Lab #3 : Arithmetic Logic Unit

CEG 2536 – ARCHITECTURE DES ORDINATEURS I

Fall 2018

Ecole de Génie Electrique et Science Informatique

Université d'Ottawa

Professeur : Dr. Mohamed Ali Ibrahim

Groupe

Etudiant 1 : Rayold Rakotonomenjanahary 8884585

Etudiant 2 : Mahmoud Lafdaoui 300032370

PARTIE THEORIQUE

1. Introduction

Le but de ce laboratoire était de permettre aux étudiants de très bien maitriser l'unité de circuit arithmétique et logique vue dans le cours. Pour ce faire, il fallait concervoir, compiler, simuler et tester chacun des composants de ce circuit. Le circuit devrait effectuer 16 opérations : 8 logiques et 8 arithmétiques. Chaque opérateur du circuit était composé de 4 bits. Cette unité donnait comme sortie 4 bits venant soit de l'unité arithmétique soit logique et aussi 4 autres venant d'un circuit d'état.

2. Matériel et fournitures

* Quartus II (édition de l'étudiant ou l'édition web)

* Carte Altera DE2-115 avec - Câble USB-Blaster - Alimentation 12 VDC, 2A

3. Discussion de problèmes et de solutions algorithmiques

Pour pouvoir concevoir l'unité de commande il nous fallait d'abord trouver comment implémenter le circuit logique, arithmétique et d'état.

Vu que les opérateurs de ce circuit était composé de 4 bits chacun, nous étions dans l'obligation d'utiliser des registres pour ces opérateurs.

1. Circuit arithmétique

Étant donné que la majorité d'opérations effectuées dans ce circuit était des additions, il fallait utiliser un additionneur dans ce circuit. Et pour arriver à une solution sans utiliser trop de composants, on a décidé d'utiliser un additionneur 1 bit pour chacune des opérations et les passer à travers un multiplexeur pour sélectionner une à la fois. Ensuite on a utilisé 4 symboles de cet additionneur pour la réalisation complète de ce circuit.

2. Circuit logique

Étant donné que la majorité d'opérations effectuées dans ce circuit était des opérations « logiques » (pour effectuer a et b, a ou b, a ou-exclusif b,...) et de décalages, il fallait utiliser des portes logiques Et, Ou, Ou-exclusif pour les réaliser. Et pour arriver à une solution sans utiliser trop de composants, nous avions utilisé un multiplexeur qui recevait les opérations pour 1 bit d'abord et par la suite effectué 4 copies du symbole du circuit pour la réalisation complète de ce circuit.

3. Circuit d'état

Un circuit d'état est un circuit qui permet soit de prévenir un problème lors de l'exécution d'une opération dans le circuit, de dire quel type d'opération est exécuté ou de donner une information sur le résultat de l'opération.

Pour la réalisation de notre circuit d'état, nous avions utilisé des portes logiques qui avaient comme entrées certaines sorties du circuit arithmétique ou logique et des sélecteurs du multiplexeur (suivant les énoncés du prélab et les notes de cours).

4. Problèmes rencontrés

Le plus grand problème rencontré lors de cette analyse était la décision sur l'utilisation d'un circuit qui réalisait d'abord l'opération avec 1 bit ou alors un circuit qui faisait une seule opération avec les 4 bits et en faire plusieurs pour les passer dans le multiplexeur. Au début, le dernier énoncé était celui que nous avions décidé d'effectuer mais par la suite, plusieurs complications nous étaient parvenues et nous avions décidé de reprendre avec la première approche

En utilisant ces 3 circuits et des registres et multiplexeurs nous sommes parvenus à réaliser l'unité de commande.

PARTIE CONCEPTION

L'unité arithmétique et logique est un élément essentiel du l'unité central. Dans ce laboratoire, il fallait implémenter l'unité arithmétique et logique au complet (ALU). Il y a eu plusieurs étapes pour arriver au résultat final. Les étapes sont discutées ci-dessous :

Composants utilisés

1. Registre 4 bits

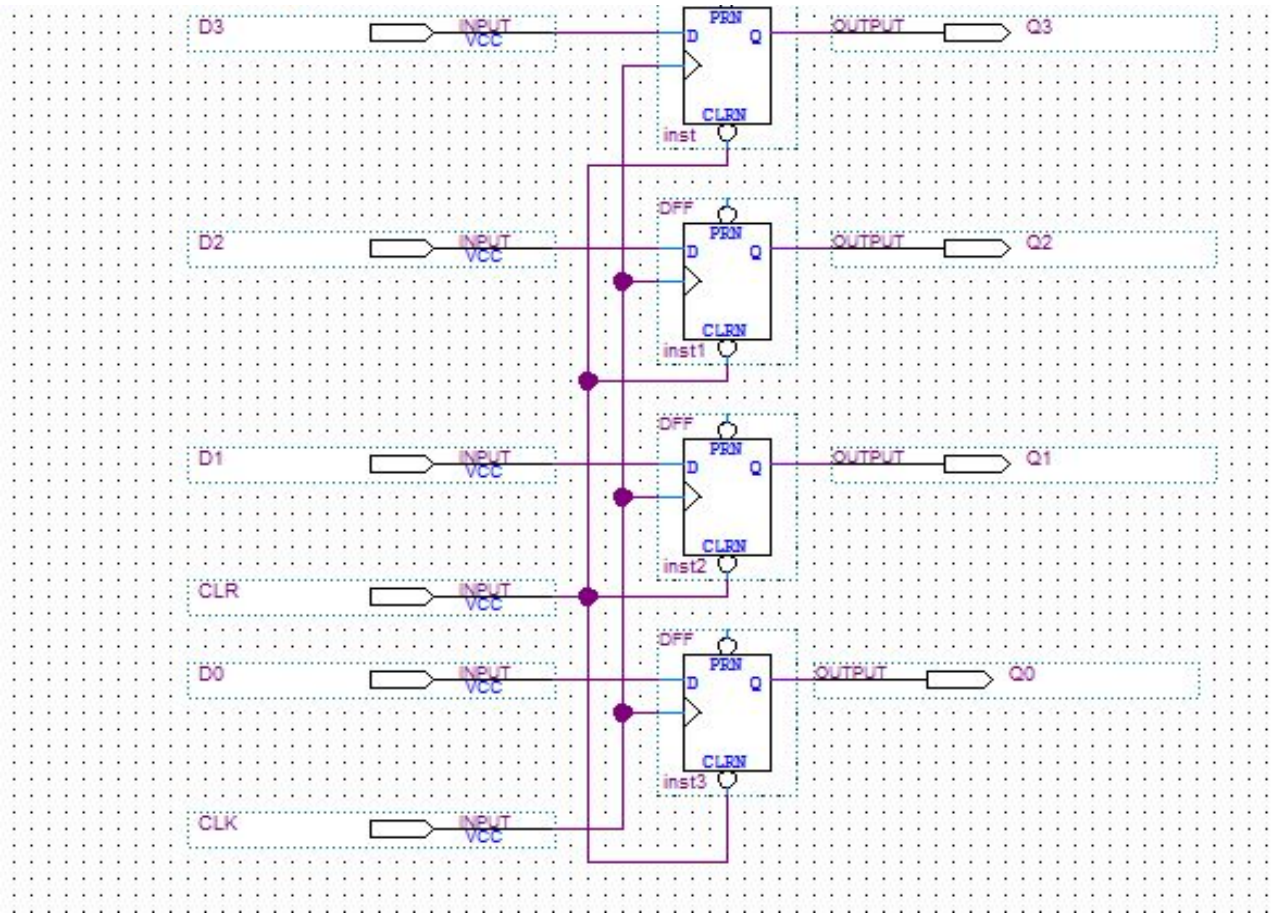
Le registre 4 bits a pour fonction de garder les valeurs des sélecteurs, des opérandes, du registre d'état et finalement la réponse finale. Pour le registre 4 bits, on a utilisé 4 bascules de type D, chaque bascules on des entrées indépendantes, des sorties indépendantes. L'horloge est la même pour les 4 bascules et c'est pareil pour le « clear » les 4 bascules ont la même entrée pour le « clear ». Après avoir créé le registre nous avons fait la compilation du diagramme puis créé un symbole pour pourvoir l'utiliser pour la suite de l'expérience :

Flow Summary	
Flow Status	Successful - Sun Nov 15 15:18:25 2015
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	lab3
Top-level Entity Name	registre
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final

Les ressources utilisées par le registre sont énoncé ci-dessous :

Total logic elements	4 / 114,480 (< 1 %)
Total combinational functions	0 / 114,480 (0 %)
Dedicated logic registers	4 / 114,480 (< 1 %)
Total registers	4
Total pins	10 / 529 (2 %)
Total virtual pins	0
Total memory bits	0 / 3,981,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 532 (0 %)
Total PLLs	0 / 4 (0 %)

Le diagramme du circuit :



2. Circuit logique et de décalage

Le circuit logique et de décalage effectue 8 opérations spécifiques.

1-Réinitialiser C

Cette opération met la valeur des sorties à 0 pour chaque bits. On a donc utilisé le symbole **GND** (valeur

de 0) pour cette opération pour chaque bit.

2-Fixer C
Cette opération met la valeur des sorties à 1 pour chaque bits. On a donc utilisé le symbole **VCC** (valeur de 1) pour cette opération pour chaque bit.

3-A et B
Cette opération effectue l'opération logique A et B entre chaque bits respectifs de A et B. On a donc mis une porte **ET** qui a pour entrées A et B.

4-A ou B
Cette opération effectue l'opération logique A ou B entre chaque bits respectifs de A et B. On a donc mis une porte **OU** qui a pour entrées A et B.

5-A ou exclusif B
Cette opération effectue l'opération logique A ou exclusif B entre chaque bits respectifs de A et B. On a donc mis une porte **XOR** qui a pour entrées A et B.

6-Réinitialiser les bits A sélectionner par le masque B
Cette opération effectue l'opération logique de A et B' en tre chaque bits respectifs de A et B. On a donc mis une porte **NON** pour B et ensuite une porte **ET** où entrent A et B'.

7-Décalage arithmétique vers la gauche de A
Cette opération effectue une multiplication par 2 de A. Pour cela chaque bits de A se décalent vers la gauche et le bit le plus à droite reçoit une valeur de 0 dont on a mis un **GND**.

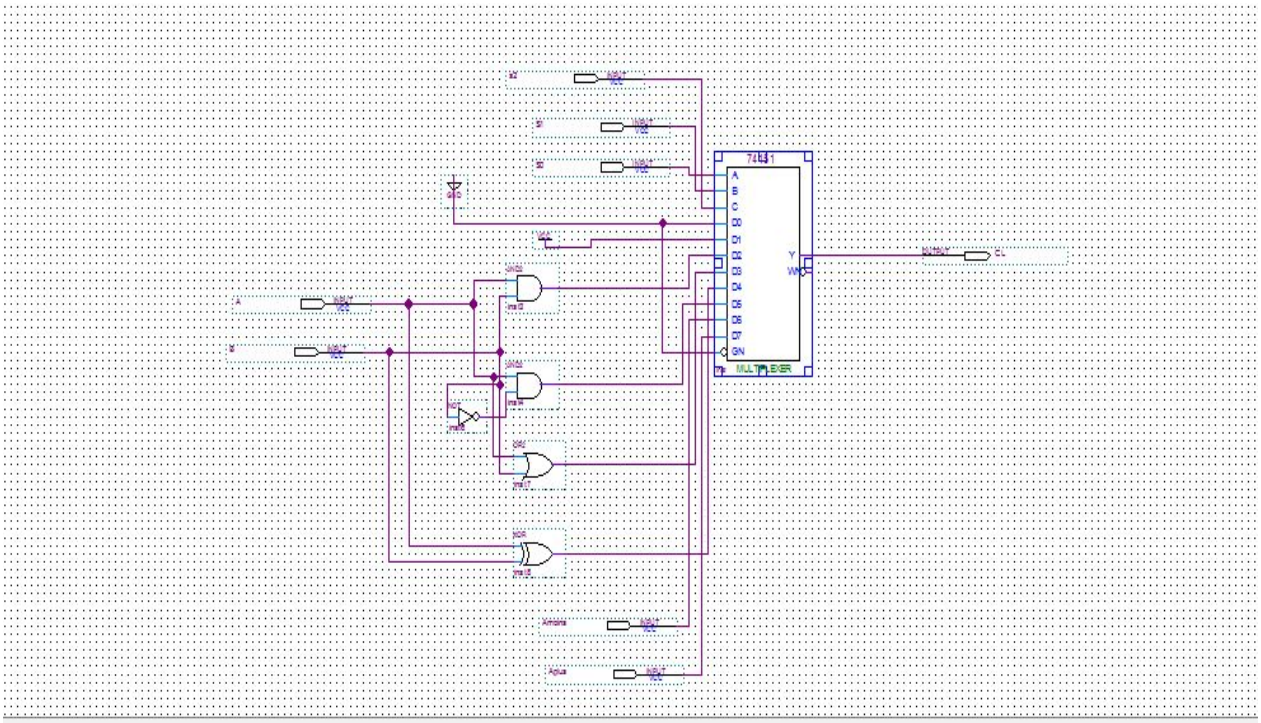
8-Décalage arithmétique vers la droite de A
Cette opération effectue une division par 2 de A. Pour cela chaque bits de A se décalent vers la droite et le bit le plus à gauche reste le **même** que précédemment.

Un multiplexeur 8 à 1 est utiliser pour pouvoir sélectionner les opérations, et il y a 3 bit de sélections S0, S1 et S2. Après avoir créé le circuit nous avons fait la compilation du diagramme puis créé un symbole pour pourvoir l'utiliser pour la suite de l'expérience. On a utilisé 4 symbole du circuit logique 1 bit pour pouvoir faire celui de 4 bit.

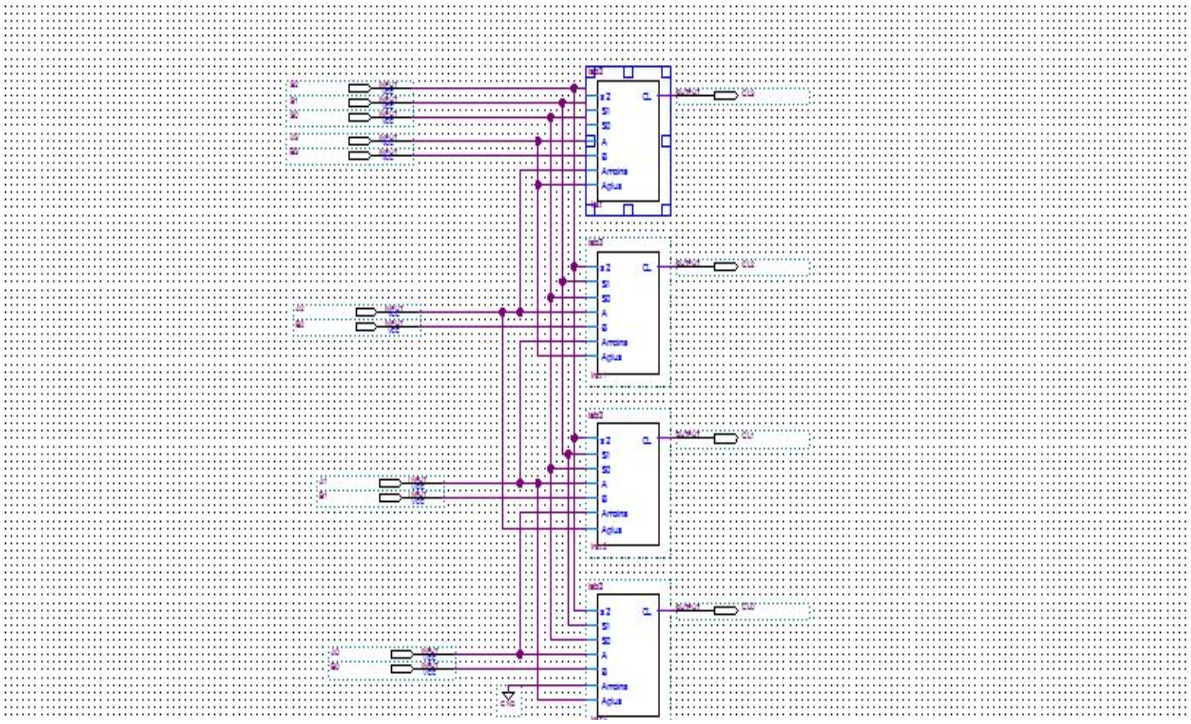
Compilation du circuit logique et de décalage 4 bits:

Flow Summary	
Flow Status	Successful - Sun Nov 15 16:44:23 2015
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	lab3
Top-level Entity Name	clogique4bits
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final
Total logic elements	17 / 114,480 (< 1 %)
Total combinational functions	17 / 114,480 (< 1 %)
Dedicated logic registers	0 / 114,480 (0 %)
Total registers	0
Total pins	15 / 529 (3 %)
Total virtual pins	0
Total memory bits	0 / 3,981,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 532 (0 %)
Total PLLs	0 / 4 (0 %)

Diagramme du circuit pour 1 bit :

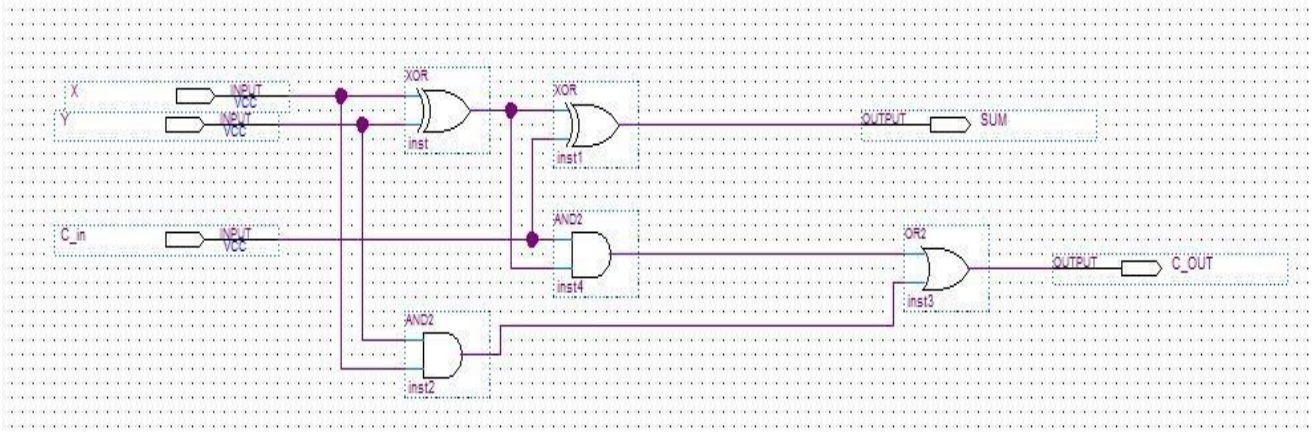


-Un symbole composé des éléments précédent afin de réaliser l'opération pour 4 bits.



3. Additionneur 1 bit

Les éléments utilisés pour la réalisation de l'additionneur sont des portes logiques, XOR, OU, ET.
Cet additionneur sera utile dans le circuit arithmétique.



4. Circuit arithmétique

Le circuit arithmétique effectue 8 opérations spécifiques.
1-A+B
Cette opération additionne A et B pour chaque bit respectivement. La valeur de A et B ne changent pas ils sont additionnés directement. Donc on a une entrée A puis B pour chaque entrée.

2-A+B+1
Cette opération additionne A et B pour chaque bit respectivement. La valeur de A et B ne changent pas ils sont additionnés directement. Donc on a une entrée A puis B pour chaque entrée. La valeur obtenue est ensuit incrémenté par 0001 dans l'additionneur.

3-A
Cette opération ne fais rien à la valeur de A, le résultat qui en sort est toujours A.

4-A+1
Cette opération incrément la valeur de A par un. La valeur des A qui rentre dans le multiplexeur est la même et c'est dans l'additionneur qu'elle est incrémenté par 0001.

5- A+B'
Cette opération additionne A et B' pour chaque bit respectivement. La valeur de A ne change et B passe par une porte NON. Et sont ensuite additionné dans l'additionneur.

6-A+B'+1
Cette opération additionne A et B' pour chaque bit respectivement puis incrémenté par un. La valeur de A ne change et B passe par une porte NON. Et sont ensuite incrémenté par 0001 dans l'additionneur.

7- A'
Cette opération complémente A et ensuite incrément de 1, ce qui donne le complément à 1 de A; pour chaque bit respectivement. Donc A entre dans une porte NON.

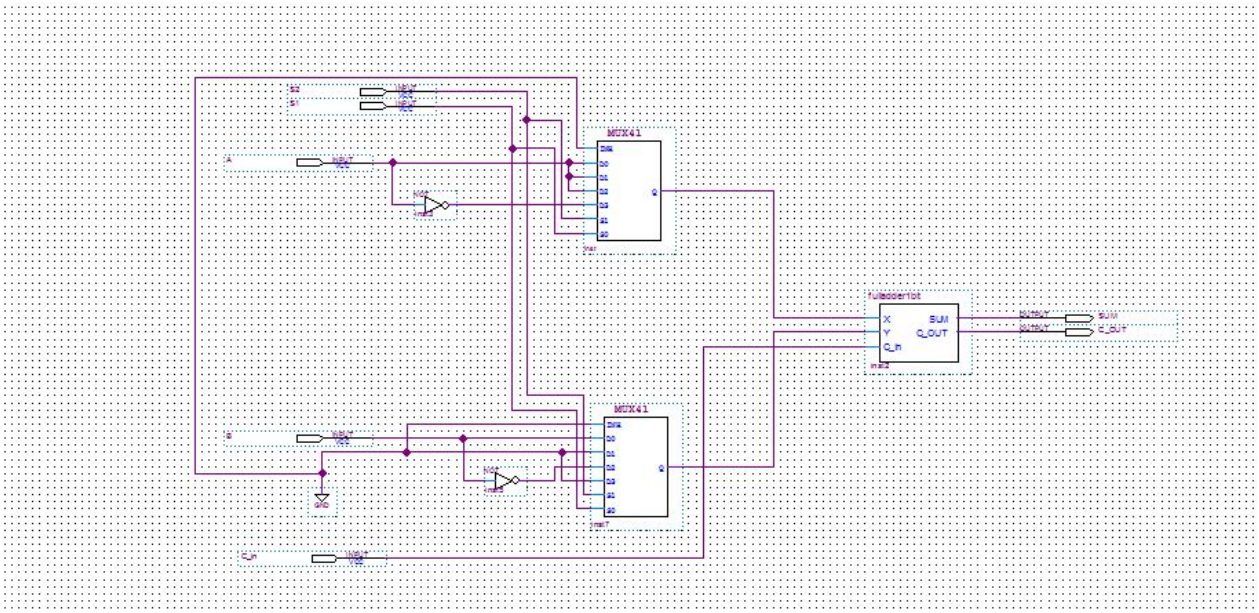
8- A'+1
Cette opération complémente A et ensuite incrémente de 1, ce qui donne le complément à 2 de A; pour chaque bit respectivement. Donc A entre dans une porte NON et ensuite incrémenté par 0001 dans l'additionneur.

On utilise un multiplexer pour sélectionner les valeurs de A pour chaque opération et un autre multiplexeur qui a la même fonction pour les valeurs de B. Les valeurs sélectionnées vont dans un additionneur, qui a comme entré A, B et aussi le C_In qui est le sélecteur S0 $\rightarrow$ C_In = S0
Le circuit arithmétique est plus complexe que le circuit logique et décalage car il y a 1 multiplexeur en plus et aussi un additionneur. Après avoir créé le circuit nous avons fait la compilation du diagramme puis créé un symbole pour pourvoir l'utiliser pour la suite de l'expérience

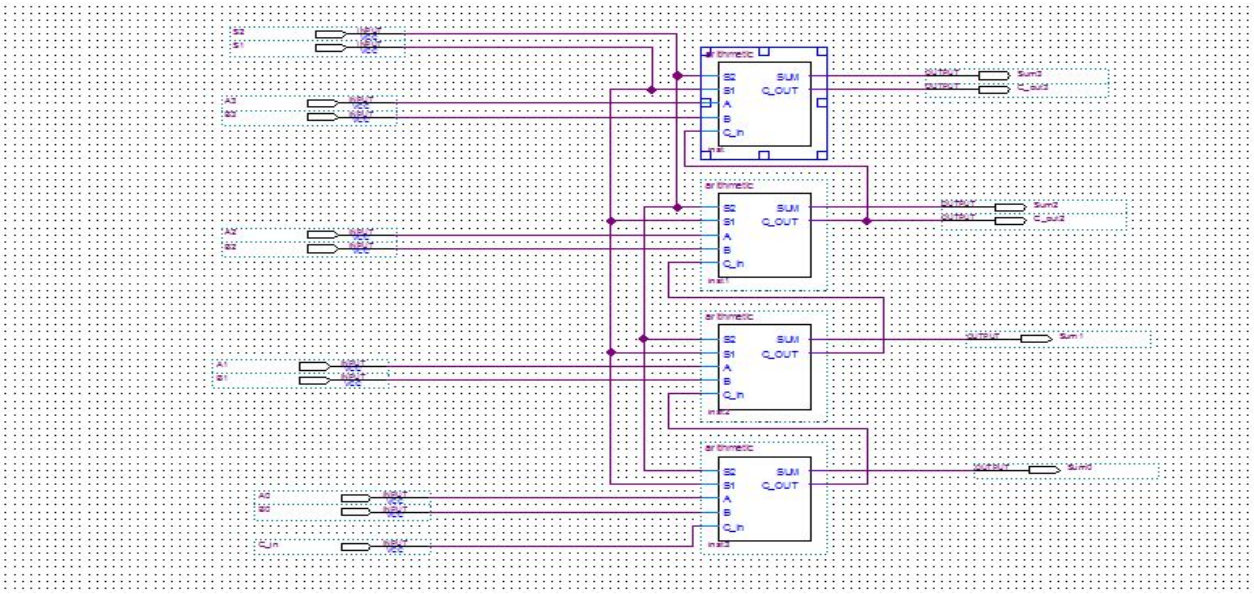
Compilation du circuit arithmétique 4bit :

Flow Summary	
Flow Status	Successful - Sun Nov 15 16:52:59 2015
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	lab3
Top-level Entity Name	arithmetic4bit
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final
Total logic elements	20 / 114,480 (< 1 %)
Total combinational functions	20 / 114,480 (< 1 %)
Dedicated logic registers	0 / 114,480 (0 %)
Total registers	0
Total pins	17 / 529 (3 %)
Total virtual pins	0
Total memory bits	0 / 3,981,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 532 (0 %)
Total PLLs	0 / 4 (0 %)

Diagramme du circuit arithmétique 1 bit :



-Un symbole composé des éléments précédent afin de réaliser l'opération pour 4 bits.



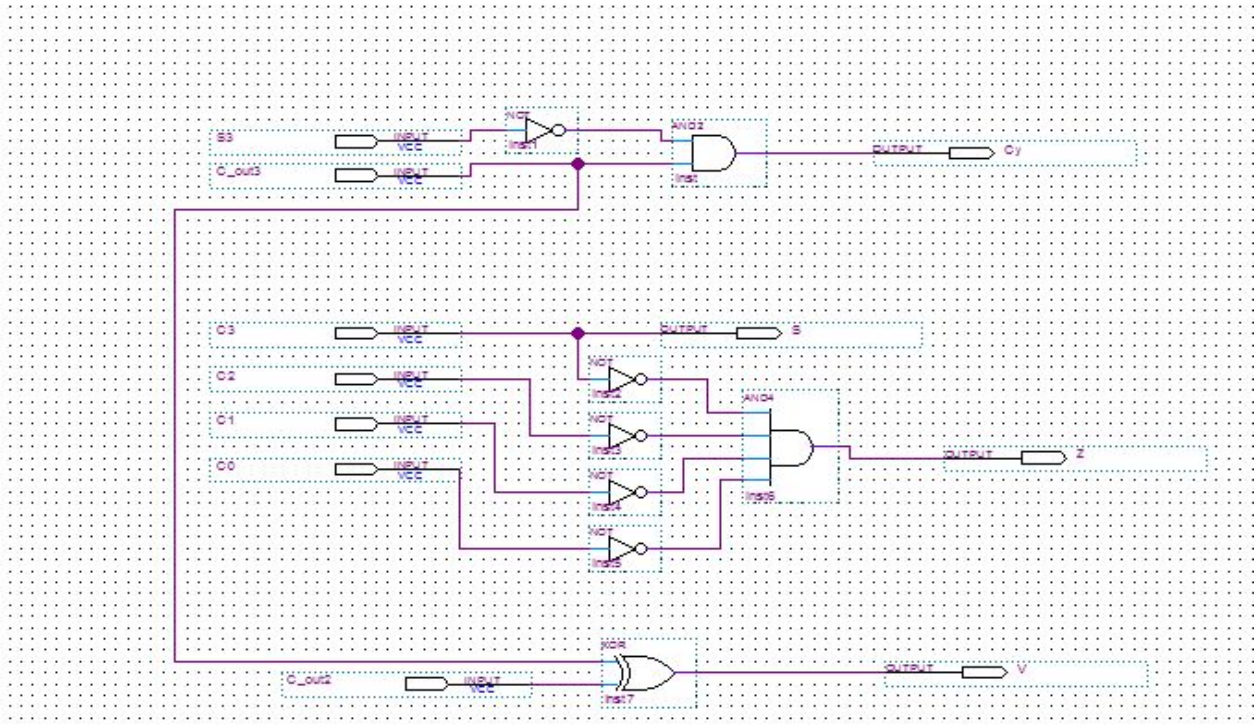
5. Circuit d'état

Le circuit d'état nous permet de connaître le signe, si la sortie est '' 0000 '', le débordement et la retenue.
La sortie S est mise à 1 si le bit le plus significatif C3 est 1, donc S = C3
La sortie Cy est mise à 1 si l'opération est arithmétique et si il y a une retenue,
(C_out3)(S3') = Cy
La sortie Z est mise à 1 si la sortie de l'ALU contient que des 0 donc, C3'C2'C1'C0' = Z
La sortie V est mise à 1 si il y a débordement, on évalue cela avec les deux dernier bit les plus significatifs, donc, C_out3 XOR C_out2 = V

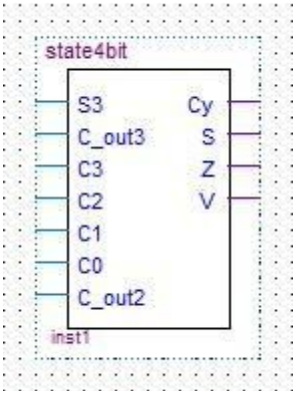
Après avoir créé le circuit nous avons fait la compilation du diagramme puis créé un symbole pour pourvoir l'utiliser pour la suite de l'expérience

Flow Summary	
Flow Status	Successful - Sun Nov 15 14:54:27 2015
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	lab3
Top-level Entity Name	state4bit
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final
Total logic elements	3 / 114,480 (< 1 %)
Total combinational functions	3 / 114,480 (< 1 %)
Dedicated logic registers	0 / 114,480 (0 %)
Total registers	0
Total pins	11 / 529 (2 %)
Total virtual pins	0
Total memory bits	0 / 3,981,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 532 (0 %)
Total PLLs	0 / 4 (0 %)

Diagramme du circuit d'état :



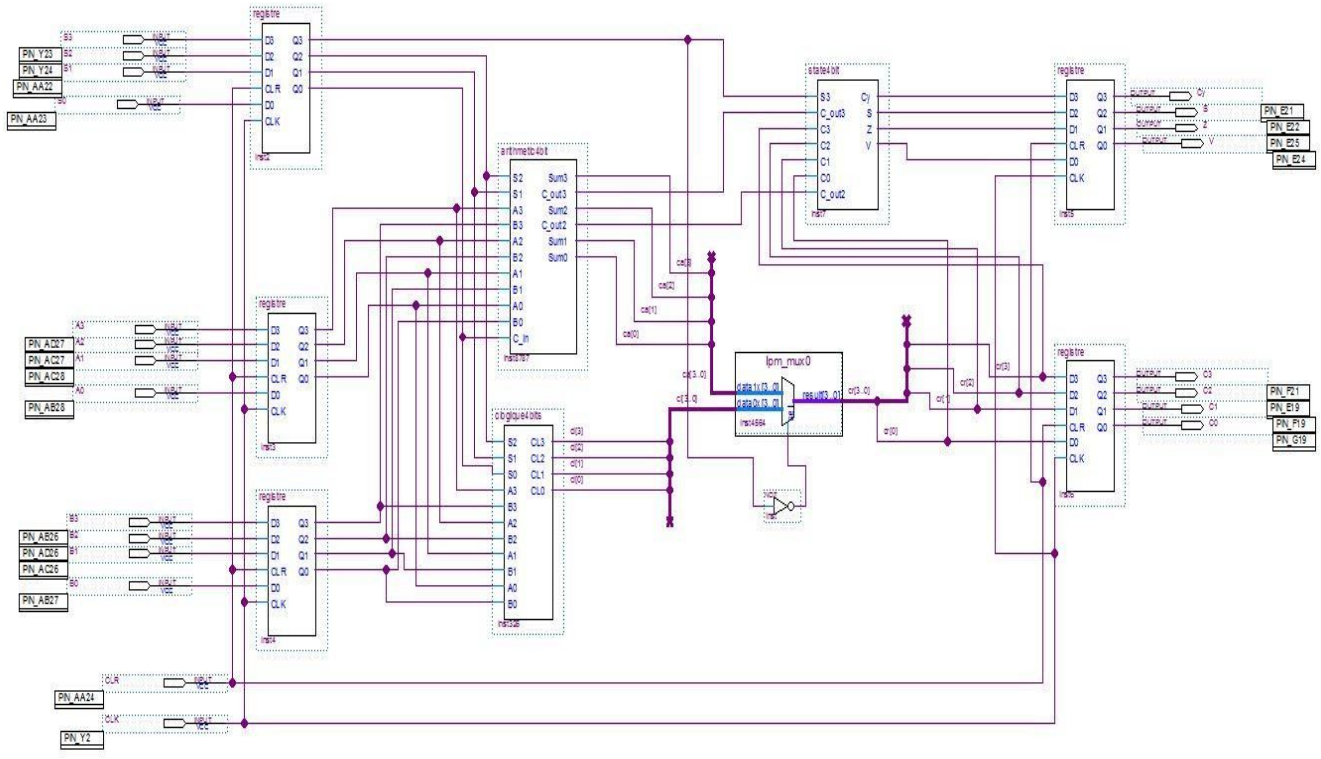
Voici le symbole créé à partir de ce circuit d'état, nous l'utiliserons plus tard dans la conception de l'ALU :



6. Unité Arithmétique et Logique

La dernière étape de la conception de l'ALU est d'assembler les circuits créés précédemment. De la droite à la gauche, tout d'abord il y a 3 registres, pour les sélecteurs, opérandes A et opérandes B. Ensuite les deux circuits principaux : arithmétique, logique et décalage. Pour le multiplexeur qui choisit entre les deux circuits arithmétique et logique nous avons utilisé le **lpm_mux**, qui simplifie la manipulation des nœuds car ce multiplexeur à comme entré et sorties des bus ce qui permet de connecter les nœuds directement. Ce qui suit est le registre d'état composé du circuit d'état et d'un registre. Et après le multiplexeur pour les valeurs du résultat final.

Diagramme de l'ALU :



Compilation de l'ALU :

Flow Summary	
Flow Status	Successful - Sun Nov 15 14:45:34 2015
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	lab3
Top-level Entity Name	ALU
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final
Total logic elements	47 / 114,480 (< 1 %)
Total combinational functions	46 / 114,480 (< 1 %)
Dedicated logic registers	20 / 114,480 (< 1 %)
Total registers	20
Total pins	22 / 529 (4 %)
Total virtual pins	0
Total memory bits	0 / 3,981,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 532 (0 %)
Total PLLs	0 / 4 (0 %)

Réalisation réelle

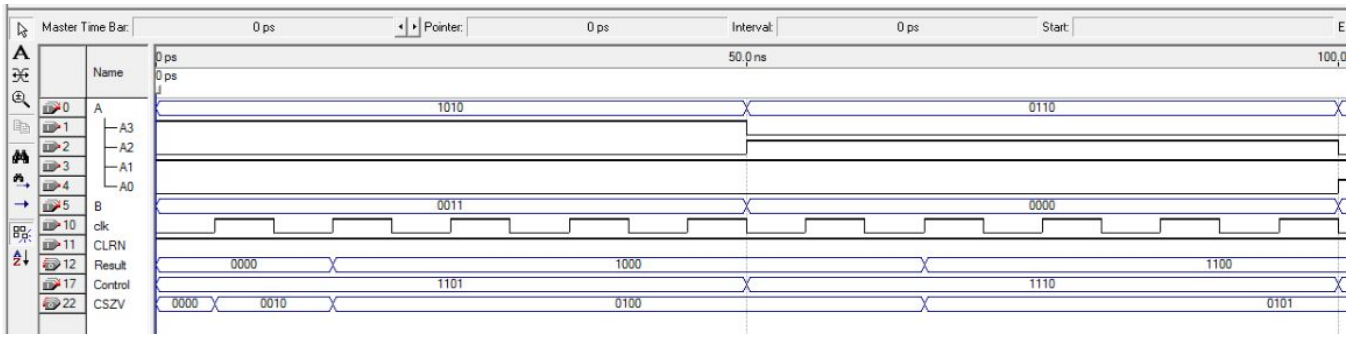
1. Résultats montrés de simulation/synthèse

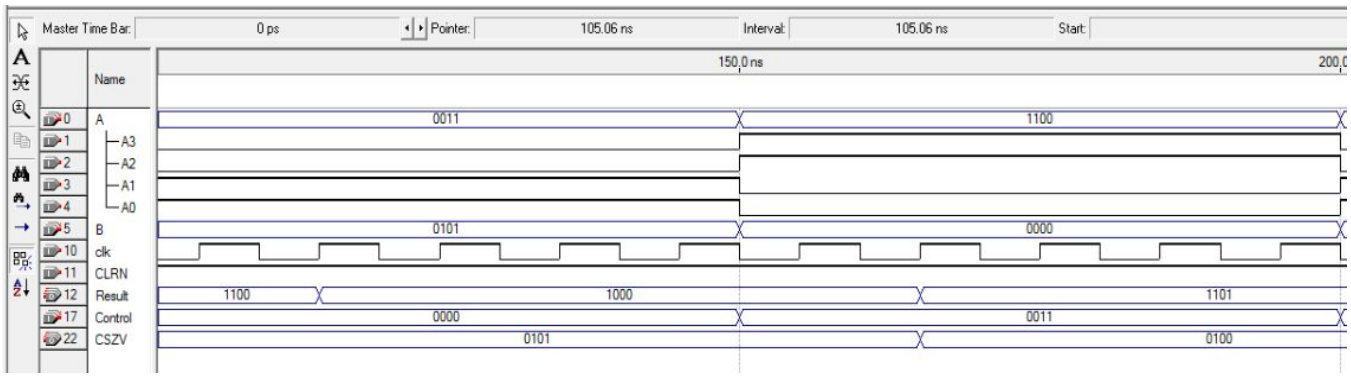
Clock cycle	RTL MICRO-OPERATIONS	S3	S2	S1	S0	A←Op1	B←0p2	C	V,Z,N, Cy	S_{16}	A_{16}	B_{16}	C_{16}	St_{16}
1	A←"1010", B←"0011" ,$C \leftarrow A \wedge B'$	1	1	0	1	1101	0011	1000	1010	D	A	3	8	A
2	A←"0110", $C \leftarrow ashl\ A$	1	1	1	0	0110	XXXX	1100	0010	E	6	X	C	2
3	A←"0011", B←"0101", $C \leftarrow A+B$	0	0	0	0	0011	0101	1000	1010	0	3	5	8	A
4	A←"1100", $C \leftarrow A+1$	0	0	1	1	1100	XXXX	1101	0010	3	C	X	D	2
5	A←"0011", B←"0101", $C \leftarrow A \oplus B$	1	1	0	0	0011	0101	0110	0000	C	3	5	6	0
6	A←"0011", $C \leftarrow A'$	0	1	1	0	1010	XXXX	0101	0000	6	A	X	5	0

7	$C \leftarrow$"0000"	1	0	0	0	XXXX	XXXX	0000	0100	B	X	X	0	4
8	$A \leftarrow$"0101", $B \leftarrow$"0011", $C \leftarrow A+B'+1$	0	1	0	1	0101	0011	0010	0001	J	5	3	2	1
9	$A \leftarrow$"1110", $C \leftarrow A$	0	0	1	0	1110	XXXX	1110	0010	2	E	X	E	2
10	$A \leftarrow$"0110", $C \leftarrow A'+1$	0	1	1	1	0110	XXXX	1010	0010	1	6	X	A	2
11	$A \leftarrow$"0101", $B \leftarrow$"0011", $C \leftarrow A \wedge B$	1	0	1	0	0101	0011	0001	0000	A	5	3	1	0
12	$A \leftarrow$"0001", $B \leftarrow$"0010", $C \leftarrow A+B+1$	0	0	0	1	0001	0010	0100	0000	1	1	2	4	0
13	$A \leftarrow$"1101", $C \leftarrow$ ashr A	1	1	1	1	1101	XXXX	1110	0010	F	D	X	E	2
14	$A \leftarrow$"0110", $B \leftarrow$"0101", $C \leftarrow A + B'$	0	1	0	0	0110	0101	0000	0101	4	6	5	0	5
15	$C \leftarrow$"1111"	1	0	0	1	XXXX	XXXX	1111	0010	9	X	X	F	2
16	$A \leftarrow$"1100", $B \leftarrow$"1010", $C \leftarrow A \vee B$	1	0	1	1	1100	1010	1110	0010	B	C	A	E	2

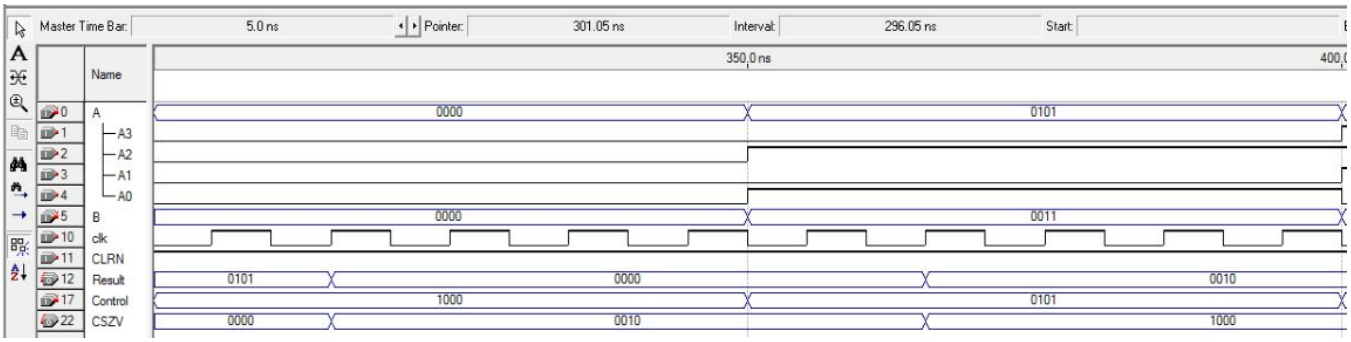
2. Vérification (simulation réelle & démonstration au professeur)

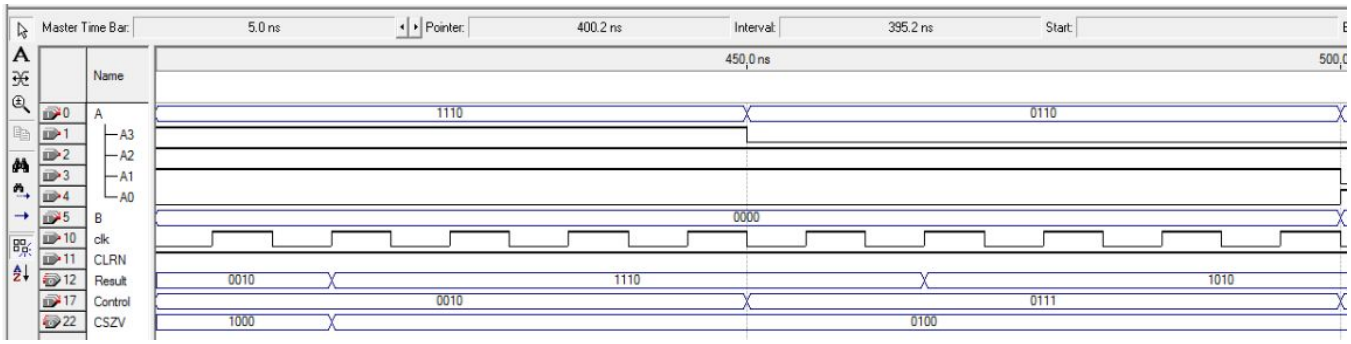
À l'issu des trois séances de laboratoire prévues, nous sommes enfin parvenus à la réalisation de l'ALU commençant par ses différentes unités. Nous n'avons manqué de simuler à chaque étape de la réalisation et pour finir nos résultats théoriques ont été vérifiés et validés par le chargé de laboratoire qui les a déclaré identiques aux résultats pratiques.

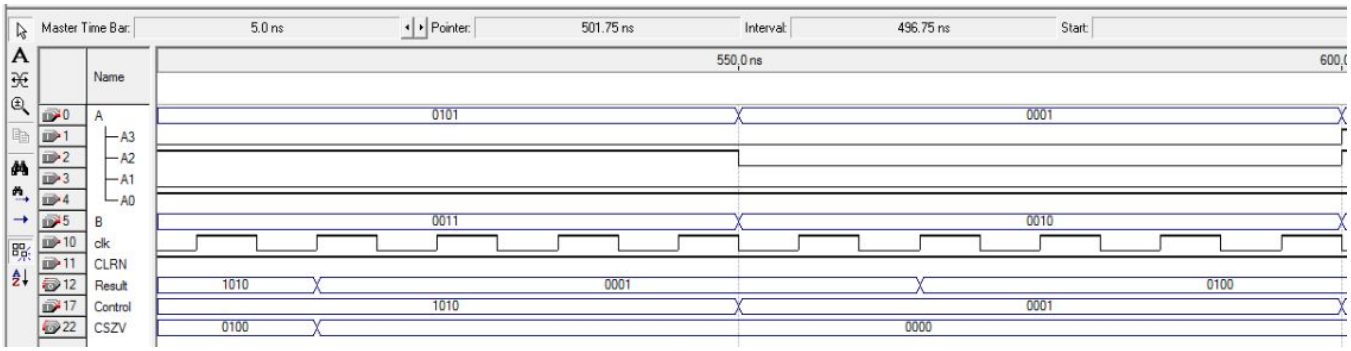


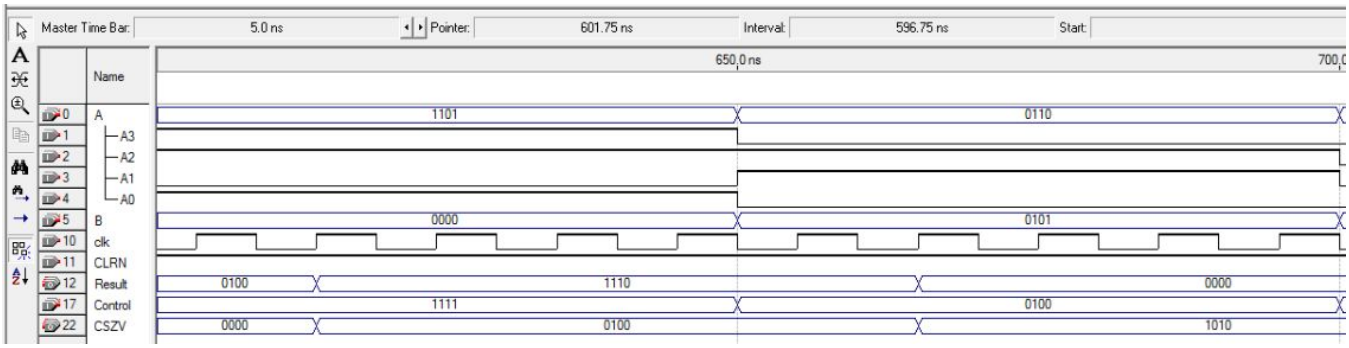

Master Time Bar: 0 ps Pointer: 105.06 ns Interval: 105.06 ns Start:
150,0 ns
Name
A
A3
A2
A1
A0
B
clk
CLRN
Result
Control
CSZV
0011
1100
0101
0000
1100
1000
1101
0000
0011
0101
0100

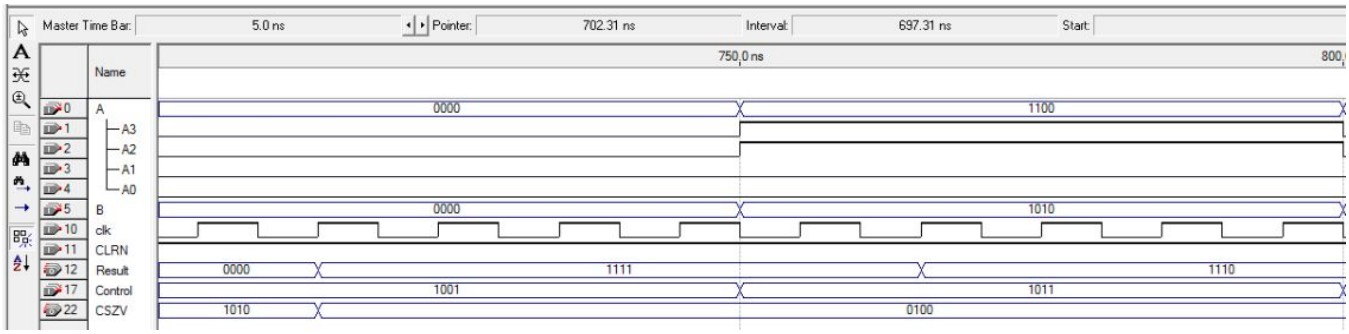

Master Time Bar: 5.0 ns Pointer: 200.15 ns Interval: 195.15 ns Start:
250,0 ns
Name
A
A3
A2
A1
A0
B
clk
CLRN
Result
Control
CSZV
0011
1010
0101
0000
1101
0110
0101
1100
0110
0100
0000

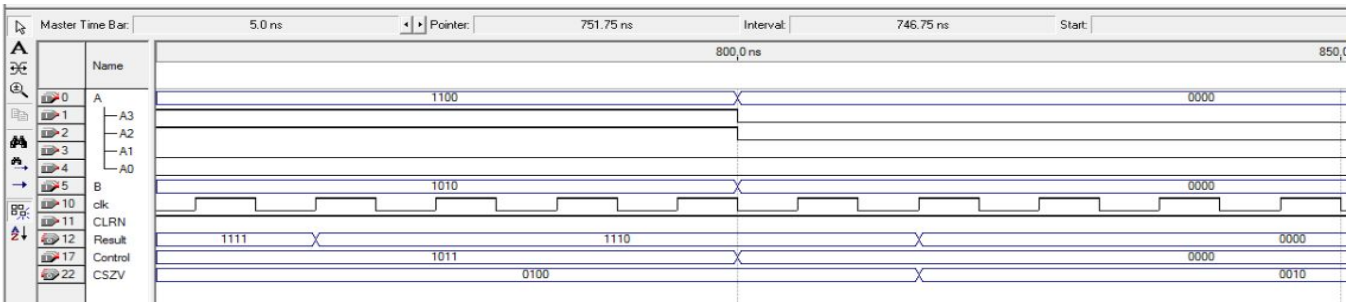

Master Time Bar: 5.0 ns Pointer: 301.05 ns Interval: 296.05 ns Start:
350,0 ns
Name
A
A3
A2
A1
A0
B
clk
CLRN
Result
Control
CSZV
0000
0101
0000
0011
0101
0000
0010
1000
0101
0000
0010
1000

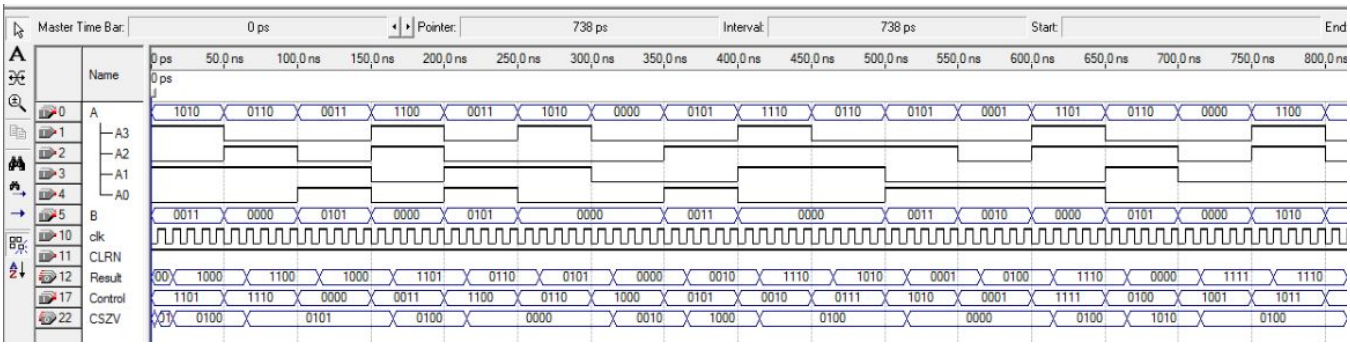

Master Time Bar: 5.0 ns Pointer: 400.2 ns Interval: 395.2 ns Start:
450,0 ns
Name
A
A3
A2
A1
A0
B
clk
CLRN
Result
Control
CSZV
1110
0110
0000
0010
1110
1010
0010
0111
1000
0100

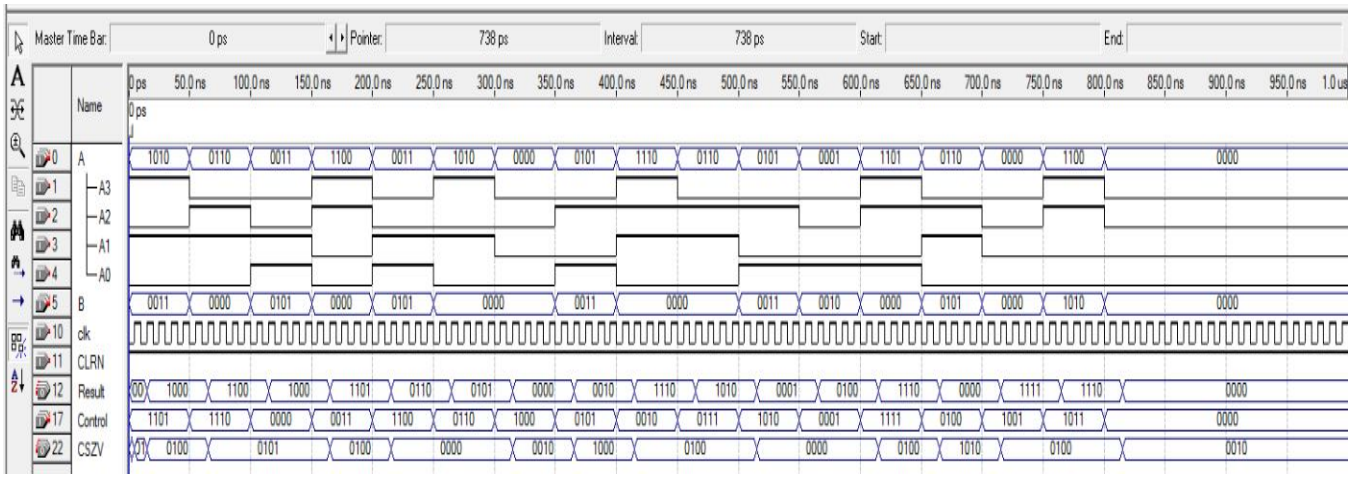

Master Time Bar: 5.0 ns
Pointer: 501.75 ns
Interval: 496.75 ns
Start:
550,0 ns
Name
A
A3
A2
A1
A0
B
clk
CLRN
Result
Control
CSZV
0101
0001
0011
0010
1010
0001
0100
1010
0001
0100
0000


Master Time Bar: 5.0 ns
Pointer: 601.75 ns
Interval: 596.75 ns
Start:
650,0 ns
Name
A
A3
A2
A1
A0
B
clk
CLRN
Result
Control
CSZV
1101
0110
0000
0101
0100
1110
0000
1111
0100
0000
0100
1010


Master Time Bar: 5.0 ns
Pointer: 702.31 ns
Interval: 697.31 ns
Start:
750,0 ns
Name
A
A3
A2
A1
A0
B
clk
CLRN
Result
Control
CSZV
0000
1100
0000
1010
0000
1111
1110
1001
1011
1010
0100


Master Time Bar: 5.0 ns
Pointer: 751.75 ns
Interval: 746.75 ns
Start:
800,0 ns
Name
A
A3
A2
A1
A0
B
clk
CLRN
Result
Control
CSZV
1100
0000
1010
0000
1111
1110
0000
1011
0000
0100
0010





Discussion

Cette expérience de laboratoire était axée sur la conception des circuits arithmétiques et logiques.

Pour concevoir notre ALU, nous avons réalisé les circuits suivants :

- Registre d'état conçu avec 4 bascules D qui changent d'état sur front montant.
- Additionneur complet 1 bit implémenté avec 2 XOR, 2 AND2 et 1 OR2.
- À l'aide l'additionneur complet 1 bit nous avons réalisé un additionneur complet 4 bits en faisant une combinaison de quatre additionneur complets 1 bit sans oublier les retenus du bi le plus faible au plus fort.
- Circuit logique et de décalage 1 bit réalisé avec 2 multiplexeurs et d'un additionneur complet.
- Circuit logique et de décalage 4 bits conçu avec 4 blocs circuits logique et de décalage 1 bit.
- Circuit d'état 4 bits comportant de 1 AND4, 1 XOR2, 5 NOT, et 1 AND2.

- Circuit ALU 4 bits avec circuit d'état 4 bits fait à partir du bloc circuit arithmétique, bloc circuit logique et de décalage complet, d'un MUX et du bloc circuit d'état 4 bits.
- Circuit arithmétique et logique complet réalisé avec le bloc circuit ALU 4 bits avec circuit d'état 4 bits connecté à 5 registres d'états 4 bits (A, B, S, F et le cinquième pour les états cszv). Les registres d'état avaient des connexions clear pour la remise à 0 et clock pour le signal d'horloge, tous deux agissant de manière instantanée.

Au cours de cette conception, nous avons créé des blocs pour chaque circuit constituant l'ALU afin de faciliter l'implémentation. L'état des sorties était observé à partir des Diodes Electroluminescentes de la carte Altera.

Au cours de l'expérience, les résultats observés sur la séquence des micro-opérations à simuler, à partir du logiciel QUARTUS II ainsi que des interrupteurs à bascule et des témoins lumineux de la carte Altera UP-2, ont permis de confirmer que les sorties observées correspondent à celles de la table des résultats théorique des micro-opérations du laboratoire. Autrement dit, les résultats théoriques et expérimentaux concordaient parfaitement. Les simulations concrètes en sont la preuve car elles reflètent la réalité. Tout cela montre que notre expérience s'est bien déroulée.

Il faut aussi rappeler que nous avions observé une inversion de résultats entre deux opérations (15 et 12) justement parce que lors de la conception de nos circuits nous avions inversé broches. Le TA nous a en effet aidé à retrouver l'erreur que nous avons corrigé pour finaliser le travail et simuler.

Conclusion

Apres cette expérience, nous pouvons conclure que le circuit arithmétique et logique comprend plusieurs étapes de réalisation indépendantes dont les circuits finissent toutes connectées, pour donner l'ALU. Ce laboratoire nous a permis de nous familiariser avec la conversion de critères fonctionnels en circuits logiques, et avec l'implémentation de ces circuits en utilisant la carte de Altera. Les différéntes manipulations que nous avons effectué nous ont permis d'atteindre les objectifs fixés au début du laboratoire.

Enfin, Grâce aux différentes manipulations que nous avons effectuées, il a été possible de se familiariser avec la conversion de critères fonctionnels en circuits logiques, et avec l'implémentation de ces circuits en utilisant l'appareil de logique programmable EP4CE115F29C7, qui se trouve sur la plaque Altera DE2-115 de Quartus II 13.0. Aussi, nous avons pu réaliser des circuits logiques séquentiels et tester leur fonctionnement. Les résultats expérimentaux étaient identiques aux résultats théoriques ce qui nous a permis de mettre en évidence leur véracité.