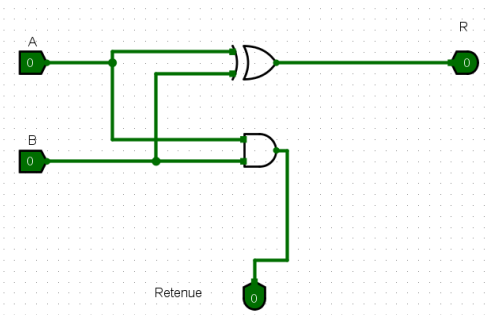
Compte Rendu Architecture TRAN Florian Grp 4-2

Introduction :

Pour commencer l'objectif de notre TP consistait à construire une Unité Arithmétique et logique simple, capable de faire 8 opérations différentes sur 16 bits de données. Nous avons utilisé le logiciel Logisim qui permet de simuler des circuits, nous avons répartis ce TP sur 4 séances sur 2 semaines.

I- Le circuit d'addition – soustraction

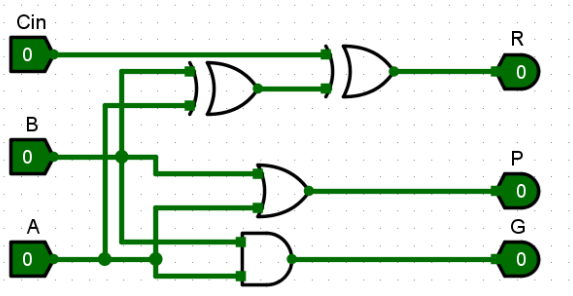
Pour la première étape de la création de l'UAL nous avons commencé par un simple additionneur 1 bit /le demi-additionneur. Ce circuit à 2 entrées A et B, et 2 sorties R le résultat de l'addition et la retenue (r).



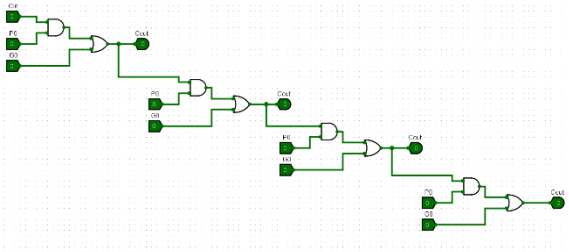
Ensuite nous avons créé le circuit grâce à la formule S = A XOR B et

r = A ^ B.

Le seul problème de ce circuit c'est qu'il ne prend pas de retenue entrante donc nous l'avons modifié pour qu'il ait C_{in} la retenue entrante, et comme cette solution est très couteuse par rapport au porte franchises nous avons rajouté 2 sorties P, qui est vraie ssi les valeurs des entrées sont telles qu'une retenue entrante sera propagée en sortie. G, qui est vraie ssi les valeurs des entrées sont telles qu'une retenue sortante est nécessairement engendrée.



Pour continuer, nous voulons créer un circuit qui permet d'additionner 4 bits. Donc nous devons interpréter P et G avec l'expression : Cout = G v (p ^C_{in}) pour calculer la retenue, mais comme dans l'additionneur 4 bits nous allons utiliser 4 fois l'additionneur 1 bit, donc 4 retenues alors nous avons créé un circuit pour s'en occuper. Tout d'abord nous avons créé un circuit (UCAR) (ci-dessous). Mais il n'était pas optimisé car les calculs doivent attendre le précédant.



Ensuite nous avons créé un nouveau circuit UCAR optimisé et qui sert à calculer les C1, C2, C3, C4 les retenues d'une addition de 4 bits :

C1= (p0^c0) v g0

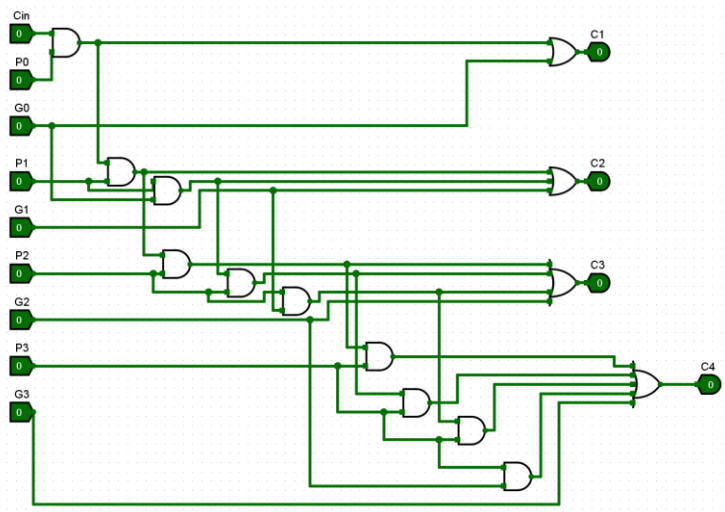
C2= (p1 ^ p0 ^ c0) v (p1 ^ g0) v g1

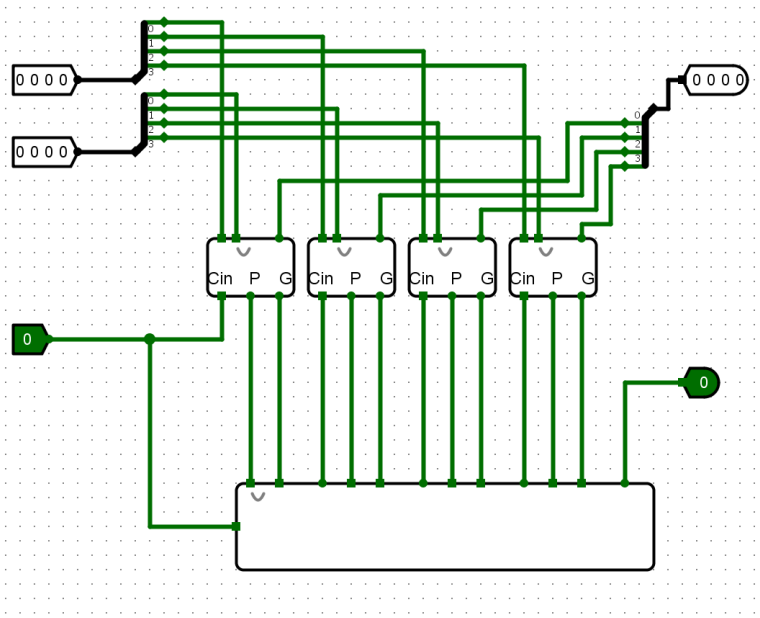
C3= (p2 ^ p1 ^ p0 ^ c0) v (p2 ^ p1 ^ g0) v (p2 ^ g1) v g2

C4= (p3 ^ p2 ^ p1 ^ p0 ^ c0) v (p3 ^ p2 ^ p1 ^ g0) v (p3 ^ p2 ^ g1) v (p3 ^ g2) v g3

Nous avons calculé de manière anticiper la retenue de toutes les sous-addition.

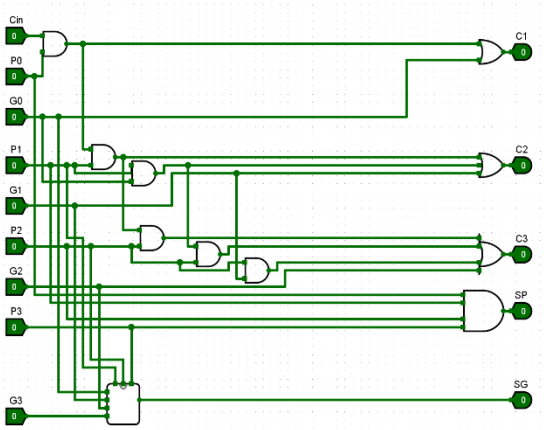
Voici mon circuit UCAR (ci-dessous) qui permet de reproduire ces calculs.



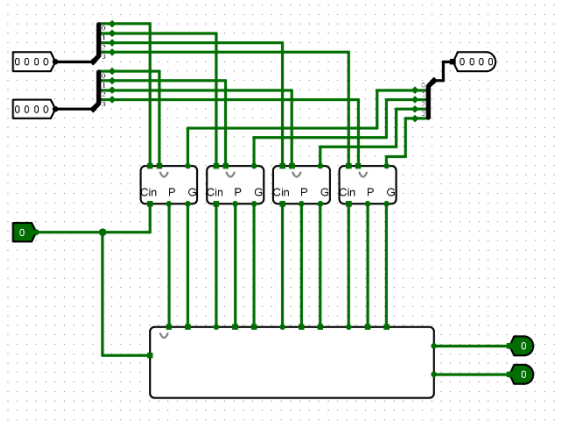


Grâce à l'UCAR nous pouvons à présent utilisé 4 additionneurs 1 bits afin de créer l'additionneur 4 bit :

Ensuite nous avons un peu modifié notre additionneur 4 bit pour qu'il utilise des sorties super P et super G qui sont des super-signaux qui pour super P indique si une retenue entrante sur l'additionneur 4 bis sera propagé en sortie et pour super G indique si une retenue sortante est engendrée par l'additionneur 4 bits. Alors nous avons modifié notre circuit UCAR pour correspondre aux supers P et G.

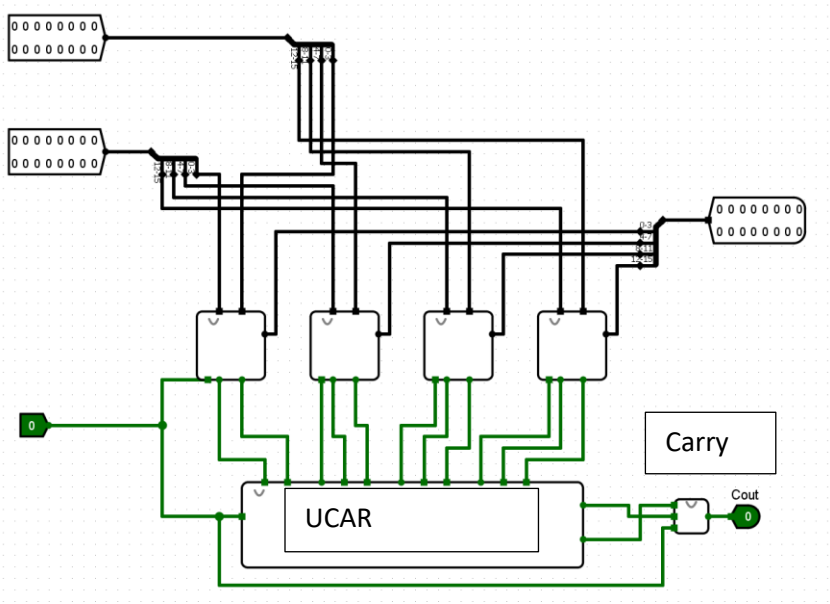


<- UCAR super G et P



<- 4-bit adder super G et P

Ensuite à partir de cet additionneur 4 bits nous voulons créer un additionneur 16 bits. Donc même organisation que notre 4-bit adder super G et P, mais à la place des additionneurs 1bit nous utilisons les 4-bit adder super G et P.



<- additionneur 16-bits

L'UCAR sert à calculer les retenues pour les prochains additionneurs et super G et super P avec la formule : sP = p0 ^p1 ^ p2 ^ p3 et sG = g0 ^g1 ^ g2 ^ g3

Le petit circuit (carry) en sortant de UCAR permet de calculer la retenue sortante Cout=g v (p ^ Cin).

II- L'unité arithmétique et logique

L'unité arithmétique et logique a pour but de faire une opération parmi plusieurs contrôlé par une entrée reliée à un multiplexeur avec une entrée en 3 bits donc 8 commandes différentes. Les fonctions à calculer sont :

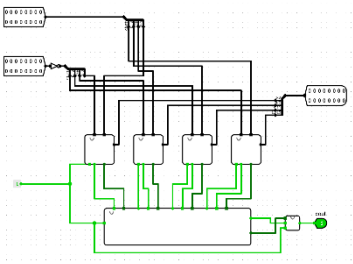
— 000 : S = A + B.

Pour cette fonction c'est l'additionneur 16 bits qu'on va utiliser

— 001 : S = A − B.

Pour cette fonction c'est l'additionneur 16 bits mais nous devons rajouter une constante = 1 en retenue entrante

Afin de réalisé une soustraction en complément à 2:



— 010 : S = A bar

Pour cette fonction on utilise la porte NOT

— 011 : S = A.B.

Pour cette fonction on utilise la porte AND

— 100 : S = A + B.

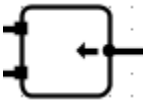
La même que la 000

— 101 : S = A ⊕ B.

Pour cette fonction on utilise la porte XOR

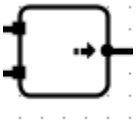
— 110 : décalage à gauche de A de 1 bit (cf. indication ci-dessous).

Pour cette fonction on utilise le décalage logique à gauche



— 111 : décalage à droite arithmétique de A de 1 bit.

Pour cette fonction on utilise le décalage arithmétique à droite



Enfin nous avons ajouté des informations de statuts :

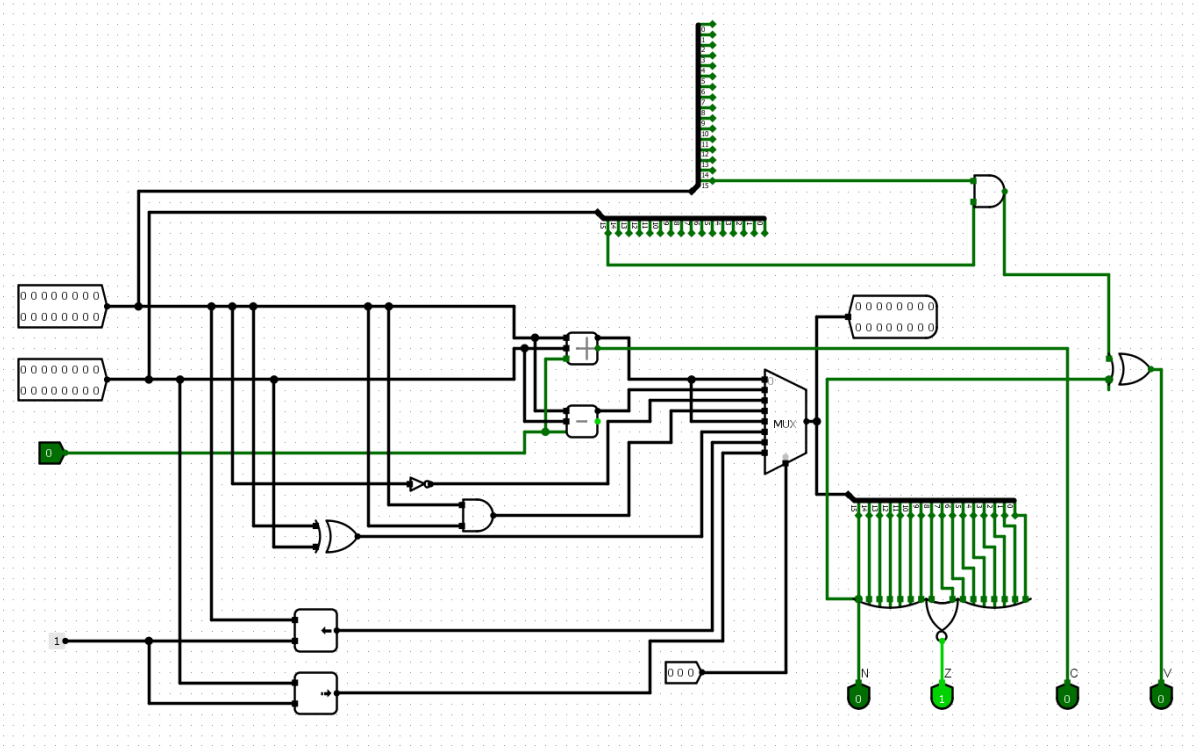
— Z : la sortie vaut 0.

— N : la sortie, interprétée en complément à 2, est négative.

— C : retenue sortante de l’addition.

— V : interprétée en complément à 2, le résultat du calcul déborde de la capacité de codage.

Et le circuit de l'UAL avec le multiplexeur avec les informations de statuts en bas à gauche :



III- Le bilan

Grâce à ce TP nous avons pu prendre connaissance de l'UAL et y découvrir son fonctionnement et mais aussi le fonctionnement d'un processeur dans lequel est implémenté des UAL. Nous avons appris à simuler des circuits comme les additionneurs 1-4-16 bits, utilisé des portes logiques et vu des façons d'optimiser. Ensuite nous avons découvert le multiplexeur avec 8 fonctions différentes et en ressortir d'autres informations comme les bits de statuts.

Les difficultés, elles sont au niveau de l'overflow car dans mon circuit, V (overflow) s'active à ce que quand le résultat dépasse de la soustraction mais pas de l'addition.

La conception matérielle est assez différente du développement logiciel car sur ce logiciel nous ne faisons que des simulations ce qui ne reflète pas la réalité. Il y a beaucoup plus de variable à envisager dans la conception comme la mémoire/optimisation…

Pour finir ce TP m'a fait comprendre l'utilité et le fonctionnement des transistors, des processeurs et m'a permis de voir une autre façon d'utiliser les portes logiques.