HEIG VD | REDS Institut Reconfigurable and Embedded Digital Systems

Prof. Romuald Mosqueron

Prof. Cédric Bornand

Laboratoire de systèmes logiques

semestre automne 2023 - 2024

Laboratoire Add/Sub & Checksum

Informations générales

Le rendu pour ce laboratoire se fera **par groupe de deux**, chaque groupe devra rendre son travail.

Ce laboratoire sera évalué de la façon suivante :

- Evaluation du rendu du laboratoire
- Evaluation du quiz

Ce quiz sera fait **individuellement** et évaluera votre compréhension du laboratoire. Il sera effectué au début de la prochaine séance de cours après la réception du feedback de votre rendu.

⚠ **N'oubliez pas de sauvegarder et d'archiver votre projet à chaque séance de laboratoire**

NOTE 1 : Afin de ne pas avoir de pénalité pensez à respecter les points suivants

- Toutes les entrées d'un composant doivent être connectées. (-0.1 sur la note par entrée non-connectée)
- Lors de l'ouverture de Logisim, bien préciser votre nom en tant que User
- Ne pas modifier (enlever/ajouter/renommer) les entrées/sorties déjà placées

NOTE 2 : Lors de la création de votre circuit, tenez compte des points suivants afin d'éviter des erreurs pendant la programmation de la carte FPGA :

- Nom d'un circuit $\neq$ Label d'un circuit
- Nom d'un signal (Pin) $\neq$ Label et/ou Nom d'un circuit, toutes les entrées/sorties doivent être nommées
- Les composants doivent avoir des labels différents

NOTE 3 : Nous vous rappelons que si vous utilisez les machines de laboratoire situées au niveau A, il ne faut pas considérer les données qui sont dessus comme sauvegardées. Si les machines ont un problème, nous les remettons dans leur état d'origine et toutes les données présentes sont effacées.

Outils

Pour ce laboratoire, vous devez utiliser les outils disponibles sur les machines de laboratoire (A07/A09) ou votre ordinateur personnel avec Logisim installé.

⚠ **La partie programmation d'une FPGA ne peut se faire que sur les ordinateurs présents dans les salles (A07/A09).**

Fichiers Logisim fourni

Vous devez télécharger à partir du site Cyberlearn le projet Logisim dédié à ce laboratoire.

Le projet contient certaines des entités que vous allez réaliser dans le cadre de ce laboratoire. Vous devrez compléter ces entités et en créer de nouvelles afin de réaliser les fonctions demandées.

De plus, **ne modifiez surtout pas les noms des entrées/sorties déjà placées dans ces entités et n'ajoutez pas d'entrée/sortie supplémentaires.**

Conseil sur l'organisation du laboratoire

Pour permettre un suivi de ce labo, vous devez remplir le fichier Excel mis a disposition sous Teams.

Ce laboratoire se déroule sur **3 séances**, selon le programme disponible sur Cyberlearn rubrique "Travail à Faire". vous pouvez suivre l'organisation suivante pour gérer votre travail sur ce laboratoire :

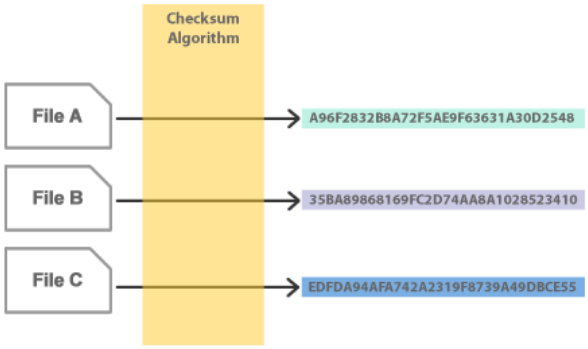
séance	**Étape à terminer**
1	add/sub 4 & 8 bits
2	checksum + mise sur carte
3	verify integrity + chronogramme –> rendu

Qu'est-ce qu'un checksum ?

L'objectif principal de ce laboratoire est la réalisation d'un checksum.

Un checksum (ou *somme de contrôle*) est une séquence de chiffres et de lettres permettant de vérifier l'intégrité d'un fichier après un transfert entre deux machines. Cela permet de vérifier si le fichier a été altéré lors de son envoi et s'il contient des erreurs.

Pour créer un checksum, on exécute un programme qui contient un algorithme de hashage. il s'agit d'un algorithme qui prend un fichier en entrée de taille quelconque (de quelques Mo à plusieurs Go) et qui exécute un calcul pour en déduire une somme de contrôle. Cette somme de contrôle est toujours de longueur fixe, quelle que soit la taille du fichier.



Une fois générée, cette séquence est transférée indépendemment du fichier. Le destinataire peu ensuite vérifier l'intégrité de son fichier (en générant à nouveau une somme de contrôle ou via un utilitaire de hashage).

Par exemple, les distributions Linux, qui font plusieurs Go fournissent une somme de contrôle pour vérifier que l'image ISO téléchargé soit fiable avant de créer un USB bootable.

Quelques exemples d'algorithme de hashage : MD5, SHA-1, SHA-256.....

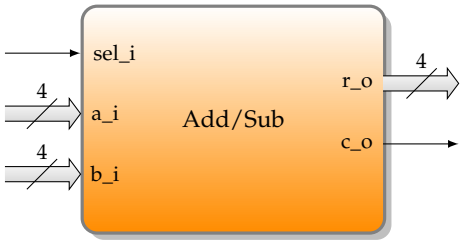
Réalisation du laboratoire

Pour réaliser votre Checksum, vous devrez d'abord créer un additionneur/soustracteur puis l'utiliser afin de créer le checksum. L'idée est de développer un système de A à Z afin que vous puissiez faire chaque étape vous-mêmes et ainsi bien comprendre les concepts vus dans la théorie du cours afin de les appliquer dans un cas pratique.

1 Add/Sub 4 bits

Travail à effectuer

Entité du bloc Add/Sub



Nom I/O	Description
sel_i	Permet de sélectionner l'addition ou la soustraction
a_i	Données A
b_i	Données B
r_o	Résultat de l'addition/soustraction
c_o	Flag indiquant un carry

Etape 1-a : Implémenter un additionneur 4 bits

Créer un composant additionneur 4 bits à l'aide d'additionneurs 1 bit dans Logisim et tester son fonctionnement.

Etape 1-b : Implémenter un soustracteur 4 bits

A l'aide d'un seul additionneur 4 bits et de portes logiques de base, créer un soustracteur 4 bits dans Logisim et tester son fonctionnement.

Etape 1-c : Implémenter un additionneur/soustracteur 4 bits

A l'aide d'un seul additionneur 4 bits et de portes logiques de base, créer un additionneur/soustracteur 4 bits dans Logisim et tester son fonctionnement.

Etape 1-d : Implémenter un additionneur 8 bits

Sur la base des additionneurs conçus précédemment, concevoir et tester un additionneur 8 bits.

Etape 1-e : Concevoir et implémenter un additionneur/soustracteur 8 bits

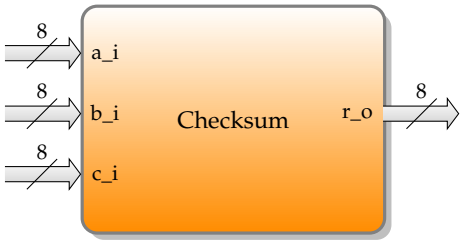
A l'aide d'un seul additionneur 8 bits et de portes logiques de base, créer un additionneur/soustracteur 8 bits dans Logisim et tester son fonctionnement.

QUESTION 1 : d'après vous, à quoi correspond le *'bit de carry'* dans un additionneur binaire, et pourquoi est-il important dans l'addition de nombres ?

2 Checksum

Travail à effectuer

Entité du bloc Checksum



Nom I/O	Description
a_i	Données A
b_i	Données B
c_i	Données C
r_o	Résultat du checksum

Etape 2-a : Concevoir et implémenter un checksum

Utiliser les additionneurs/soustracteurs 8 bits developpés pécedemments et d'autres composants si nécessaires, pour créer un système combinatoire qui calcule le checksum de type *modular sum* de trois octets de données en entrée.

Définition du *modular sum* :
Un *modular sum* se calcule en additionnant les données entre elles sans carry puis en effectuant le complément à deux de ce résultat.

> **QUESTION 2** : A partir de la définition proposée, expliquer ce qu'est un *modular sum* à l'aide d'un schéma ou d'une explication simple.

Etape 2-b : integration sur carte MAX_V

Intégrer votre checksum dans la carte « MAX_V_CONSOLE ».
Placez les constantes suivantes sur les entrées A et B :

- une constante 8 bits avec la valeur `0xA9`
- une constante 8 bits avec la valeur `0x2E`

Connectez les signaux C aux interrupteurs de la console et les sorties R aux LEDs de la console.
Lors de la programmation, dans le menu FPGA commander sélectionnez la carte « MAX_V_CONSOLE » (Choose target board).

Tester le fonctionnement sur la carte. Faites valider le fonctionnement par l'assistant.

3 Vérification de l'intégrité de données

Etape 3-a : Concevoir un circuit permettant de vérifier l'intégrité de données

Créer un circuit *VerifyIntegrity* prenant quatre entrées (trois octets de données et leur checksum) et permettant de calculer un bit de sortie valide égal à 1 lorsqu'aucune altération des données n'est détectée à la réception.

Ce circuit reçoit donc quatre octets et ne peut pas savoir lequel est l'octet de checksum.

Etape 3-b : test et validation du fonctionnement par un chronogramme

Afin de valider le fonctionnement global du système, créer un nouveau composant et le nommer `Chronogramme`.

Dans celui-ci, utiliser le composant *VerifyIntegrity* créé à l'étape 3-a et connecter à ses entrées les éléments suivants :

- une constante 8 bits avec la valeur `0xF0`
- une constante 8 bits avec la valeur `0x0F`
- une constante 8 bits avec la valeur `0xFC`
- un compteur avec 8 bits de données

L'ordre de connexion de ces éléments sur les entrées du composant *VerifyIntegrity* ne doit pas influencer le résultat.

Générer un chronogramme montrant que la sortie du composant *VerifyIntegrity* est activée à un certain moment.

Une fois votre chronogramme généré , répondez aux questions suivantes :

QUESTION 3 : Quelle est la valeur de checksum attendue pour les constantes définies ?
(utiliser les composants créés précédemment).

QUESTION 4 : Sauvegarder le chronogramme (capture d'écran). Quelles observations peut-on faire ?
Selon quelles conditions la sortie *VerifyIntegrity* passe-t-elle à 1 ?
Expliquer votre raisonnement dans le rapport au format *.pdf*.

Rendu

Pour ce laboratoire, chaque binôme devra rendre :

- votre fichier *.circ*
- un fichier au format *.pdf* contenant les réponses aux différentes questions théoriques et un chronogramme.

Vous devez déposer les rendus sur Cyberlearn jusqu'à la date indiquée dans l'espace de rendu consacré à votre classe. Ainsi, vous recevrez un feedback dans le courant de semaine suivante.

CONSEIL : Faire une petite documentation sur cette partie vous préparerait directement pour le quiz et vous fera directement un résumé pour l'examen.