# Laboratoire 3

## Additionneurs et multiplieurs

L'objectif de ce sujet est d'examnier et de comparer plusieurs manière de concevoir des circuits arithmétiques et leur écriture VHDL et d'en observer les conséquences sur le circuit réalisé. Nous appliquerons ces principes à la conception d'un multiplieur qui sera plus tard réutilisé durant le projet.

### Partie I

Nous allons nous intéresser dans ce sujet à la réalisation de l'opération de multiplication en matériel en ré-utilisant le composant additionneur du sujet précédent.
La figure $1a$ fournit un exemple de calcul manuel de la multiplication $P = A \times B$, où $A = 11$ et $B = 12$.

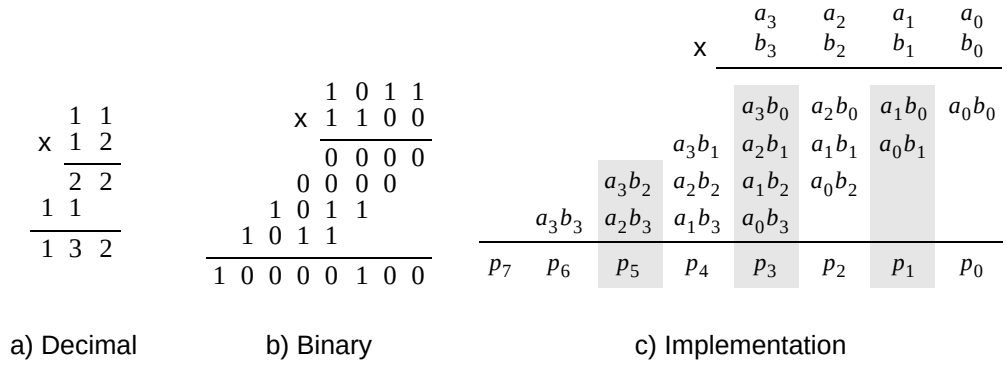


Figure 1: Exemple de multiplication de deux nombres binaires.

On calcule l'opération $P = A \times B$ comme une addition des produits partielles. A gauche (partie a), en décimal, le premier produit partiel est égal à $A$ fois le premier chiffre de $B$. Le second correspond à $A$ fois le chiffre des dizaines de $B$, décalé d'une position à gauche. On additione ensuite ces deux produits partiels pour obtenir le résultat $P = 132$.

Dans la partie $b$ de la figure, on détaille la même opération en binaire en utilisant des opérandes codées sur 4 bits. Le calcul de $P = A \times B$ se réalise alors en préparant les produits partiels binaires en multipliant succéssivement $A$ avec chaque bit de $B$ en partant du bit de poids faible (LSB, Less Significant Bit). Puisque chaque bit de $B$ vaut soit 1 soit 0, les produits partiels correspondent soit à la valeur de $A$ décalée à chaque bit d'une position, soit à 0000. La figure $1c$ montre comment utiliser l'opération logique AND pour produire les produits partiels à partir de $A$ et de chaque bit de $B$.

Un circuits numérique implémentant cette opération $P = A \times B$ sur 4 bits est illustrée dans la figure 2. A cause de sa structure régulière ce type de multiplieur est appelé *array multiplier*. On retrouve cette structure régulière assez facilement dans le layout des circuits VLSI, au même titre que les mémoires caches ou des bancs de registres. Les colonnes grisées correspondent aux sommes grisées dans la figure $1c$. Dans chaque ligne du multiplieur des portes AND sont utilisées pour produire les sommes partielles, et des additionneurs complets (*full adder*) sont utilisés pour produire chaque bit de la somme finale.

Pour implémenter le circuit *array multiplier* :

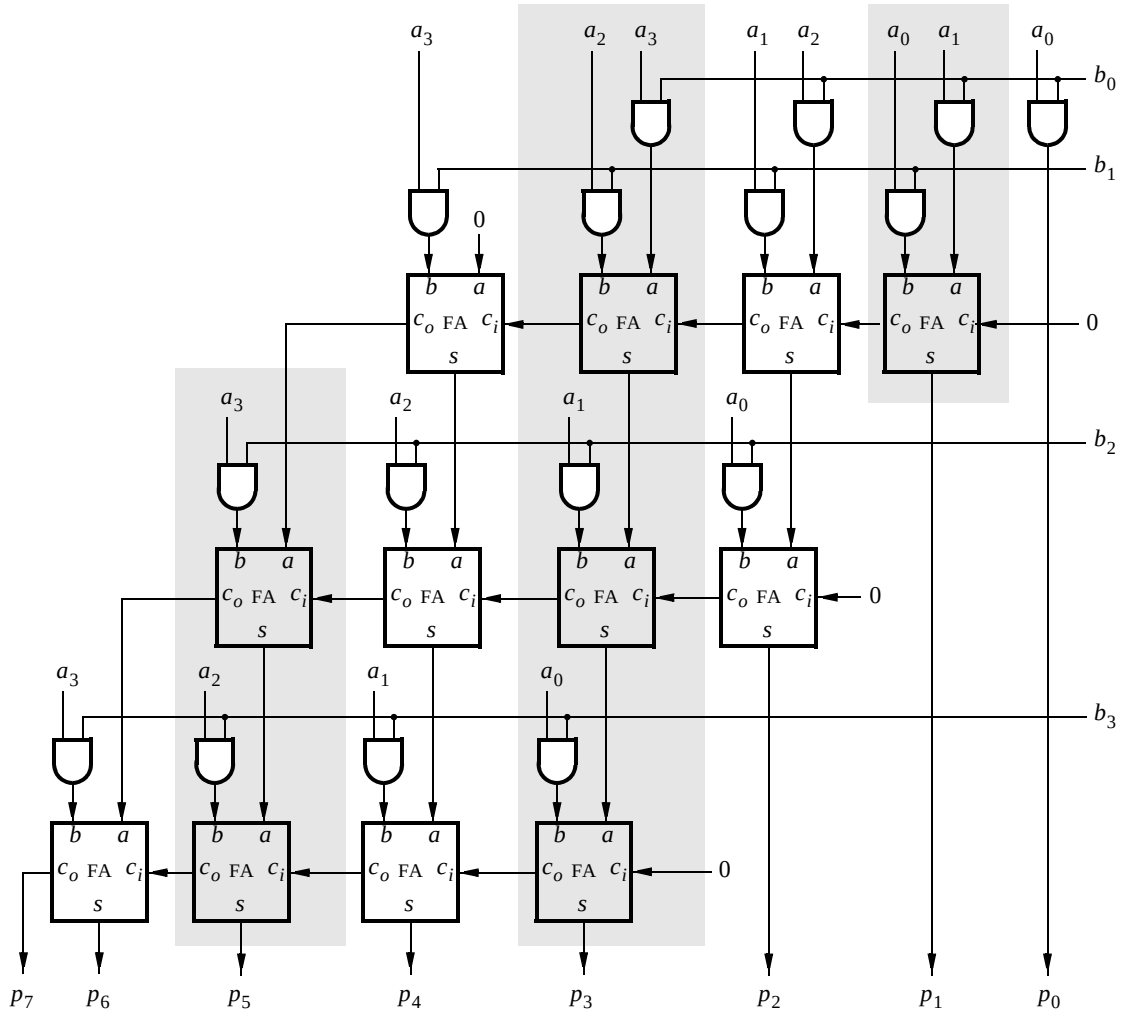


Figure 2: Un circuit de type *array multiplier*.

1. Créer un nouveau projet Quartus II et paramétrer votre carte DE2.
2. Sur combien de bits doit être codé le résultat d'une addition d'opérandes codées sur 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 ?
3. Décrivez votre code VHDL en réutilisant les blocs déjà conçus dans les séances précédentes.
4. Utilisez la simulation fonctionnelle pour vérifier votre circuit.
5. Encapsuler votre module dans une entité de haut niveau (top), qui utilisera les switches $SW_{11-8}$ pour représenter l'entrée $A$ et les switches $SW_{3-0}$ pour $B$. Les valeurs hexadecimales de $A$ et $B$ seront affichées sur les 7-segments *HEX6* et *HEX4*, respectivement. Les résultat $P = A \times B$ sera affiché sur *HEX1* et *HEX0*.
6. Tester le fonctionnement sur la carte.
7. Quelle est l'utilisation en ressources (Logic Elements) de votre design ? Observer avec Chip Planner (onglet Tools) l'utilisation de la matrice FPGA.
8. Quelle est la fréquence max du circuit ?

**Part II**

Dans la partie I, un array multiplier a été implémenté en utilisant des composants *Full Adder* (FA) 1-bit. En considérant le problème à plus haut-niveau une ligne de FA peut être remplacée par un additionneur $n$-bits et le multiplieur complet peut alors être représenté à la manière de la Figure 3.

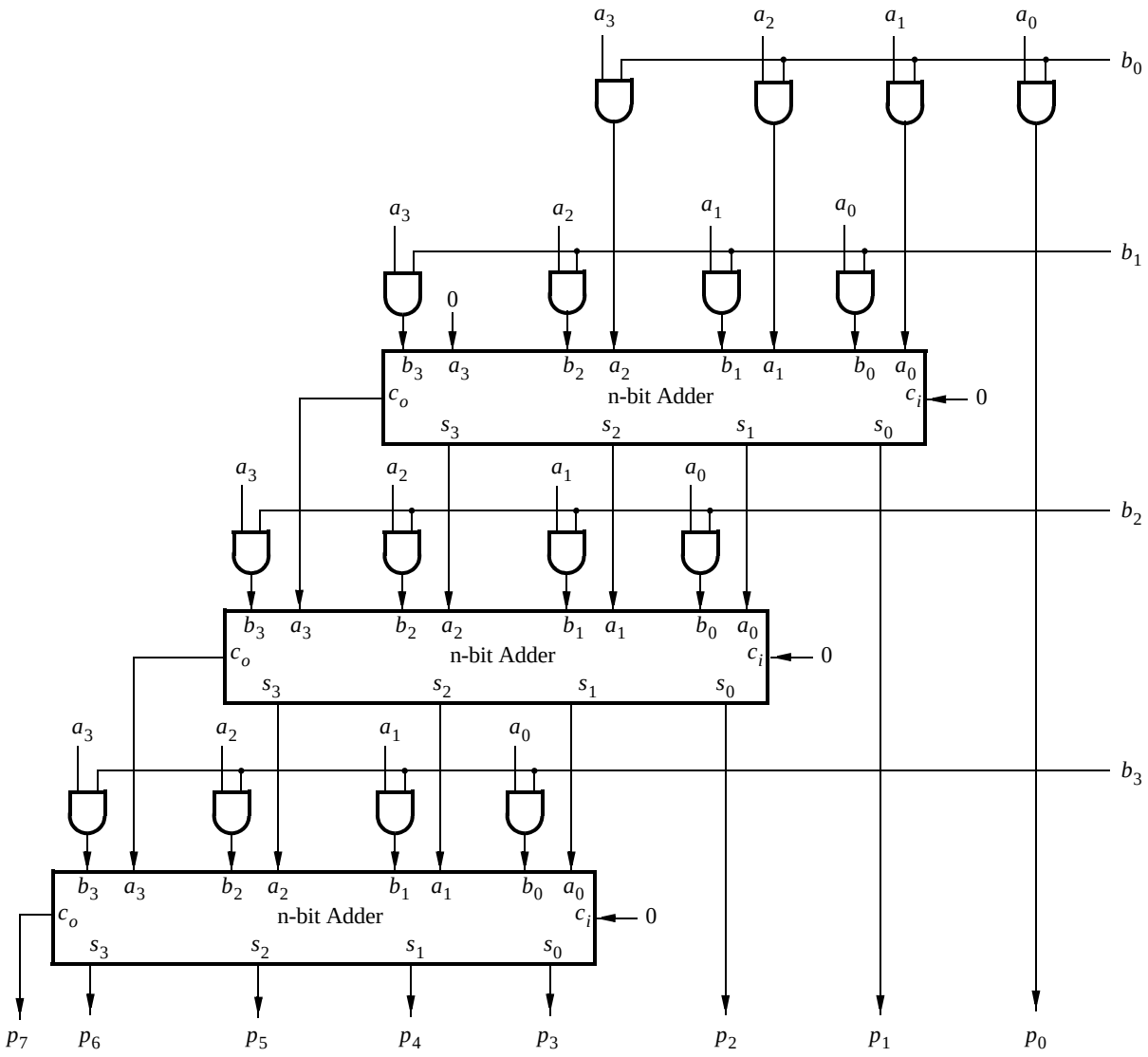


Figure 3: Un *array multiplier* implémenté en utilisant des additionneurs $n$-bit.

Chaque additionneur fait ici l'addition de la valeur $A$ décalée succéssivement et du produit partiel de la ligne précédente. Faire l'abstraction du détail des additionneurs permet de raisonner à plus haut niveau et donc de construire des additionneurs de plus grande taille. Utilisez cette approche pour implémenter un multiplieur 8x8 avec entrées et sorties mis en registres comme représenté dans la Figure 4.

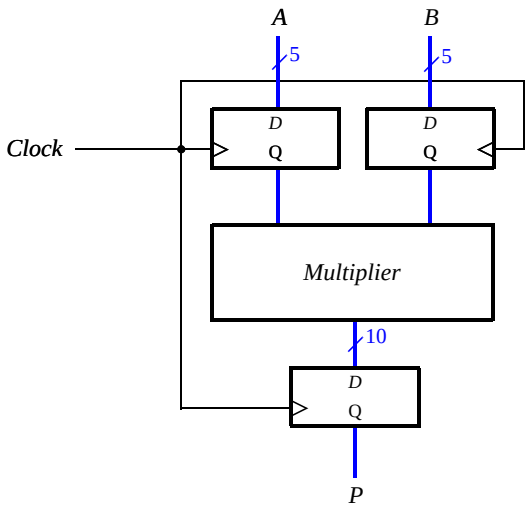


Figure 4: Le multiplieur 8x8 et ses registres.

1. Créer un nouveau projet Quartus II.
2. Ecrire le code de ce multiplieur et compiler le.
3. Utilisez la simulation fonctionnelle pour vérifier le comportement.
4. Encapsuler votre circuit pour utiliser les switches $SW_{15-8}$ pour paramétrer la valeur de l'opérande $A$ et les switches $SW_{7-0}$ pour $B$. Les valeurs hexadécimales de $A$ et $B$ seront affichées sur *HEX7-6* et *HEX5-4*, respectivement. Le résultat $P = A \times B$ sera affiché sur *HEX3-0*.
5. Tester la functionalité du design sur la carte.
6. Quelle est l'utilisation en ressources (Logic Elements) de votre design ? Observer avec Chip Planner (onglet Tools) l'utilisation de la matrice FPGA.
7. Quelle est la fréquence max du circuit ?

**Part III**

La partie II utilisait une cascade d'additionneurs pour réaliser la multiplication. Les valeurs de $A$ décalées étaient additionnées une par une.
Une autre manière d'implémenter ce circuit est de réaliser l'addition par un arbre d'additionneurs.

C'est une méthode alternative pour additionner plusieurs nombres de manière parallèle, et donc mieux adaptée à l'implémentation matérielle. Cette idée est illustrée dans la Figure 5. On peut y observer que les nombres $A$, $B$, $C$, $D$, $E$, $F$, $G$, et $H$ sont additionner en parallèle. L'addition $A + B$ se fait en même temps que $C + D$, $E + F$ et que $G + H$. Le résultat de ces opérations est lui aussi additionné en parallèle jusqu'à obtenir la somme finale $P$.

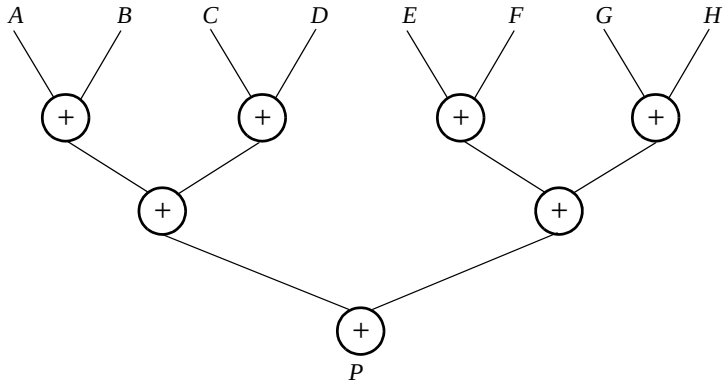


Figure 5: Un exemple d'additionneur en arbre.

Nous allons donc dans cette parte réaliser un multiplieur 8x8 basé sur un additionneur en arbre. Ce circuit est illustré dans la Figure 3. Les entrées $A$ et $B$, ainsi que la sortie $P$ doivent être mises en registres, comme dans la Partie IV.

Noter et comparer l'utilisation des ressources du FPGA ainsi que la fréquence max de fonctionnement.