8a1, 0xa81a664b, 0xc24b8b70, 0xc76c51a3, 0xd192e819, 0xd6990624, 0xf40e3585, 0x106aa070,
       0x19a4c116, 0x1e376c08, 0x2748774c, 0x34b0bcb5, 0x391c0cb3, 0x4ed8aa4a, 0x5b9cca4f, 0x682e6ff3,
       0x748f82ee, 0x78a5636f, 0x84c87814, 0x8cc70208, 0x90befffa, 0xa4506ceb, 0xbef9a3f7, 0xc67178f2 }
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
                     sortie_valid = 0

                     // 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)

                     // Boucle principal
                     pour i = 0, 63
                            si i < 16
                                   // Morceaux de 32 bits de l'entree a commencer de bits de poid fort
                                   W_i = entree_i
                            sinon
                                   W_i = σ_1^{256}(W_{i-2}) + W_{i-7} + σ_0^{256}(W_{i-15}) + W_{i-16}

                            fin si

                            T_1 = h + Σ_1^{256}(e) + Ch(e,f,g) + K_i + W_i
                            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
                     fin pour

                     // 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 = 1
                    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 arithmétique afin d'implementer l'algorithme.
2. Réalisez la description de votre module en VHDL synthétisable.
3. Ré-utilisez l'interface (ports d'entrées et sorties) présentés dans le module top fourni.
4. Validez avec les chargés de TP le diagramme d'etats de votre machine à états ainsi que le diagramme de bloc (chemin de données) de votre architecture.
5. Veuillez noter que les fonctions de compression (TP1) ainsi que les types composés et constantes sont définis dans un fichier "package" qui vous est fournis.

Points à considérer :

1. Faites attention à la logique de "reset".
2. Rappelez-vous que dans un circuit sequential les signaux sont mis à jour au cycle suivant.
3. Faites attention aux dépendences 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 et votre diagramme de bloc.

Partie B : Simulation

Un banc d'essai ainsi qu'un fichier "top" vous est fournit afin de valider votre système.
À remettre : des captures d'écran de votre simulation.

Partie C : Implémentation

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

1. Utilisez le module "top" fourni.
2. Utilisez le fichier XDC fourni.

Partie D : Analyse et Discussion

Veuillez remplir le tableau suivant et discuter les résultats.

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

Fichiers fournis

Voici les fichiers de base pour l'implementation du laboratoire 4.

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	2
Partie C : Implémentation	6
Partie D : Analyse et Discussion	2
Rapport : Présentation et qualité du français	2
Total	20