**Introduction**

L’avancement de la technologie a apporté plusieurs changements à notre quotidien. En autre, elle a servi à l’améliorer et la facilité. Parmi les myriades de gadgets inventés, l’invention de la calculatrice nous a permis d’effectuer une forte quantité d’opérations arithmétiques. Dans le cadre de ce laboratoire, il faudra concevoir une petite UAL (Unité Arithmérique et Logique), où, similairement à une calculatrice, réalisera non seulement des opérations arithmétiques mais aussi des opérations logique telles que l’addition, la soustraction, la division par 2 et la comparaison. Ainsi, le circuit devra donc posséder un additionneur/soustracteur, un diviseur et un comparateur sur deux entrées de 4 bits. Il y aura également un multiplexeur afin de choisir l’opération désirée. L’objectif de ce laboratoire est la mise en pratique de la conception de circuits en logiques combinatoire, de les simuler et de se familiariser avec l’implantation de ces circuits simples sur une carte ainsi qu’aux notions apprises en cours.

**Les tables de vérités et les équations déduites**

Tableau 1 : Table de vérité pour diviseur pour le UAL

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| a3 | a2 | a1 | a0 | s3 | s2 | s1 | s0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |

Équations déduites :

s1=0

s1= a0

s2= a1

s3 = a2

Tableau 2 : Table de vérité pour comparateur du UAL

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a3 | b3 | a2 | b2 | a1 | b1 | a0 | b0 | s3 | s2 | s1 | s0 |
| x | x | x | x | x | x | 0 | 0 | x | x | x | 1 |
| x | x | x | x | x | x | 0 | 1 | x | x | x | 0 |
| x | x | x | x | x | x | 1 | 0 | x | x | x | 0 |
| x | x | x | x | x | x | 1 | 1 | x | x | x | 1 |
| x | x | x | x | 0 | 0 | x | x | x | x | 1 | x |
| x | x | x | x | 0 | 1 | x | x | x | x | 0 | x |
| x | x | x | x | 1 | 0 | x | x | x | x | 0 | x |
| x | x | x | x | 1 | 1 | x | x | x | x | 1 | x |
| x | x | 0 | 0 | x | x | x | x | x | 1 | x | x |
| x | x | 0 | 1 | x | x | x | x | x | 0 | x | x |
| x | x | 1 | 0 | x | x | x | x | x | 0 | x | x |
| x | x | 1 | 1 | x | x | x | x | x | 1 | x | x |
| 0 | 0 | x | x | x | x | x | x | 1 | x | x | x |
| 0 | 1 | x | x | x | x | x | x | 0 | x | x | x |
| 1 | 0 | x | x | x | x | x | x | 0 | x | x | x |
| 1 | 1 | x | x | x | x | x | x | 1 | x | x | x |

Équations déduites :

F0= a0’b0+ a0b0’

F1= a1’b1+ a1b1’

F2= a2’b2+ a2b2’

F3= a3’b3+ a3b3’

Tableau 3 : Table de vérité du multiplexeur du UAL

| r1 | r0 | x3 | x2 | x1 | x0 | y3 | y2 | y1 | y0 | z3 | z2 | z1 | z0 | s3 | s2 | s1 | s0 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | X | X | X | X | X | X | X | x | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | X | X | X | X | X | X | X | x | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | X | X | X | X | X | X | X | x | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | X | X | X | X | X | X | X | x | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | X | X | X | X | X | X | X | x | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | X | X | X | X | X | X | X | x | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 | X | X | X | X | X | X | X | x | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | X | X | X | X | X | X | X | x | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | X | X | X | X | X | X | X | x | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | X | X | X | X | X | X | X | x | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | X | X | X | X | X | X | X | x | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | X | X | X | X | X | X | X | x | 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | X | X | X | X | X | X | X | x | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | X | X | X | X | X | X | X | x | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | X | X | X | X | X | X | X | x | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | X | X | X | X | X | X | X | x | 1 | 1 | 1 | 1 |
| 0 | 1 | X | X | X | x | 0 | 0 | 0 | 0 | X | X | X | x | 0 | 0 | 0 | 0 |
| 0 | 1 | X | X | X | x | 0 | 0 | 0 | 1 | X | X | X | x | 0 | 0 | 0 | 1 |
| 0 | 1 | X | X | X | x | 0 | 0 | 1 | 0 | X | X | X | x | 0 | 0 | 1 | 0 |
| 0 | 1 | X | X | X | x | 0 | 0 | 1 | 1 | X | X | X | x | 0 | 0 | 1 | 1 |
| 0 | 1 | X | X | X | x | 0 | 1 | 0 | 0 | X | X | X | x | 0 | 1 | 0 | 0 |
| 0 | 1 | X | X | X | x | 0 | 1 | 0 | 1 | X | X | X | x | 0 | 1 | 0 | 1 |
| 0 | 1 | X | X | X | x | 0 | 1 | 1 | 0 | X | X | X | x | 0 | 1 | 1 | 0 |
| 0 | 1 | X | X | X | x | 0 | 1 | 1 | 1 | X | X | X | x | 0 | 1 | 1 | 1 |
| 0 | 1 | X | X | X | x | 1 | 0 | 0 | 0 | X | X | X | x | 1 | 0 | 0 | 0 |
| 0 | 1 | X | X | X | x | 1 | 0 | 0 | 1 | X | X | X | x | 1 | 0 | 0 | 1 |
| 0 | 1 | X | X | X | x | 1 | 0 | 1 | 0 | X | X | X | x | 1 | 0 | 1 | 0 |
| 0 | 1 | X | X | X | x | 1 | 0 | 1 | 1 | X | X | X | x | 1 | 0 | 1 | 1 |
| 0 | 1 | X | X | X | x | 1 | 1 | 0 | 0 | X | X | X | x | 1 | 1 | 0 | 0 |
| 0 | 1 | X | X | X | x | 1 | 1 | 0 | 1 | X | X | X | x | 1 | 1 | 0 | 1 |
| 0 | 1 | X | X | X | x | 1 | 1 | 1 | 0 | X | X | X | x | 1 | 1 | 1 | 0 |
| 0 | 1 | X | X | X | x | 1 | 1 | 1 | 1 | X | X | X | x | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | X | X | X | x | X | X | X | X | x | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | X | X | X | x | X | X | X | X | x | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | X | X | X | x | X | X | X | X | x | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | X | X | X | x | X | X | X | X | x | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | X | X | X | x | X | X | X | X | x | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | X | X | X | x | X | X | X | X | x | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | X | X | X | x | X | X | X | X | x | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | X | X | X | x | X | X | X | X | x | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | X | X | X | x | X | X | X | X | x | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | X | X | X | x | X | X | X | X | x | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | X | X | X | x | X | X | X | X | x | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | X | X | X | x | X | X | X | X | x | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | X | X | X | x | X | X | X | X | x | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | X | X | X | x | X | X | X | X | x | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | X | X | X | x | X | X | X | X | x | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | X | X | X | x | X | X | X | X | x | 1 | 1 | 1 | 1 |
| 1 | 1 | X | X | X | x | X | X | X | x | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | X | X | X | x | X | X | X | x | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | X | X | X | x | X | X | X | x | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | X | X | X | x | X | X | X | x | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | X | X | X | x | X | X | X | x | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | X | X | X | x | X | X | X | x | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | X | X | X | x | X | X | X | x | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | X | X | X | x | X | X | X | x | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | X | X | X | x | X | X | X | x | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | X | X | X | x | X | X | X | x | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | X | X | X | x | X | X | X | x | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | X | X | X | x | X | X | X | x | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | X | X | X | x | X | X | X | x | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | X | X | X | x | X | X | X | x | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | X | X | X | x | X | X | X | x | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | X | X | X | x | X | X | X | x | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Équations déduites :

S0 = r0’ x0 + r0r1’ y0 + r0r1 z0

S1 = r0’ x1 + r0r1’ y1 + r0r1 z1

S2 = r0’ x2 + r0r1’ y2 + r0r1 z2

S3 = r0’ x3 + r0r1’ y3 + r0r1 z3

**Schémas et descriptions des modules**

Figure 1 : Schéma de l’additionneur/ soustracteur pour 4bit (4bit full adder) :

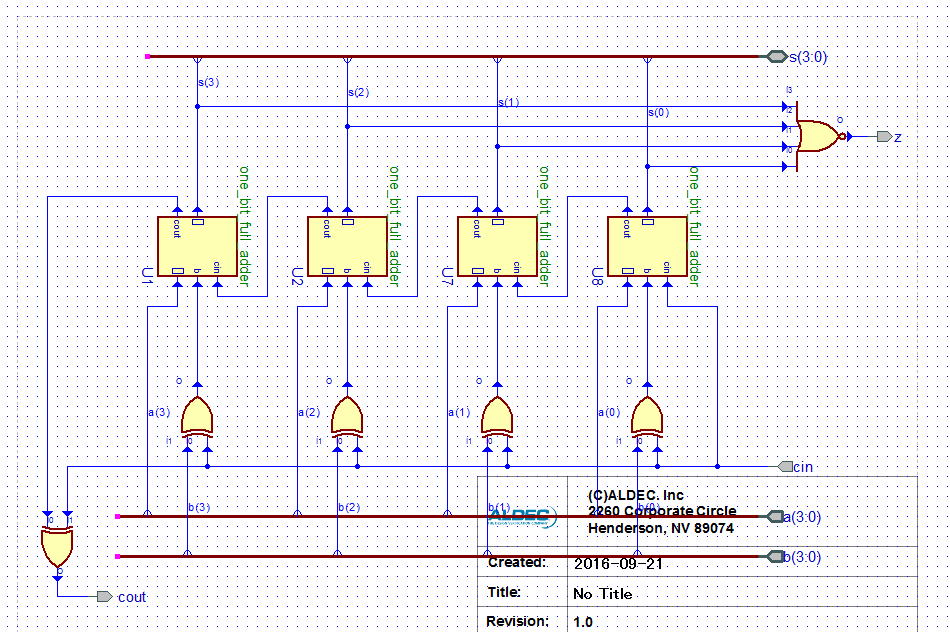
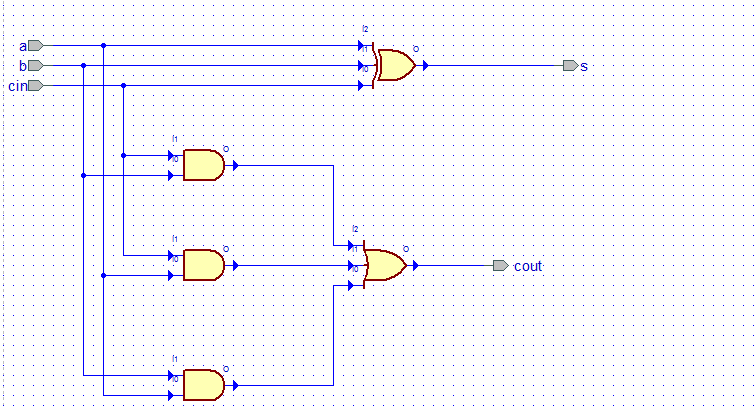
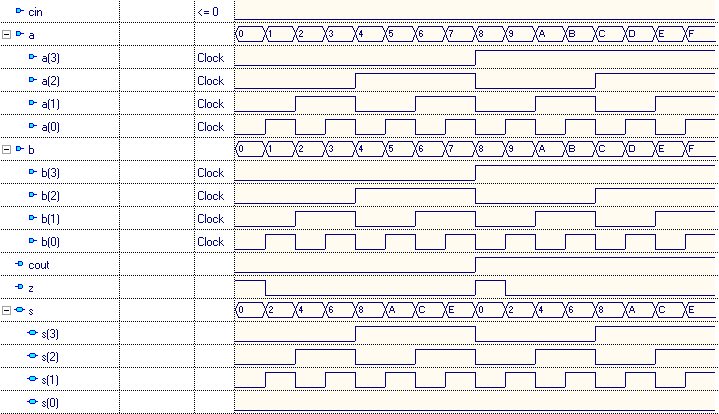


Figure 2 : Schéma de l’additionneur/ soustracteur pour 1bit (one bit full adder)

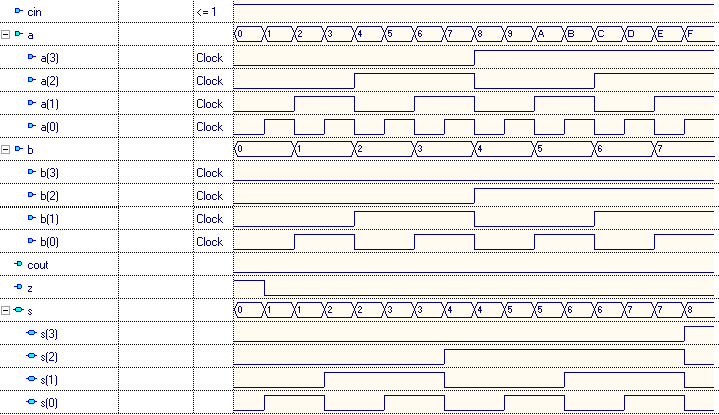
Le 4 bit full adder permet d’additionner ou de soustraire deux valeurs de 4 bit dépendant de la valeur de cin. Pour cin = 0, nous avons une addition. Au contraire, pour cin = 1, nous avons une soustraction.

Figure 3 : Simulation de l’addition du 4bit full adder



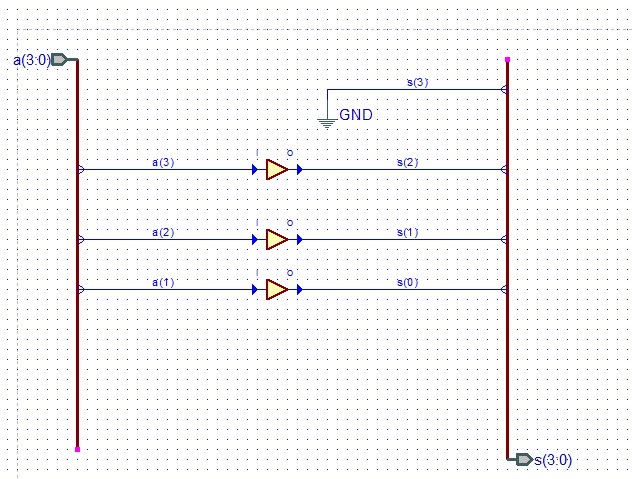
On observe dans la figure 3 que la valeur de *s* correspond à l’addition de *a* et de *b*.

Figure 4 : Simulation de la soustraction du 4bit full adder



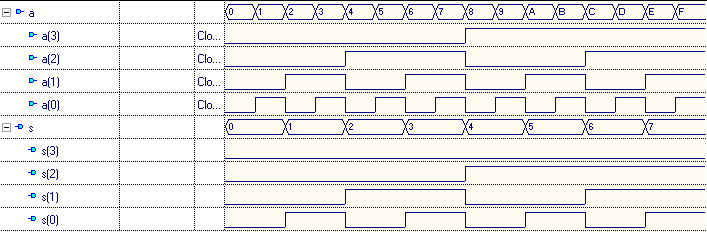
On observe dans la figure 3 que la valeur de *s* correspond à la valeur de *a* moins la valeur de *b.*

Figure 5: Schéma pour le module du diviseur



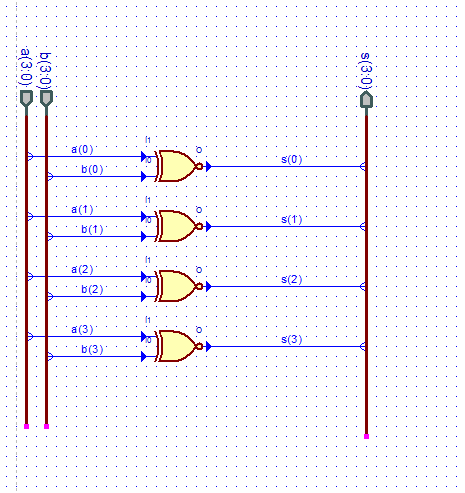
Avec la table de vérité de la division binaire par deux, on peut remarquer qu’à chaque division, on décale les bits d’une place vers la droite : *a3* devient *s2*, *a2* devient *s1* et *a1* devient *s0*. Donc, nous avons relié *s3* avec *a2*, *s2* avec *a1* et *s1* avec *a0*. Nous avons utilisé un buffer pour pouvoir associer des noms différents aux lignes de chaque côté du buffer. Dans le tableau, nous avons que *s(0)* est toujours égale à 0 puisque 0001/2=0. Nous avons utilisé ground puisque celui-ci fournit toujours une valeur de 0.

Figure 6: Simulation du module de diviseur



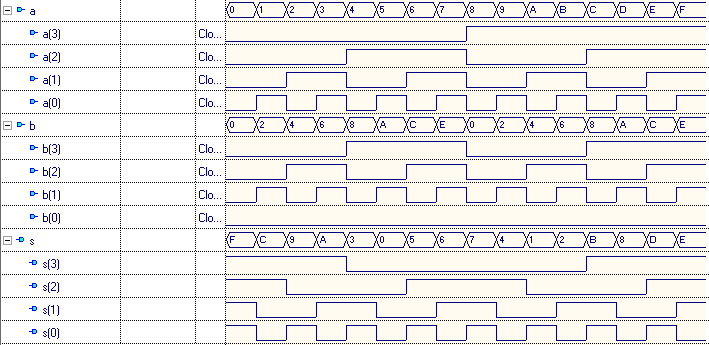
Comme on peut l’observer avec le *timing driagram*: *a(3)* correspond à *s(2)*, *a(2)* correspond à *s(1)* et *a(1)* correspond à *s(0)*. On observe aussi que la valeur de *s(3)* reste toujours 0 puisque sa valeur provient du ground.

Figure 7 : Schéma du module de comparateur



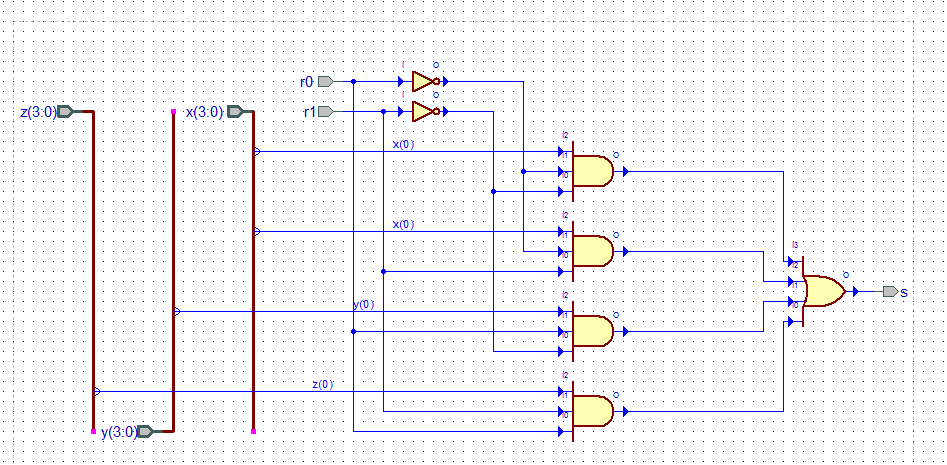
Pour comparer deux sources différentes, nous avons construit un circuit qui compare les bus *a* et *b* bit par bit : on compare le premier bit de *a* avec le premier bit de *b*, le deuxième bit de *a* avec le deuxième bit de *b* et ainsi de suite. La porte XNOR retourne une valeur de 1 si les deux bit sont égaux et une valeur de 0 si les bits sont différents. Les portes XNOR envoient alors le résultat dans le bus *s*.

Figure 8 : Simulation du module de comparateur



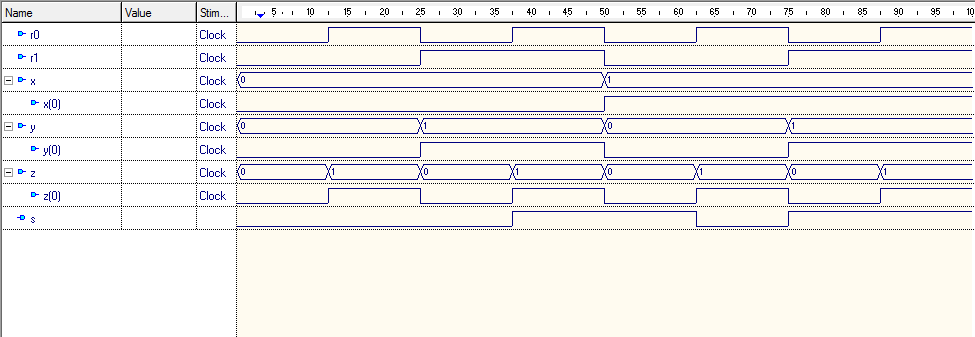
Dans notre simulation, nous avons fait varier les valeurs de *a* et de *b*. Prenons par exemple la comparaison quand *a* = 0111 et *b* = 1110, nous obtenons *s* = 0110. Cette valeur nous indique que les premiers et le quatrième bit de *a* et *b* sont différents tandis que leur deuxième et troisième bit sont identiques.

Figure 9 : Schéma du multiplexeur pour 1bit d’entrée



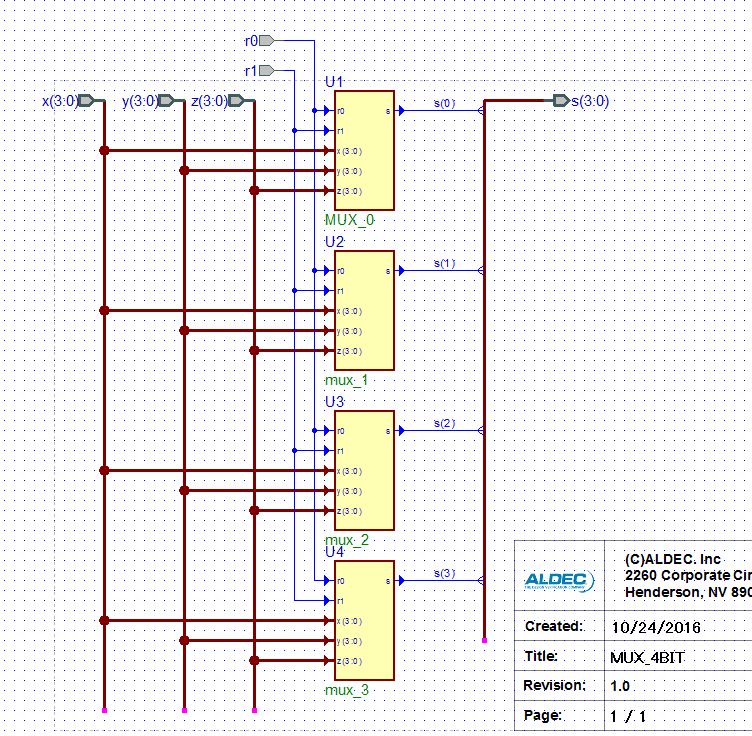
Le multiplexeur 4 à 1 nous permet de sélectionner une des 4 valeurs à l’aide de deux sélectionneurs (*r0* et *r1*). 4 combinaisons différentes de *r0* et de *r1* nous permettent de choisir un des 4 entrées : *r0* = 0, *r1* = 0 donne l’addition; *r0* = 0, *r1* = 1 donne la soustraction; *r0* = 1, *r1* = 0 donne la division par 2 et *r0* = 1, *r1* = 1 donne la comparaison. Pour obtenir l’addition, nous avons mis *r0, r1* et *a(0)* comme entrées d’une porte ET. Puisque pour obtenir un résultat de 1 il faut que toutes les entrées soient 1, il faut changer les valeurs de *r0* et *r1* à 1 à l’aide d’inverseurs. Donc si *r0* = 0, *r1* = 0 et *a(0)* = 1, nous aurons 1. Si par exemple : *r0* = 0 et *r1* = 1, il ne faut pas inverser les deux valeurs. En effet, il faut dans ce cas seulement mettre un inverseur devant *r0*.

Figure 9 : Schéma du multiplexeur pour 1bit d’entrée



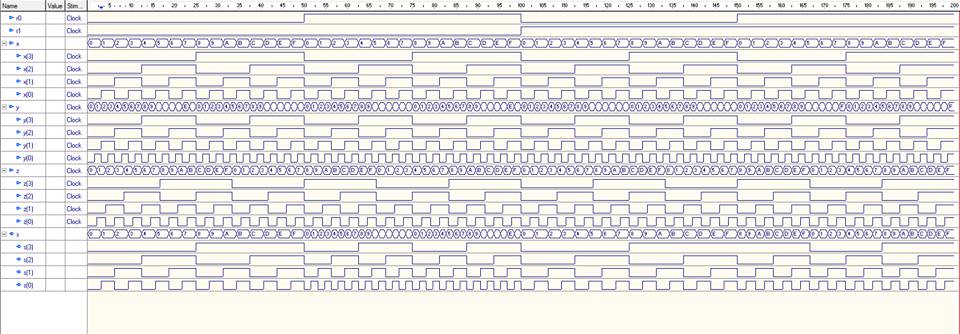
Le schéma montre que le multiplexeur 1bit à 4 entrées fonctionne comme prévu : *r(0)* = 0 *r(1)* = 0 sélectionne bel et bien la valeur de *x(0)* = *s(0)*, *r(0)* = 1 *r(1)* = 0 sélectionne la valeur de *y(0)* = *s(0)* et ainsi de suite.

Figure 10 : Schéma du multiplexeur (4bit)



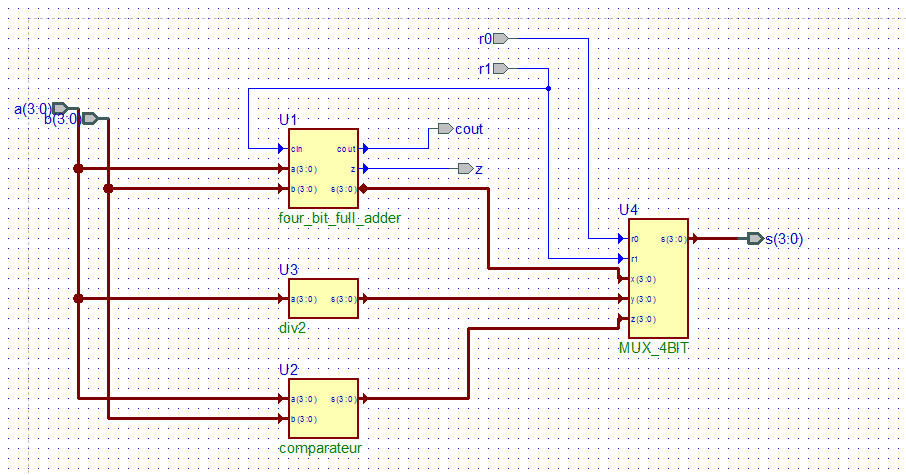
Pour obtenir un multiplexeur 4 bit 4 à 1, nous avons relié 4 multiplexeurs 1 bit 4 à 1 qui sélectionnent un des 4 bit chacun. Ils ont pour entrées 3 bus (*x, y et z)*, *r0*, *r1* et chacun a comme sortie un des 4 bits du bus *s*. Par exemple : si *r0* = 1 et *r1* = 0, ce qui correspond au bus *y* (division par 2), *U1* sélectionne le premier bit de *y*, *U2* sélectionne le deuxième bit de *y* et ainsi de suite. Par la suite, les 4 multiplexeurs à 1 envoient leur données dans un bus commun *s*.

Figure 11 : Schéma du multiplexeur pour 4bit d’entrée



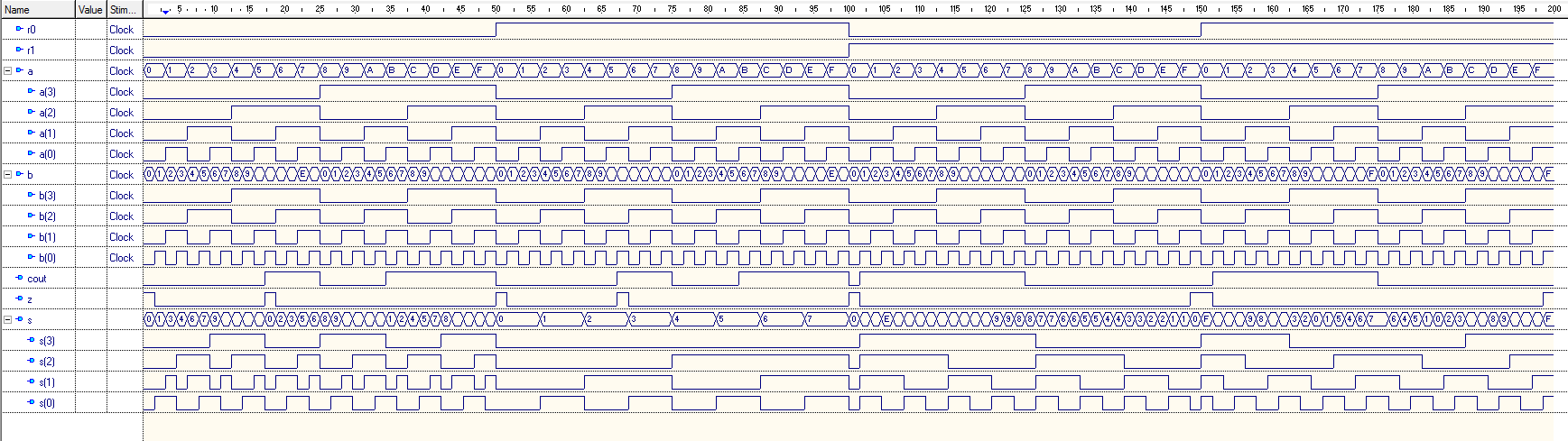
Pour la simulation, nous avons choisi des valeurs différentes pour *x, y et z* pour voir clairement lequel des 3 bus est sélectionné par *r0* et *r1*. Dans le premier quart, *x* est sélectionné puisqu’on a *r0* = 0 et *r1* = 0. Le deuxième quart, nous avons encore *x* de sélectionné (l’addition et la soustraction partagent la même entrée dans le multiplexeur). Dans le premier quart, *y* est sélectionné puisqu’on a *r0* = 0 et *r1* = 1. Dans le premier quart, *x* est sélectionné puisqu’on a *r0* = 1 et *r1* = 1.

Figure 12 : Schéma du UAL



Dans le dessin final pour le UAL nous avons 2 entrées (*a* et *b*). Ces entrées sont reliées au 4 bit full adder, diviseur par 2 et au comparateur avec des bus puisque les valeur de *a* et de *b* sont à 4 bit. Les 3 modules sont ensuite reliées au multiplexeur permettant de choisir l’opération désirée avec *r0* et *r1*. Finalement, le multiplexeur transmet le résultat de l’opération désirée dans le bus *s*.

Figure 13 : Simulation du UAL



Dans cette simulation, nous avons fait varier *r0, r1, a* et *b* dans le but de simuler les entrées possibles. Après avoir vérifié le résultat *s*, nous pouvons confirmer que le UAL fonctionne comme prévu.

Conclusion

En conclusion, l’objectif du laboratoire, qui était de se familiariser avec les circuits à logique combinatoire et avec le matériel utilisé (programme Active HDL et l’implantation d’une carte). Nous avons réussi à dessiner chacune des composantes à l’aide de la tableau de vérité(diviseur, comparateur et multiplexeur). De plus, nous avons réussi à insérer les modules dans un seul dessin et nous avons réussi à les simuler tout en obtenant les résultats prévus. Finalement, nous avons aussi réussi l’implémentation sur la carte FPGA avec les même résultats que dans les simulations.