Lab 2 : VHDL

Additionneurs, Soustracteurs

- ## Introduction

L'objectif de cet exercice est d'étudier les circuits arithmétiques pour l'addition, la soustraction et la multiplication. Chaque circuit sera décrit en VHDL puis implémenté sur une des cartes FPGA DE2 d'Altera.

A la fin du document se trouve une page de rappels pour la création d'un projet sous Quartus. Normalement, vous ne devez créer qu'un seul projet QUARTUS dans un répertoire de votre espace, et celui-ci a déjà été créé lors de la première séance. On rajoutera des fichiers VHDL au fur et à mesure des séances directement ce projet.

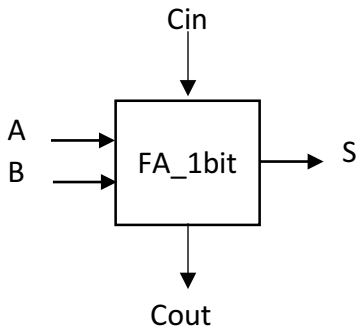
Rappel sur le lancement de quartus :
-Ouvrir le terminal, puis taper cd **/home/embed/intelFPGA_lite/18.1/quartus/bin** puis taper **./quartus** (ne pas oublier le « ./ »)

-Ouvrer ensuite votre projet Quartus, créé lors du lab 1.

- ## Exercice 1 : Additionneur

1. Additionneur 1 bit

On souhaite dans un premier temps, réaliser un additionneur 1 bit comme représenté sur la figure suivante :



Toutes les entrées et sorties sont sur 1 bit. S représente la somme des A et B et Cin (retenue entrante) tandis que Cout représente la sortie indiquant la retenue sortante éventuelle.

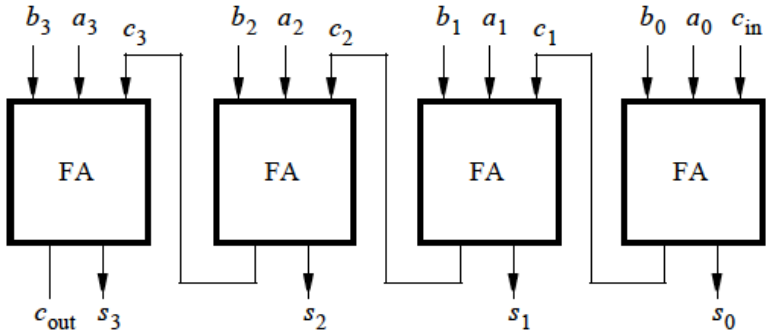
A faire :

- Sous Quartus, File -> New, VHDL File

- Sauvegarder la immédiatement via **File, Save As**, et nommer la **FA_1bit**
- Dresser sur papier la table de vérité de cet additionneur et déterminer l'équation logique optimisée pour S et Cout.
- Ecrire le code VHDL correspondant
- Décrire un fichier **Test_FA_1bit** permettant le test sur la carte du **FA_1bit**
- Faites valider par l'enseignant

2. Additionneur N bits à propagation de retenue

Une fois l'additionneur 1 bit validé, on souhaite construire un additionneur N bits à propagation de retenue. La figure suivante présente un additionneur 4 bits :



A four-bit ripple carry adder

On prendra soin de réaliser ce nouvel additionneur à partir de l'additionneur 1 bit précédent. On veillera aussi à définir un paramètre générique N permettant de concevoir un FA travaillant sur des opérandes de N bits.

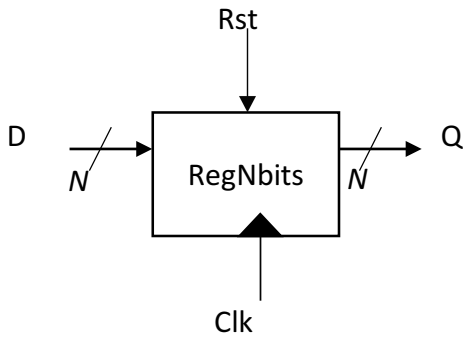
A faire :

- Sous Quartus, File, New, VHDL File
- Sauvegarder la immédiatement via **File, Save As**, et nommer la **FA_Nbits**
- Ecrire le code VHDL correspondant
- Faire valider par l'enseignant
- Créer un nouveau fichier VHDL que l'on nommera **Test_FA_Nbits**. Le définir en tant que **Top level** (clic droit sur le nom de du fichier et **set as top level entity**). Dans ce module, créer une instance du FA_Nbits avec N=4. Connecter ensuite les 4 bits de A sur les switches SW3 à SW0, les 4 bits de B sur SW7 à SW4. La retenue entrante Cin sera connectée à SW8. Les sorties S3 à S0 seront connectées aux LEDG3à0 et la retenue sortante Cout sera connectée au LEDR0.
- Compiler le module **Test_FA_Nbits**
- Faites valider par l'enseignant.

Question : Quel est l'intérêt du paramètre générique ?

Exercice 2 : Registre

On souhaite réaliser maintenant un registre de N bits, disposant d'un reset asynchrone. Un paramètre générique N sera utilisé, permettant d'instancier des registres de tailles différentes.



Registre générique N bits

A faire :

a. Sous Quartus, File, New, VHDL File
b. Créer le module **Reg1bit** permettant la génération d'un registre 1 bit.
c. Tester le sur la carte après la création d'un module **Reg1bit_test** qui instanciera notre registre 1 bit.
d. Faire valider par l'enseignant.
e. Créer un nouveau fichier VHDL que l'on nommera **RegNbits**. Ce nouveau module instanciera N registres 1 bit.
f. Faire valider par l'enseignant.

Exercice 3 : Circuit Soustracteur N bits

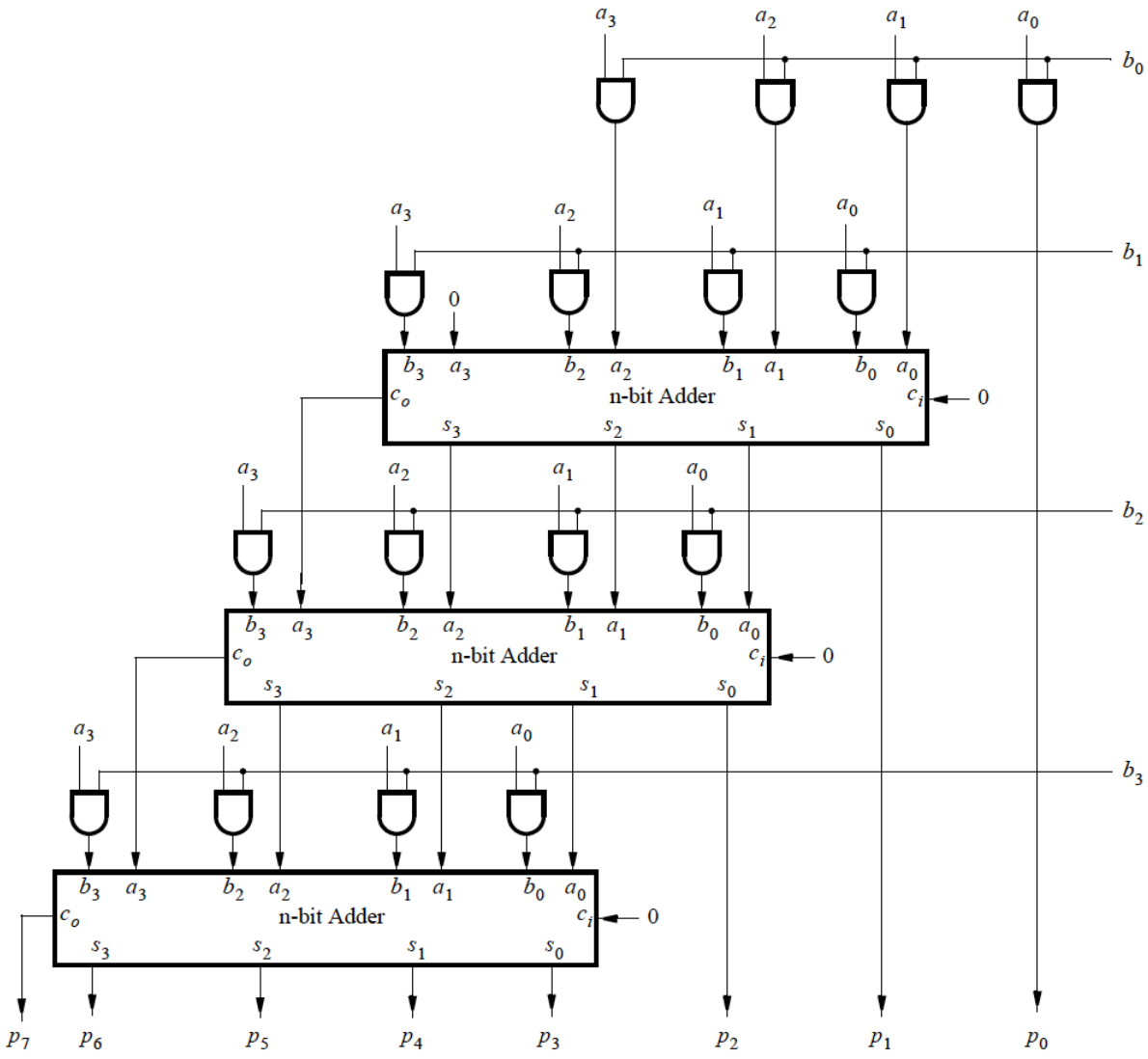
Dans ce nouvel exercice, on souhaite étendre le circuit de l'additionneur Nbits à propagation de retenue afin de pouvoir réaliser en plus de l'addition, des soustractions. Un signal supplémentaire, appelé add sub et relié au switch SW17, permettra de sélectionner l'opération désirée. Lorsque add sub est à 1, le circuit fournira en sortie la soustraction A-B. Autrement on calculera l'addition.

A faire :

a. Sous Quartus, File, New, VHDL File
b. Sauvegarder la immédiatement via **File, Save As**, et nommer la **AddSubNbits**
c. Ecrire le code VHDL correspondant. Vous veillerez à ré-utiliser l'additionneur.
d. Créer ensuite un nouveau fichier VHDL que l'on nommera **Test_AddSubNbits**. Ce nouveau module permettra d'instancier le module **AddSubNbits** et de le connecter aux broches du FPGA.

- # Exercice 4 : Multiplieur

Dans cet exercice, nous allons réaliser un circuit multiplieur en utilisant des composants Full Adder (FA) n-bits. Le multiplieur complet peut alors être représenté à la manière de la Figure suivante :



Array multiplier conçu à partir de full adder n-bits

Quelques questions avant de commencer :

-Sur combien de bits est codé le résultat d'une addition de deux opérandes de N bits ?

-Combien utilise-t-on d'additions partielles pour produire le résultat d'une multiplication d'opérandes codées sur n bits ?

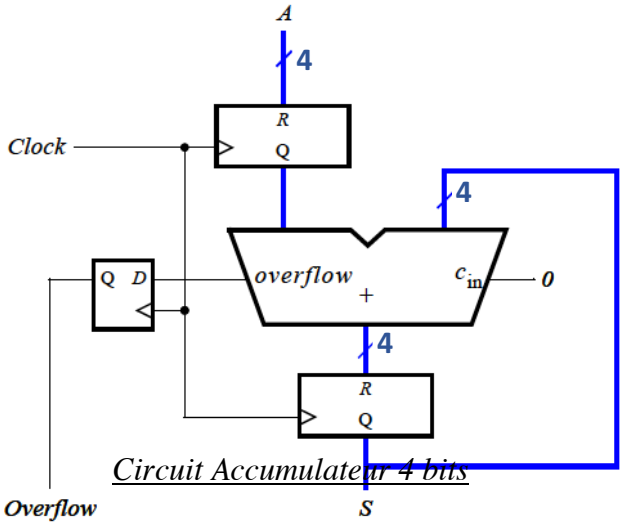
-Sur combien de bits doit être codé le résultat d'une multiplication d'opérandes codées sur n bits ?

A faire :

- Sous Quartus, File -> New, VHDL File
- Sauvegarder le nouveau fichier immédiatement via **File, Save As**, et nommer la **MultArray_4bits**
- Ecrire le code VHDL correspondant à *l'array multiplier* en réutilisant des blocs de la séance précédente, i.e. l'additionneur n-bits.
- Faites une simulation fonctionnelle pour valider votre multiplieur
- Créer un nouveau fichier VHDL appelé **Test_MultArray_4bits** pour encapsuler le composant. On connectera aux entrées A, les switches SW7 à SW4 et à l'entrée B les Switches SW3 à SW0. Les valeurs A et B seront envoyées sur les afficheurs 7 segments HEX0 et HEX1. Le résultat P=AxB sera envoyé sur les afficheurs HEX2 et HEX3.
- Compiler l'entité **Test_MultArray_4bits** et tester sur la carte
- Noter le nombre de ressources logiques utilisées (Logic Elements) de votre circuit. Observer avec Chip Planner (onglet Tools) l'utilisation du FPGA.
- Noter la fréquence maximale du circuit.
- Faites valider par l'enseignant

Exercice 5 : Circuit Accumulateur 4 bits

On dispose maintenant de registre et d'un additionneur génériques. On souhaite maintenant réaliser un accumulateur 4-bits comme présenté sur la figure suivante :



Circuit Accumulateur 4 bits

A faire :

a. Sous Quartus, File, New, VHDL File
b. Sauvegarder la immédiatement via **File, Save As**, et nommer la **Acc4bits**
c. Ecrire le code VHDL structurel correspondant
d. Créer ensuite un nouveau fichier VHDL que l'on nommera **Test_Acc4bits**. Ce nouveau module permettra d'instancier le module **Acc4bits** et de le connecter aux broches du FPGA. On veillera à respecter les connexions suivantes :

Entrées de A sur 4 bits connectées aux SW3 à SW0. On utilisera le bouton poussoir KEY(0) comme reset asynchrone actif à l'état bas, et le bouton KEY(1) comme horloge manuelle pour les registres (porte inverseuse nécessaire ?). Le résultat de l'addition sera affiché sur les LEDR3à0 et la retenue sortante sur LEDR(4). On affichera aussi l'opérande A en sortie de son registre ainsi que S en sortie de son registre. On utilisera les 2 afficheurs 7 segments HEX0 et HEX1.

e. Définir **Test_Acc4bits** en tant que **top level entity.**
f. Compiler et tester sur la plateforme
g. Faire valider par l'enseignant.