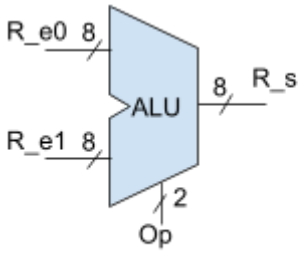
Sujet 1: Développement d'une ALU simple (Logique combinatoire, séances 1 et 2)

L'objectif de ces deux séances de TP est de développer une Unité Arithmétique et Logique simple (ALU -- Arithmetic and Logic Unit), prenant en entrée deux opérandes de 8 bits et un signal (2 bits) sélectionnant l'opération à effectuer parmi les 4 opérations suivantes :

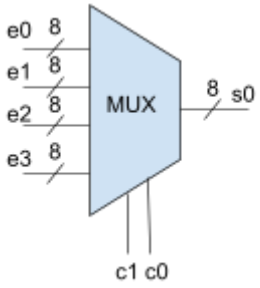
- Addition des deux opérandes.
- Soustraction des deux opérandes.
- ET bit à bit des deux opérandes.
- Fonction identité Id.



Chacun des éléments nécessaires au fonctionnement de cette ALU sera développé séparément dans le fichier **TP1.vhd**, avant d'être combinés à la fin du TP. Tout au long du TP, à chaque fois qu'un composant est développé (multiplexeur, additionneur 8 bits, soustracteur 8 bits, ET bit à bit, fonction identité), vous devrez tester ce composant en mettant en oeuvre un banc de test dans le fichier **TP1_test.vhd**. Vous devez respecter les noms imposés pour les composants : l'évaluation du TP sera réalisée par un script automatique !

I. Multiplexeur 4 voies

En utilisant une assignation conditionnelle (vue en cours-TD) **ou** en adaptant les équations (aussi vues en cours-TD) pour un multiplexeur 4 voies 1 bit, écrire l'architecture flot-de-données réalisant l'entité **MUX** du fichier **TP1.vhd**, correspondant à un multiplexeur 4 voies 8 bits.



II. Additionneur 8 bits

On souhaite désormais développer en VHDL un additionneur 8 bit à retenue propagée, comme celui vu en cours-TD. On commencera par développer un additionneur 1 bit avec retenue en VHDL flot de données, puis on combinera de tels additionneurs en VHDL structurel afin de former un additionneur 8 bits complet

II.1. Additionneur 1 bit

Ecrire en VHDL flot de données l'architecture d'un additionneur 1 bit réalisant l'entité **add1b** fournie dans le fichier **TP1.vhd**.

II.2. Additionneur 8 bits à retenue propagée

Ecrire en VHDL structurel l'architecture d'un additionneur 8 bits réalisant l'entité **add8b** fournie dans le fichier **TP1.vhd**, et en utilisant l'additionneur 1 bit réalisé à la question précédente.

III. Soustracteur 8 bits

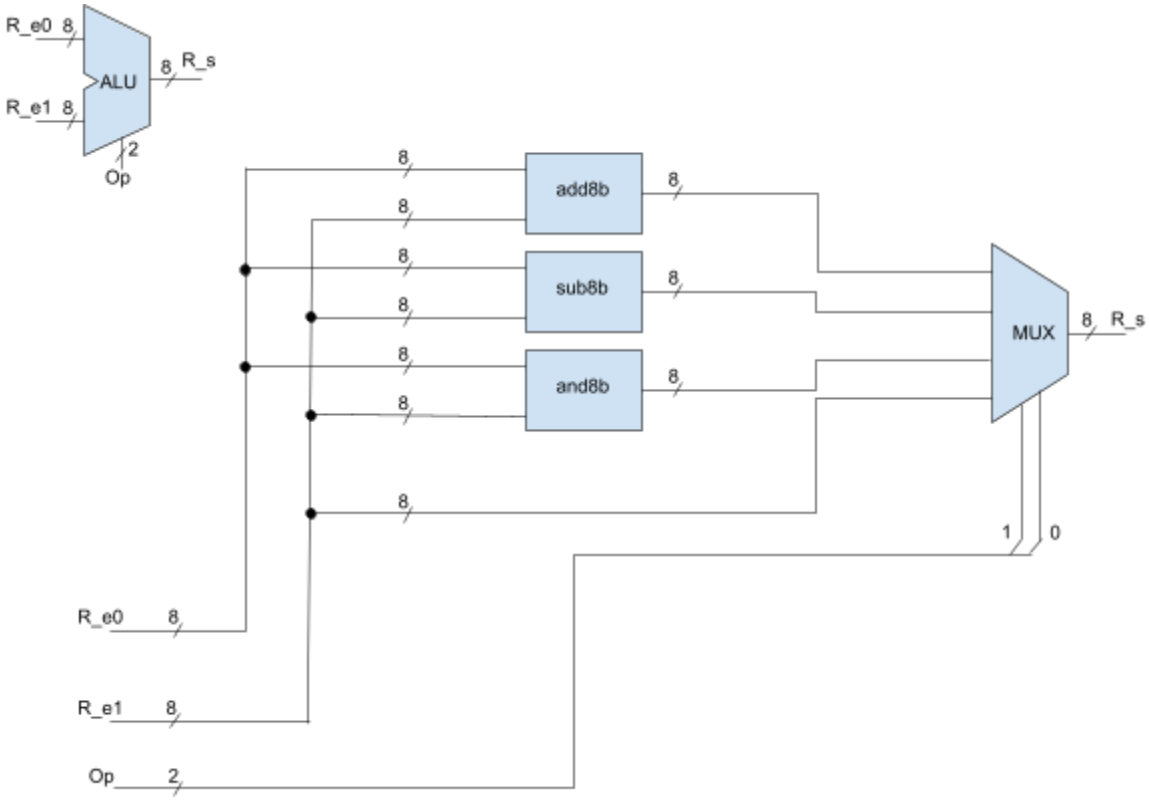
En suivant le même cheminement que pour l'additionneur, écrire un soustracteur 8 bits réalisant l'entité **sub8b** fournie dans le fichier **TP1.vhd**.

IV. AND 8 bits

Ecrire l'architecture flot de données d'un composant réalisant l'entité **and8b** fournie dans le fichier **TP1.vhd**. Cette entité effectue le ET logique bit à bit sur les valeurs de ses deux entrées.

V. Réalisation de l'ALU

On souhaite désormais assembler les modèles de composants dont on dispose (en description structurelle), afin de réaliser une ALU simple, dont l'entité **ALU** est fournie dans le fichier **TP1.vhd**.
Le schéma de cette ALU est donné ci-dessous:



La fonctionnalité souhaitée de l'ALU est la suivante:

- Si le signal Op vaut "00", le signal R_s prend pour valeur R_e0 + R_e1
- Si le signal Op vaut "01", le signal R_s prend pour valeur R_e0 - R_e1
- Si le signal Op vaut "10", le signal R_s prend pour valeur R_e0 & R_e1 (ET bit à bit)
- Si le signal Op vaut "11", le signal R_s prend pour valeur R_e1 (identité)

VI. Rendu du TP

Déposez sur moodle les fichiers TP1.vhd et TP1_test.vhd compressés dans une archive nommée <nom>.<prénom>.tar.gz:

tar -czvf nom.prenom.tar.gz TP1.vhd TP1_test.vhd