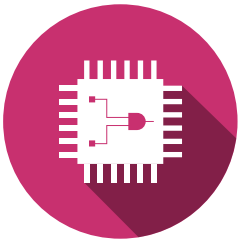


Additionneur à grande vitesse (en pipeline)

Table des matières

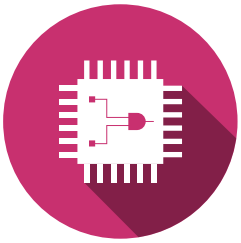
1 Introduction

Ce laboratoire a pour but de transformer un additionneur simple en une version "rapide", basée sur le principe du "pipeline".

2 Lancement

Le circuit se trouve dans la librairie **PiepelinedOperators**. Le banc de test dans la librairie **PiepelinedOperators_test**.

*Rappel : le programme de modélisation doit être lancé à travers le fichier **pipelinedOperators.bat**.*



3 Objectifs

3.1 Additionneur combinatoire

Un additionneur combinatoire de n bits se compose de n additionneurs 1 bit, chacun composés de portes avec jusqu'à 3 niveaux de retard (le chemin le plus long passant successivement par les portes XOR, AND, puis OR pour déterminer Cout). La validité du calcul ne peut donc pas être garantie avant un délai minimal de $3 * t_{delay}$ pour le premier additionneur, puis $2 * t_{delay}$ par additionneur supplémentaire (la première XOR ayant une valeur déjà valide une fois a et b chargés au début du calcul, il ne reste qu'à recalculer Cout).

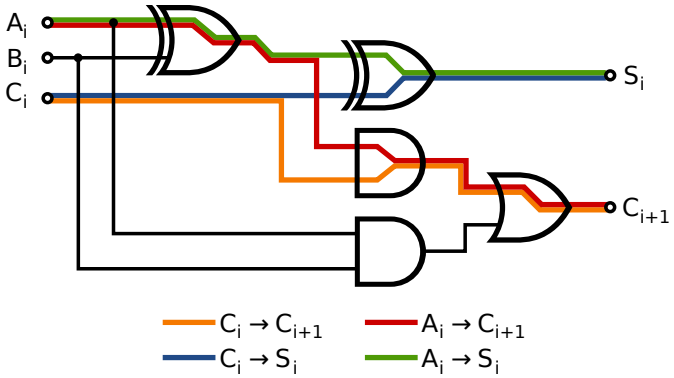


FIGURE 1 – Additionneur combinatoire complet

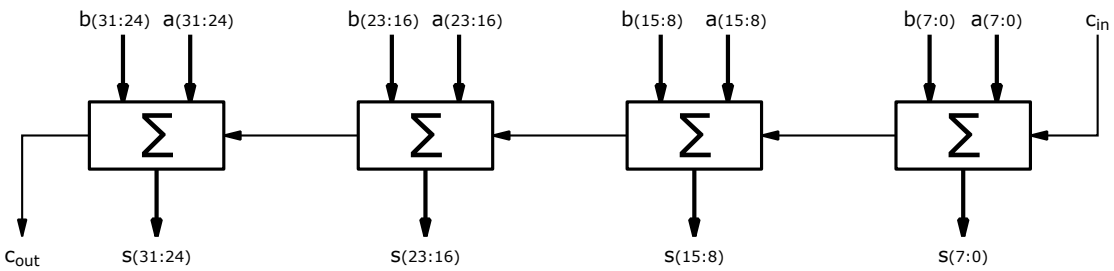
Au final, une simple addition 16 bits revient à 33 t_{delay}. Selon la technologie, un délai de 10 ns peut s'appliquer, impliquant un temps d'attente minimal de 330 ns, et ainsi une horloge de fréquence maximale de 3 MHz. Il devient donc nécessaire de découper cet additionneur en plus petites pièces pour pouvoir accélérer le système dans sa globalité.

3.1.1 Todo

Ouvrir le banc de test de l'additionneur combinatoire **parallelAdder_tb**, écrire l'architecture VHDL de l'additionneur **parallelAdder** et vérifier le bon fonctionnement du tout.

3.2 Additionneur itératif

L'additionneur rapide utilisera un ensemble d'additionneurs combinatoires. Cet additionneur est réalisé dans un premier temps dans le but de comprendre la manière de décrire un circuit en VHDL en instanciant des composants à l'aide de boucles.



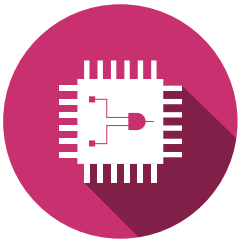


FIGURE 2 – Additionneur itératif

*L'architecture **noPipe** mise à disposition donne un exemple de description VHDL d'un système itératif par le placement de composants à l'aide d'une boucle for ... generate.*

3.2.1 Todo

- Dans la librairie **pipelinedOperators**, définir l'architecture **noPipe** comme architecture par défaut du bloc **pipelineAdder**.
- Ouvrir le banc de test de l'additionneur combinatoire **pipelineAdder_tb** et vérifier le bon fonctionnement de l'additionneur.

3.3 Additionneur rapide

L'additionneur rapide se fait en insérant des bascules entre les additionneurs combinatoires :

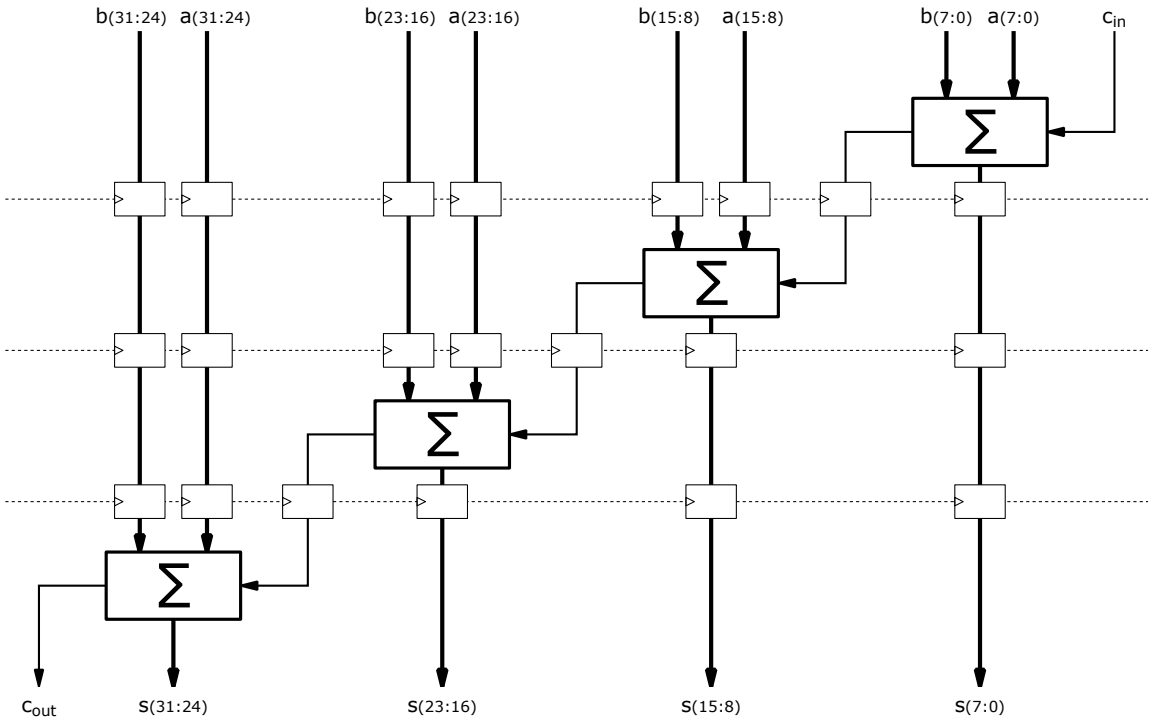
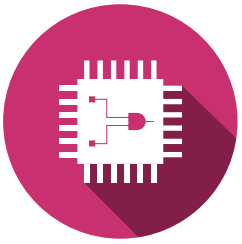


FIGURE 3 – Additionneur rapide

Bien sûr, d'un point de vue "clock", un simple calcul prend maintenant n coups d'horloge pour fournir un résultat. Mais pour rappel, il est possible d'avoir une horloge bien plus rapide et, si le pipeline est alimenté par plusieurs calculs successifs, il est possible d'avoir un résultat à chaque coup de clock.

3.3.1 Todo

- Dans la librairie **pipelinedOperators**, définir l'architecture **studentVersion** comme architecture par défaut du bloc **pipelineAdder**.
- Ecrire l'architecture VHDL de l'additionneur **pipelineAdder** et vérifier le bon fonctionnement de l'additionneur.



3.4 Compteur rapide

L'additioneur rapide est donc capable de fournir une nouvelle somme à chaque période d'horloge. Il lui faut cependant un certain nombre de périodes d'horloge avant de fournir le résultat de la somme.

3.4.1 Todo

Ecrire l'architecture VHDL du compteur **pipelineCounter** et vérifier son bon fonctionnement.

3.5 Comparaison

Le compteur avec pipeline devrait en principe pouvoir fonctionner à plus haute vitesse qu'un compteur avec un additionneur combinatoire, à propagation de report.

3.5.1 Todo

Effectuer la synthèse du compteur avec les deux variantes de l'additionneur et comparer les vitesses de fonctionnement. *Avec la sélection de l'additionneur combinatoire (architecture **noPipe**), le fonctionnement ne serait pas correct, mais le but ici est uniquement de comparer les vitesses de fonctionnement.*