Introduction des Architectures Système

Compte Rendu TP – Titouan GAUTIER

Sommaire :

Le circuit d'addition et soustraction

Notre additionneur a pour but d'être un additionneur 16 bits, pour le réaliser nous y avons été étapes par étapes. J'ai d'abord réalisé **un additionneur 1bit.** C'est la première brique de notre additionneur 16 bits. Il possède 3 entrés : A et B les deux opérandes et Cin la retenue entrante, et 3 sorties : S qui est le résultat de l'addition, P qui est vrai si les valeurs des entrées sont telles qu'une retenue sera propagée en sortie et G qui est vraie si les valeurs des entrées sont telles qu'une retenue sortante est nécessairement engendrée.

L'équation de calcul de S est : ~A ~B Cin + ~A B ~Cin + A ~B ~Cin + A B Cin

L'équation de P est : A + B

L'équation de G est : A . B

Après avoir réalisé notre additionneur 1 bit, nous devons créer **un additionneur 4bit**. Pour ce faire, nous devons assembler 4 additionneurs 1 bit et y ajouter une unité de calcul anticipé de la retenue.

L'unité de calcul anticipé de la retenue a pour but de gagner en performance en calculant la retenue en parallèle de l'additionneur. Pour la réaliser, il faut prendre la retenue entrante Cin et l'additionner à la sortie P et G de notre première additionneur dites P1 et G1, ce qui donne :

C1 = Cin

C2 = G1 + Cin . P1

C3 = G2 + C2 . P2

Ainsi de suite, Ce qui donne un circuit possédant 9 entrées : 4 entrées P, 1 pour chaque additionneur 1 bit, et 4 entrées G, 1 pour chaque additionneur 1 bit. Il y a aussi l'entée Cin, la

retenue entrante. Il possède 5 sorties, 4 retenue calculée par nos additionneur 1 bit et la retenue sortante Cout.

Ce circuit fonctionne mais il a un problème, il n'est pas performant car il possède trop d'opération en série ce qui met plus de temps à s'exécuter que des opérations en parallèles. J'ai donc récupéré les équations de calcul du circuit sur Logisim à l'aide de l'outil analyse :

C1 = Cin

C2 = G1 + Cin.P1

C3 = G2 + G1.P2 + Cin.P1.P2

C4 = G3 + G2.P3 + G1.P2.P3 + Cin.P1.P2.P3

Cout = G4 + G3.P4 + G2.P3.P4 + G1.P2.P3.P4 + Cin.P1.P2.P3.P4

A l'aide de ces équations de calculs j'ai pu réaliser **l'unité de calcul anticipé de la retenue plus performante**. Ce circuit possède les mêmes entrées et sorti que le précédent. J'en ai profité pour rajouter les sortie Super-P et Super-G qui nous servirons plus tard.

Maintenant qu'on a fait tout ça, il ne nous reste plus qu'à assembler 4 additionneurs 1 bit avec l'unité de calcul anticipé de la retenue. **L'additionneur 4 bit** possède une entrée A et B de largeur 4 bits, ce sont les opérandes et l'entrée Cin, la retenue entrante. Il possède une sortie S de largeur 4 bits qui est la somme de nos opérandes et la sortie Cout, retenue sortante. Les 4 additionneur 1 bit qui le compose donne 4 signal P et G qui rentre dans l'unité de calcul anticipé de la retenue qui permettent de calculer la retenue.

Il y a aussi deux sorti qui sont Super-P et Super-G. Super-P est un super-signal de propagation, qui indique si une retenue sortante est engendrée par l'additionneur 4 bits et Super-G est un super-signal de génération, qui indique si une retenue sortante est engendrée par l'additionneur de 4 bits. Ces deux sorties vont nous permettre de calculer plus facilement la retenue dans notre additionneur 16 bit et nous permettre de gagner en performance. Les équations permettant de les calculer sont celle-ci :

Super-P : P1 . P2 . P3 . P4

Super-G : G4 + G3.P4 + G2.P3.P4 + G1.P2.P3.P4

Pour l'additionneur 16 bits nous allons reprendre la même démarche, assembler 4 additionneurs 16 bits.

L'additionneur 16 bits se compose de deux entrées A et B de largeur 16 bits qui se propage dans 4 fils de largeur 4 bits. Chaque fils A et B rentre dans un additionneur 4 bit et donne la sortie Super-P et Super-G qui rentre dans une unité de calcul anticipé de la retenue. Il y a aussi l'entrée Cin. Il possède une sortie S de 16 bits qui est le résultat de l'addition 16bits. Elle se compose de 4 fils de largeur 4 bits qui sont les sortie S de chaque additionneur. Il y a aussi la sortie Cout.

Maintenant, il ne nous reste plus qu'à adapter notre additionneur pour qu'il puisse faire des soustractions. En base 2, faire une soustraction reviens à additionner à son opposé, c'est le principe du complément à deux. J'ai donc simplement mis un inverseur en sortie de l'additionneur et mis un multiplexeur.

Ce circuit comporte une entrée A de 16bits qui rentre directement dans un additionneur 16 bits, une entrée B de 16 bits qui inversée ou non, rentre dans un multiplexeur. Le multiplexeur choisit l'entré B inversé ou non selon le bit de sélection, pour 0, B n'est pas inversé et pour 1, B est inversé. Il y a aussi l'entrée Cin qui est la même que le bit de selection. Au niveau de sortie il y a la sortie S de 16 bits qui est le résultat, et les sorties Super-P, Super-G et Cout, la sortie sortante.

L'unité Arithmétique et Logique

Une unité arithmétique logique est un ensemble de circuits électroniques connectés logiquement de façon à réaliser, sous l'action de commandes élémentaires, les opérations arithmétiques ou logiques pour lesquelles cet ensemble a été conçu. Notre circuit d'addition et soustraction était la première étape de notre UAL, nous allons maintenant rajouter certaines opérations arithmétiques comme : l'inverseur logique(~A), le ET logiques ($A \cap B$), le OU (A U B) et le OU exclusif logiques ($A \oplus B$), ainsi que le décalage à gauche de A de 1 bit et le décalage à droite arithmétique de A de 1 bit.

Toutes les sorties de nos opérations arithmétiques rentrent dans un multiplexeur de largeur 16 bit avec 3 bits de sélection. Les 3 bits de sélection nous permette de faire entrer

nos 8 opérations car il y a 8 possibilité entre $(000)_2$ et $(111)_2$. Chaque combinaison de bit de sélection corresponde à une opération arithmétique :

- 000 → A + B
- 001 → A – B
- 010 → ~A
- 011 → A.B
- 100 → A U B
- 101 → A ⊕ B
- 110 → décalage à gauche de A de 1 bit
- 111 → décalage à droite arithmétique de A de 1 bit

Malheureusement je n'ai pas pu faire la suite de l'UAL car je fais parti du groupe qui à eu deux séance en moins.

Bilan

Durant notre TP nous avons appris à réaliser une UAL et à quoi cela servait. Nous avons démarré d'un additionneur 1 bit pour petit à petit atteindre l'additionneur 16 bits et l'UAL en utilisant les connaissances acquises dans le cours. Nous avons notamment appris comment un processeur fonctionnait et comment le rendre plus performant selon le circuit réalisé. Lors de la conception du circuit j'ai rencontré quelque problème comme lors de la conception de l'unité de calcul anticipé de la retenue, j'avais réalisé un circuit simple mais qui avait beaucoup de calcul en série ce qui faisait passer le calcul dans beaucoup de porte logique ce qui le rendait donc pas très performant. J'ai donc dû le modifier pour en réaliser un autre plus compliqué mais avec des calculs en parallèle ce qui le rendait bien plus performant. L'unité de calcul anticipé de la retenue sert elle-même à rendre un circuit plus performant car elle permet de calculer la retenue en parallèle de l'addition.

Lors de ce TP on a pu remarquer qu'un processeur ne fonctionne pas forcément mieux si le circuit est simple, il fonctionne mieux s'il passe dans le moins de porte logiques possible

même si le circuit est plus compliqué. De plus les processeurs ne fonctionnent qu'en porte logique NAND, on va donc devoir adapter toutes les autres portes logiques pour les réaliser avec des NAND.

Le développement d'un circuit sur logicielle n'est qu'une représentation graphique du circuit, il est la juste pour pouvoir tester de nouveau circuit, expérimenter, et permet de tester si le circuit marche. Grace à ces logicielles on peut réaliser des circuit complexe plus facilement et il permet de trouver les équations de calculs plus simplement. Il permet aussi de faire des économie de porte logiques et donc de transistors à la conception matériel, sans ces logiciels la conception de circuit matérielle serait beaucoup plus complexe car il faudrait réaliser chaque circuit en matériel pour pouvoir le tester et si il ne marche pas en fabriquer un autre. A l'aide de ces logicielles, on réalise le produit avec des transistors CMOS et on reproduit le circuit réalisé sur le logicielle. Cela nous permet de réaliser des circuit plus performant et comptant moins de porte logique et aussi de ne pas se tromper sur notre circuit.

Lors de la conception d'un circuit nous rencontrons plusieurs problèmes comme par exemple la recherche de performance. En effet, lors de la conception de notre UAL nous avons pensé certains circuits pour les rendre plus rapides notamment l'unité de calcul anticipé de la retenue qui permet de calculer la retenue en parallèle du calcul de l'addition. Pour la rendre plus performante nous avons dû réduire les opérations en série pour privilégier les opérations en parallèle qui sont plus performante et permettent de passer dans moins de porte logique. Au total pour un opérations, on passe par 9 portes logiques, ce qui est très peu.

Lors de ce TP on a pu remarque qu'un processeur ne fonctionne par forcément mieux si le circuit est simple, il fonctionne mieux s'il passe dans le moins de porte logiques possible

même si le circuit est plus compliqué. De plus les processeurs ne fonctionnent qu'en porte logique NAND, on va donc devoir adapter toutes les autres portes logiques pour les réaliser avec des NAND.

Le développement d'un circuit sur logicielle n'est qu'une représentation graphique du circuit, elle est la juste pour pouvoir tester de nouveau circuit et expérimenter. Grace à ces logicielles on peut réaliser des circuit complexe plus facilement et il permet de trouver les équations de calculs plus simplement. A l'aide de ces logicielles, on réalise le produit avec des transistors CMOS et on reproduit le circuit réalisé sur le logicielle. Cela nous permet de réaliser des circuit plus performant et comptant moins de porte logique.