

## Laboratoire de systèmes logiques semestre automne 2023 - 2024

### Laboratoire ALU

---

#### 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
- Ne pas modifier le nom des composants déjà présents

**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

---

Vous devez télécharger à partir du site Cyberlearn le projet Logisim dédié à ce laboratoire. Ce fichier est nommé ALU : Projet Logisim dans la section "laboratoire 2 : ALU" sur cyberlearn.

## Logisim fourni

---

Vous allez recevoir un projet Logisim qui contient deux entités : ALU\_MAXV et ALU\_Top. Vous ne devez pas modifier l'entité ALU\_MAXV. Vous devrez compléter l'entité ALU\_Top afin de réaliser les fonctions demandées.

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

Libre à vous de créer tous les circuits nécessaires au bon éroulement de ce laboratoire.

## Conseil sur l'organisation du laboratoire

---

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

Ce laboratoire se déroule sur **4 séances**. vous pouvez suivre l'organisation suivante pour gérer votre travail sur ce laboratoire :

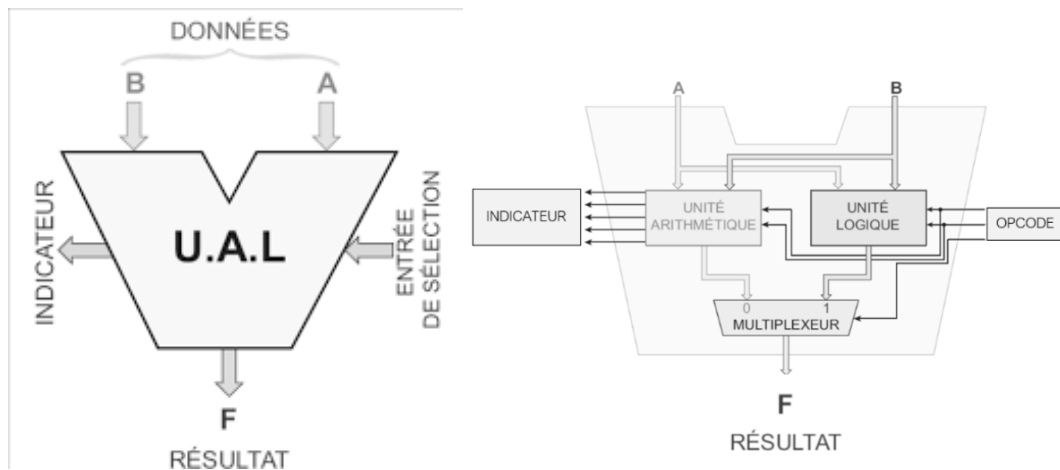
séance	Étape à terminer
1	Opcode & add/sub 4 bits & overflow
2	comparateur & unité logique
3	custom & ALU
4	constitution du bloc ALU , test et validation -> rendu

# 1 Objectifs du laboratoire

L'objectif principal de ce laboratoire est la réalisation d'une unité arithmétique et logique (ALU) 4 bits.

## 1.1 Définition de l'ALU

Il s'agit du composant électronique essentiel au fonctionnement du CPU, GPU, micro-contrôleur, puisqu'il est chargé d'effectuer des opérations logiques et arithmétiques (soustraction, addition, multiplication ...). Il s'agit en réalité du composant qui exécute les calculs.



Le composant prend en entrée 2 valeurs A et B, effectue une opération choisie et en ressort un résultat F.

L'ALU renvoie également un indicateur. Cet indicateur permet de surveiller l'état du composant et les erreurs survenues : division par zéro, dépassement de capacité (overflow), Opcode inconnu ...

## 1.2 Définition de l'Opcode

Pour connaître l'opération à effectuer, un code opération (ou Opcode) est transmis à l'ALU. Ce code permet d'identifier l'unité à joindre et l'opération à effectuer au sein de ce composant.

*Par exemple : si l'opération à effectuer est une soustraction, il faut joindre l'unité arithmétique et sélectionner l'opération de soustraction.*

Les ALU sont présents partout, mais peuvent avoir des fonctionnalités différentes selon l'utilisation qu'on souhaite leur donner *ex : calcul de virgule flottante (FPU).*

## Réalisation du laboratoire

---

Pour réaliser votre laboratoire, vous devrez d'abord créer un certain nombre de composants puis les utiliser afin de concevoir l'ALU. L'idée est de développer un système de A à Z afin que vous puissiez faire chaque étape vous-même et ainsi bien comprendre les concepts vus dans la théorie du cours afin de les appliquer dans un cas pratique.

Ce laboratoire est noté. Vous devez rendre le projet Logisim ainsi qu'un document pdf contenant les réponses aux questions demandées et répondre à un quiz. Le quiz sera orienté de façon à pouvoir évaluer votre compréhension du laboratoire.

## 2 ALU

---

Dans le cadre de ce laboratoire, vous allez implémenter une ALU 4-bits capable de réaliser les opérations arithmétiques et logiques listées ci-dessous.

Opération	Fonctionnalité	Opcode
A + B (signé)	Add/Sub	0000 $\phi\phi$
A - B (signé)		0010 $\phi\phi$
A + B (non-signé)		0001 $\phi\phi$
A - B (non-signé)		0011 $\phi\phi$
A $\geq$ B (signé)	Comparateur	011001
A < B (signé)		011010
A $\neq$ B		011011
A = B		011100
A $\geq$ B (non-signé)		011101
A < B (non-signé)		011110
A and B	Logique	10 $\phi\phi$ 00
A or B		10 $\phi\phi$ 01
A nand B (non-signé)		10 $\phi\phi$ 10
A xor B (non-signé)		10 $\phi\phi$ 11
Détecteur de puissances de 2	Custom	11 $\phi\phi\phi\phi$

Note : le symbole  $\phi$  signifie valeur indéterminée (0 ou 1).

### Architecture de l'ALU

Cette ALU est composée de quatre sous-blocs (voir figure ci-dessous), chacun réalisant une fonctionnalité particulière. Ces quatre blocs contiennent :

- Un circuit combinatoire capable d'additionner et de soustraire,
- Un circuit combinatoire exploitant le résultat du soustracteur pour déterminer si deux nombres sont égaux, ou pas,
- Un circuit combinatoire réalisant les fonctions logiques indiquées dans la liste d'opérations de l'ALU,
- Un circuit combinatoire "custom" réalisant une opération spécifique. Dans notre cas, le bloc custom indique le nombre de 1 dans l'entrée A.

## Schéma bloc de l'ALU

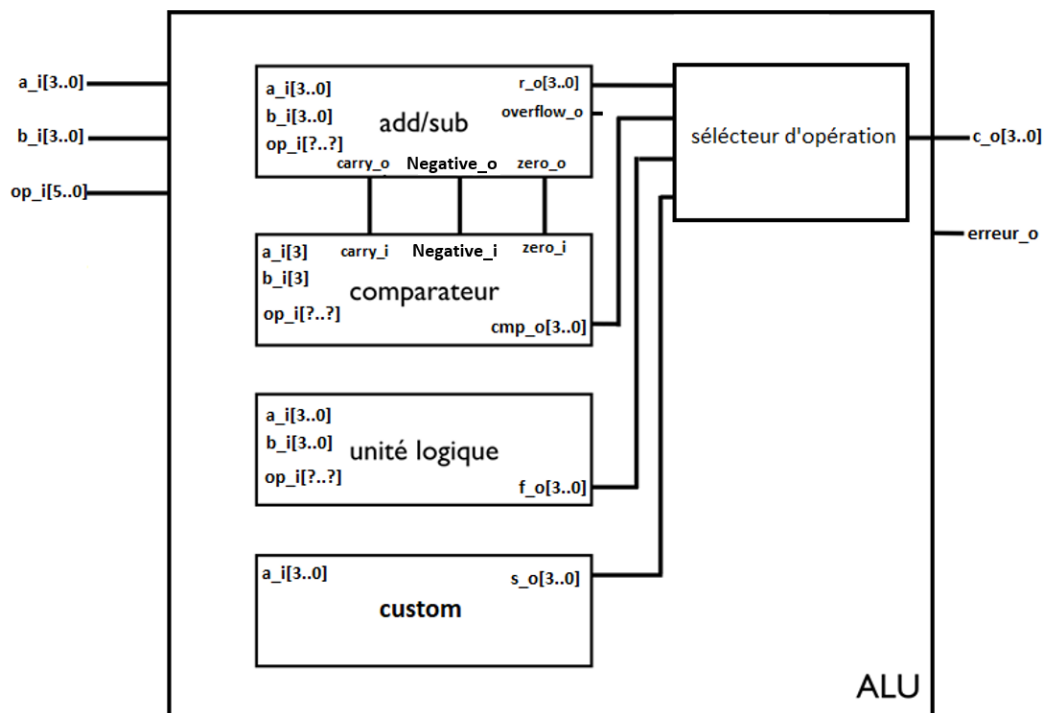


FIGURE 1 – Schéma bloc de l'ALU

Nom I/O	Description	Nombre de bits
<code>a_i</code>	Données A	4 bits
<code>b_i</code>	Données B	4 bits
<code>op_i</code>	Opcode	6 bits
<code>c_o</code>	Résultat de l'ALU	4 bits
<code>erreur_o</code>	Signal d'erreur	1 bit

## Travail à effectuer

### Etape 1 : Opcode

Identifier les bits de l'opcode (op\_i) déterminant la fonctionnalité à exécuter. Ces bits seront utilisés par un multiplexeur (voir schéma bloc de l'ALU).

Dans le cas où l'opcode est invalide, l'ALU doit retourner une erreur.

Une fois votre chronogramme généré, répondez aux questions suivantes :

**QUESTION 1 :** Identifiez et documentez les différents opcodes entraînant une erreur.

**QUESTION 2 :** Reprenez le schéma disponible en Figure 1 et donnez, pour chaque bloc, les bits d'opcode utilisés (l'idée est de répondre aux points d'interrogation dans le schéma). Vos réponses doivent être de la forme "les bits d'opcode du bloc add/sub sont les bits [X:Y]".

### Etape 2-a : Add/Sub 4 bits

Concevoir un additionneur/soustracteur 4-bits. Il doit être constitué **d'un seul** additionneur 4 bits.

L'entrée op[?:?] sélectionne l'opération que réalise le bloc Add/Sub.

Le bloc Add/Sub génère une sortie R[3:0] à 4 bits avec la somme ou la différence entre A et B, ainsi que quatre sorties à 1 bit. Ces sorties à 1 bit sont :

- Overflow : pour un dépassement de capacité.
- Carry : représente une retenue.
- Zero : si le résultat de l'opération Add/Sub vaut zéro.
- Negative : représente le signe du résultat.

Ces trois dernières sorties seront utilisées par le bloc comparateur.

**Le bit overflow est différent pour les opérations arithmétiques avec des nombres signés et non-signés. Il peut être calculé avec l'étape suivante.**

### Etape 2-b : Overflow

Concevoir le bloc (add/sub) de l'ALU en intégrant l'additionneur-soustracteur 4-bits et le calcul de dépassement de capacité (overflow). Prévoir les sorties indiquées sur le schéma de l'ALU afin de le connecter au bloc comparateur.

**NOTE :** Le bit « overflow » se calcule de manière différente lorsque l'on fait des opérations avec des nombres signés et non-signés.

- Cas non-signé : le bit carry (ou emprunt) sera considéré.
- Cas signé : Voir tableau ci-dessous

**Une seule sortie « overflow » sortira du bloc add/sub.**

Opération	A	B	Résultat indiquant un overflow
A + B	$\geq 0$	$\geq 0$	$< 0$
A + B	$< 0$	$< 0$	$\geq 0$
A - B	$\geq 0$	$< 0$	$< 0$
A - B	$< 0$	$\geq 0$	$\geq 0$

**QUESTION 3 :** Lorsqu'un overflow se produit, le signe n'est plus géré correctement et la sortie `negative_o` n'a donc pas la bonne valeur. Corrigez ce problème et expliquez votre raisonnement dans le fichier pdf.

### Etape 3 : Comparateur

L'entrée  $op[? : ?]$  du bloc comparateur sélectionne l'opération à réaliser.

Toutes ces opérations peuvent être réalisées par des fonctions combinatoires exploitant les entrées Negative, Carry et Zero.

Le bloc comparateur génère une sortie  $cmp[3 : 0]$  4-bits indiquant si le résultat de la comparaison est vrai ( $cmp[3 : 0] = '0001'$ ) ou faux ( $cmp[3 : 0] = '0000'$ ).

Concevoir le bloc comparateur en synthétisant les fonctions logiques indiquées dans le tableau ci-dessous.

Opération	Fonction logique
$A \geq B$ (signé)	$\text{not}(\text{Negative})$
$A < B$ (signé)	Negative
$A \neq B$	$\text{not}(\text{Zero})$
$A = B$	Zero
$A \geq B$ (non-signé)	Carry
$A < B$ (non-signé)	$\text{not}(\text{Carry})$

**NOTE :** Ce comparateur se base sur l'utilisation du bloc Add/Sub. Par exemple, pour déterminer si deux nombres sont égaux, il fait une soustraction et détermine une différence lorsque le résultat n'est pas zéro.

### Etape 4 : Unité Logique

Concevoir le bloc « unité logique » réalisant les opérations logiques de l'ALU. Prévoir un bus de sortie F à 4 bits comme indiqué ci-dessous.

L'entrée  $op[? : ?]$  de l'unité logique sélectionne l'opération logique bit à bit des entrées A et B à réaliser. L'unité logique génère une sortie à 4 bits  $F[3 : 0]$ .

Par exemple, si  $op = 00$ , la sortie de l'unité logique doit avoir le comportement suivant :

Sortie	Equation
$F[3]$	$A[3] \text{ and } B[3]$
$F[2]$	$A[2] \text{ and } B[2]$
$F[1]$	$A[1] \text{ and } B[1]$
$F[0]$	$A[0] \text{ and } B[0]$

### Etape 5 : Unité custom

Concevoir le bloc « unité custom » afin de réaliser la fonction suivante :

La sortie  $s[3 : 0] = '0000'$  si l'entrée A **n'est pas** une puissance de 2 et  $s[3 : 0] = '0001'$  si l'entrée A **est** une puissance de 2.

**Indication :** Vous ne devez pas utiliser de divisions ou de multiplications. Vous devez utiliser les propriétés de la représentation binaire pour trouver un moyen de savoir si un nombre est une puissance de 2 ou non.

**QUESTION 4 :** Mettez en place une table de vérité, puis justifiez le développement de votre circuit.

### Etape 6-a : ALU

Concevoir le bloc « ALU\_Top » en intégrant les quatre précédents blocs comme indiqué sur le schéma-bloc. Il doit être constitué des 3 entrées `a_i`, `b_i`, `opcode_i` et des 2 sorties `c_o`, `erreur_o`.

Si l'opcode utilisé ne correspond pas à une opération valide, le signal `erreur_o` doit être mis à '1'.

**QUESTION 5 :** En plus des opcodes invalides, certains cas d'utilisation doivent générer une erreur. Déterminez ces différents cas et documentez les avant d'implémenter les modifications nécessaires.

**QUESTION 6 :** Avec `a_i[3:0] = 0001`, `b_i[3:0] = 1001` et `op_i[6:0] = 011011` quelle est la valeur observée pour l'overflow ? Et pour la sortie `erreur_o` ? Cela correspond-t-il à ce que vous attendiez à la suite de la Question 5 ?

### Etape 6-b : Intégration / Validation

Pour l'intégration sur carte, utilisez le composant `ALU_MAXV`.

Intégrez le projet « `ALU_MAXV` » avec le but de programmer un circuit programmable et tester votre ALU. Utilisez la console d'interrupteurs pour donner les opérandes (A et B), les « dip-switch » de la carte Max V pour indiquer l'opération à réaliser et les LEDs sur la console d'interrupteurs pour afficher les sorties du système. Lors de la programmation, dans le menu « FPGA commander », sélectionner la carte « `MAX_V_CONSOLE` » (Choose target board).

Tester le fonctionnement sur la carte.

Faites valider le fonctionnement par l'assistant.



## Rendu

---

Pour ce laboratoire, vous devez rendre :

- votre fichier *.circ*
- un rapport au format *pdf* contenant les réponses aux questions posées

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.

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