

Université d'Ottawa
Faculté de génie

École d'ingénierie et de
technologie de l'information



uOttawa

L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Information
Technology and Engineering

CEG2536 : ARCHITECTURE DES ORDINATEURS I

Laboratoire 3 : Conception d'une unité arithmétique et logique

Notes

- Une pénalité de 10 points sera appliquée pour chaque jour de retard. Une heure de retard sera considéré comme un jour!
- Tous les étudiants doivent être au courant du règlement de l'université concernant la fraude scolaire, et doivent être conscients des sanctions applicables en cas de fraude :
http://www.uottawa.ca/academic/info/regist/fraud_f.html
<http://www.uottawa.ca/plagiat.pdf>
<http://www.sass.uottawa.ca/redaction/plagiarism-f.pdf>

Objectifs

Dans ce laboratoire, les étudiants devront concevoir et construire une unité arithmétique et logique ALU. Elle doit effectuer 16 opérations différentes sur deux entrées de 4 bits, produisant une sortie unique de 4 bits ainsi qu'un indicateur de l'état de la sortie (débordement, signe, zéro, retenue de sortie). Le circuit résultant doit fonctionner en simulation et dans la carte de développement UP2 d'Altera (en utilisant des interrupteurs DIP pour gérer les entrées, et des DELs pour observer les sorties).

Equipement

- Système Quartus II
- Carte UP2 d'Altera avec:
 - Câble Byte blaster
 - EMP7128S CPLD
 - Générateur de tension 7 VDC, 250 mA
- Outils : fils de gage 22

Références

- Notes de cours.
- Chapitres 1, 2, 3 et 4 du livre: *Computer Systems Architecture*, Morris Mano, 3rd edition, 1993, ISBN 0-13-175563-3.
- Manuel d'utilisation de l'environnement de développement d'Altera UP2 (<http://www.altera.com/literature/univ/upds.pdf>). Cette carte est identique aux cartes disponibles dans notre lab, excepté la partie FLEX (une composante qui n'est pas utilisée dans ce lab).

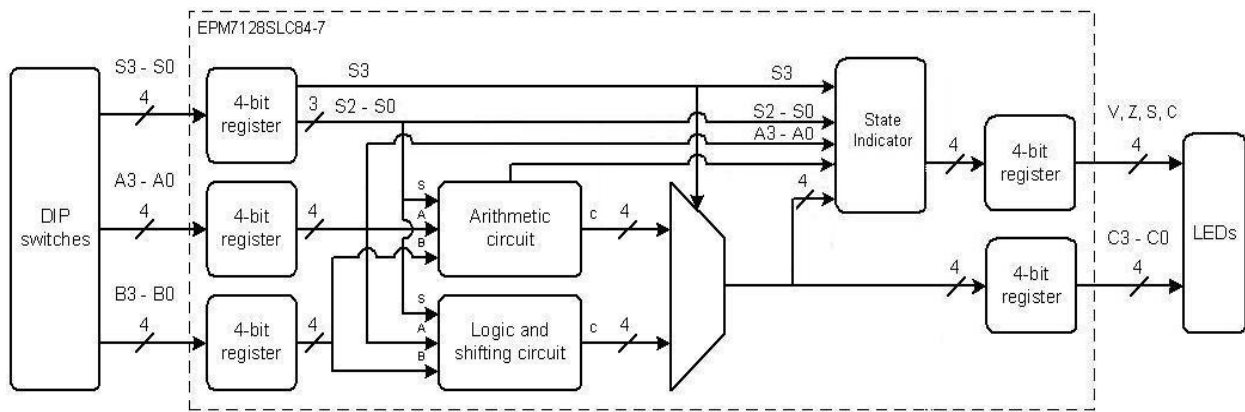


Figure 1: Vue d'ensemble du dispositif expérimental

Pré-lab

Dans ce lab, vous allez concevoir une ALU qui inclut non seulement un circuit arithmétique et logique, mais aussi un circuit de décalage, comme montré sur la figure 1. L'unité aura deux entrées de 4 bits A et B, une sortie de 4 bits C, et 4 sorties d'état V, Z, S, et C. La Table 1 montre les 16 micro-opérations que l'ALU sera capable d'effectuer. Le signal de contrôle S3 sera utilisé pour choisir la sortie: soit le résultat du circuit arithmétique, soit le résultat du circuit logique. Les trois autres signaux de contrôle – S2, S1, et S0 – serviront à choisir les opérations de chaque circuit. Toutes les entrées et sorties doivent passer par des registres (des bascules D toutes déclenchées à la même impulsion d'horloge). C'est une pratique courante et raisonnable, car alors les entrées et les sorties sont stables pendant la période d'horloge entière.

Table 1: Micro-opérations arithmétiques et logiques

Circuit	S3	S2	S1	S0	Sortie de l'ALU	Description
Circuit Arithmétique	0	0	0	0	$C \leftarrow A + B$	Addition
	0	0	0	1	$C \leftarrow A + B + 1$	Addition avec retenue
	0	0	1	0	$C \leftarrow A$	Transfert vers A
	0	0	1	1	$C \leftarrow A + 1$	Incréméntation vers A
	0	1	0	0	$C \leftarrow A + \bar{B}$	Soustraction $A - B$ avec emprunt (prendre le complément à 1 de B). Ceci donne un résultat inférieur de 1 à une soustraction conventionnelle.
	0	1	0	1	$C \leftarrow A + \bar{B} + 1$	Soustraction $A - B$ (en prenant le complément à 2 de B)
	0	1	1	0	$C \leftarrow \bar{A}$	NON A (complément à 1 de A)
	0	1	1	1	$C \leftarrow \bar{A} + 1$	Complément à 2 de A
Circuit de logique et de décalage	1	0	0	0	$C \leftarrow "0000"$	Remise à 0
	1	0	0	1	$C \leftarrow "1111"$	Mise à 1
	1	0	1	0	$C \leftarrow A \wedge B$	A ET B
	1	0	1	1	$C \leftarrow A \vee B$	A OU B
	1	1	0	0	$C \leftarrow A \oplus B$	A OU-EXCLUSIF B
	1	1	0	1	$C \leftarrow A \wedge \bar{B}$	Remise à 0 sélective
	1	1	1	0	$C \leftarrow ash\ l\ A$	Décaler A à gauche (multiplication signée par 2)
	1	1	1	1	$C \leftarrow ash\ r\ A$	Décaler A à droite (division signée par 2)

Conception hiérarchique

Puisque les signaux d'entrées et de sorties de l'ALU ont tous une dimension de 4 bits, la façon la plus efficace d'implémenter l'ALU est de le concevoir de façon hiérarchique. Les fichiers sur la couche la plus basse seront ceux d'un additionneur complet de 1 bit, et d'un circuit logique de 1 bit. Les fichiers intermédiaires consisteront d'un registre de 4 bits, d'un circuit arithmétique de 4 bits, un circuit logique de 4 bits, et un circuit d'état. Finalement, le fichier au niveau le plus élevé couvrira le circuit complet, chargeable dans la carte UP2. La relation entre ces fichiers est montrée à la Figure 2. Les parties en jaune seront expliquées plus loin dans ce document. Pour la préparation de ce lab, il faut concevoir et tester les circuits des parties vertes: le circuit logique de 1 bit, le circuit logique de 4 bits, le circuit arithmétique de 4 bits, et le circuit d'état.

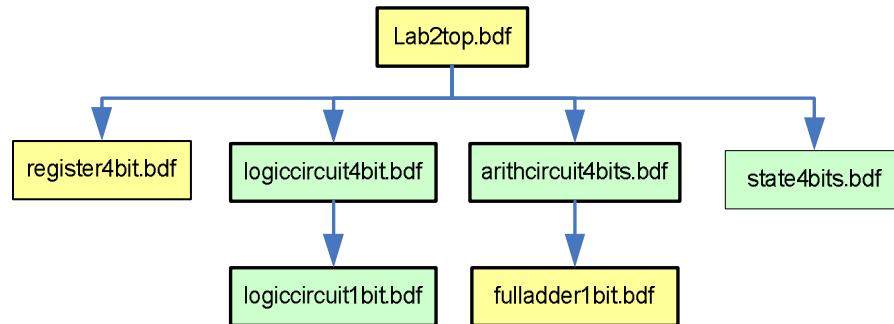


Figure 2: Hiérarchie des fichiers

Conception du circuit de logique et de décalage de 1-bit

Au niveau le plus bas, le circuit de logique et de décalage opère bit par bit (par exemple, sur les bits A2 et B2). Concevez un circuit qui implémente toutes les fonctions de logique et de décalage montrées au bas de la Table 1. Le circuit doit avoir 3 entrées de contrôle (S2, S1, et S0), deux entrées logiques (A_i et B_i), deux entrées additionnelles pour les décalages à gauche et à droite (A_{i-1} et A_{i+1}), et une sortie (C_i). Vous pouvez vous inspirer de la Figure 11 qui représente un exemple de circuit logique de 1-bit, et de la figure 9 pour un exemple sur la façon d'implémenter un circuit de décalage en utilisant un multiplexeur. Pour ce lab, A3 et B3 seront les bits les plus significatifs, et A0 et B0 seront les bits les moins significatifs (on trouve parfois des notations opposées dans certains livres). Vous pouvez utiliser le multiplexeur 74151, dont le symbole et la table de vérité se trouvent ci-dessous. Notez que dans la table, A, B, et C sont les lignes de sélection du multiplexeur, et ne correspondent donc pas aux A, B, et C de notre ALU!

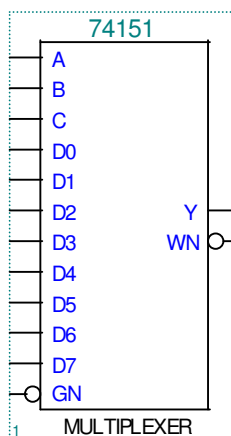


Figure 3: Symbole du multiplexeur 74151

Table 2: Table de vérité du multiplexeur 74151

Entrées				Sortie
Sélection			Enable	
C	B	A	GN	Y
X	X	X	1	0
0	0	0	0	D0
0	0	1	0	D1
0	1	0	0	D2
0	1	1	0	D3
1	0	0	0	D4
1	0	1	0	D5
1	1	0	0	D6
1	1	1	0	D7

Conception du circuit de logique et de décalage de 4-bit

Déterminez comment connecter 4 de vos circuits de décalage et de logique de 1 bit pour implémenter un circuit de 4 bits. La conception finale doit avoir 3 entrées de contrôle (S2 – S0), 8 entrées de données (A3 – A0 et B3 – B0), et 4 sorties de données (C3 – C0).

Conception du circuit arithmétique de 4-bit

Le circuit arithmétique de 4 bits sera basé sur l'additionneur complet montré sur la Figure 7. Déterminez comment connecter 4 de ces additionneurs pour implémenter les fonctions arithmétiques de la Table 1. Votre circuit final devra avoir 3 entrées de contrôle (S2 – S0), 8 entrées de données (A3 – A0 et B3 – B0), 4 sorties de données (C3 – C0) ainsi que deux sorties de 1 bit : les deux dernières retenues, qui seront utilisées pour le circuit d'état. Voici maintenant quelques indices pour simplifier cette tâche!

- La Figure 10 et la Table 7 sont de bons exemples desquels il serait sage de s'inspirer
- Utilisez le bit de contrôle S0 comme retenue d'entrée
- Notez que la table de vérité de l'ALU ne vous permettra pas de placer les entrées A3 – A0 directement sur les additionneurs complets, et donc il vous faudra un peu de logique supplémentaire. Commencez votre conception en analysant quelles entrées doivent apparaître aux entrées des additionneurs complets pour chaque combinaison de S2 et de S1.
- Le symbole et la table de vérité du multiplexeur dual 4-à-1 74153 sont montrés ci-dessous. Notez que les deux multiplexeurs partagent les mêmes lignes de sélection A et B (là encore, elles n'ont aucun rapport avec les entrées de l'ALU qui ont le même nom!)

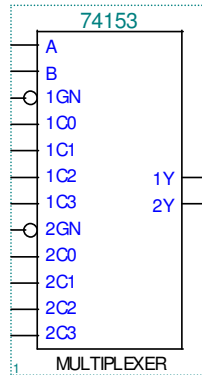


Figure 4: Symbole du multiplexeur dual 74153

Table 3: Table de vérité du multiplexeur dual 74153

Entrées			Sortie
Sélection B	A	Enable GN	
X	X	1	0
0	0	0	C0
0	1	0	C1
1	0	0	C2
1	1	0	C3

- La table de vérité et le symbole du quadruple multiplexeur 2-à-1 74257 sont montrés ci-dessous:

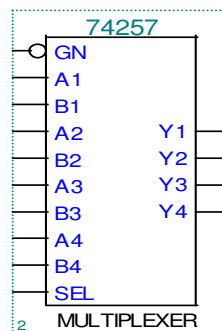


Figure 5: Symbole du quadruple multiplexeur 74257

Table 4: Table de vérité du quadruple multiplexeur 74257

Entrées		Sortie
Sélection SEL	Enable GN	
X	1	Z (haute impédance)
0	0	A
1	0	B

Notez que l'utilisation des multiplexeurs 74151, 74153 et 74257 n'est pas obligatoire. En faite, vous pouvez obtenir **5% de points bonis** si vous décidez d'utiliser le composant ***lpm-mux*** aux lieux de ces multiplexeurs. Figure 12 exhibe un exemple de ce multiplexeur.

Conception du registre d'état

Pour faciliter la tâche de l'utilisateur, un ALU est souvent équipé d'un registre à usage spécifique de 4-bits appelé le **registre d'état**. Le registre d'état est composé de 4 bits d'état qui sont automatiquement mis à jour après chaque opération de l'ALU. Les 4 bits d'état sont symbolisés par C, S, Z, et V, et sont définis par:

- Le bit C (retenue de sortie « carry out ») est mis à 1 uniquement lorsque l'opération est une opération arithmétique et sa retenue de sortie est 1. Autrement, il est mis à 0.
- Le bit S (signe) est mis à la valeur du bit le plus significatif de la sortie de l'ALU.
- Le bit Z (zéro) est mis à 1 uniquement lorsque les bits de sortie de l'ALU sont tous 0. Il est mis à 0 sinon.
- Le bit V (débordement) est mis à 1 uniquement lorsqu'un débordement a eu lieu. Dans le cas de notre ALU, par exemple, la valeur de V doit être 0 lorsque $S3-S0=1100$ (A OU-EXCLUSIF B) car il est impossible que ce genre d'opération génère un débordement. Par contre, les opérations arithmétiques et celles de décalage arithmétique peuvent générer un débordement. Pour bien déterminer la valeur de V, on supposera que les opérations arithmétiques sont faites avec deux opérandes signés A et B.

Procédure

1. Utilisez Quartus II pour entrer le fichier graphique suivant, qui implémente un registre de 4 bits. Utilisez des bascules D (symbole « dff »). Sauvegardez le fichier sous le nom « register4bits.bdf » (des noms standard aideront vos TAs à vérifier votre circuit si besoin).

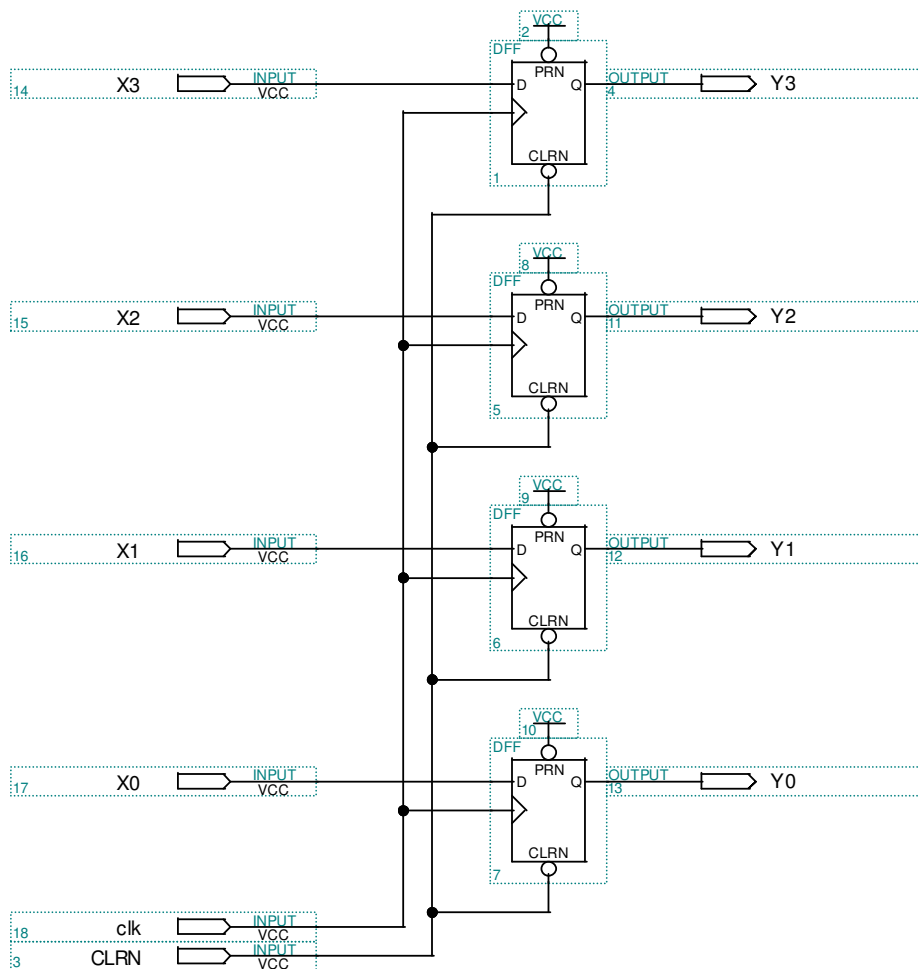


Figure 6: Schéma d'un registre de 4-bits

2. Choisissez **Set as Top Level Entity** (assignez le fichier ouvert au projet), sélectionnez EPM7128SLC84-7 comme appareil, et assurez-vous que le fichier compile sans erreurs (vous n'avez pas besoin d'assigner des pins à cette étape) Vérifiez dans le fichier « register4bits.fit.rpt » le pourcentage des ressources utilisées de la puce dans la section **Fitter Summary**. Créez un symbole pour le registre qui pourra être utilisé plus tard.

3. Entrez votre circuit de logique et de décalage de 1 bit dans un fichier que vous appellerez « logiccircuit1bit.bdf ». Pour entrer le symbole du multiplexeur 74151 dans votre schéma, faites un clic-droit, sélectionnez **Insert >> Symbol**, et tapez «74151» dans la boîte de dialogue **Name**. **Si vous avez décidé d'utiliser le multiplexeur lpm_mux (pour obtenir des points bonis), faites la même chose mais entrer «lpm_mux » dans la boîte de dialogue Name**. Si le multiplexeur n'apparaît pas sous la forme que vous préférez, vous pouvez faire un clic droit de souris dessus, et vous pouvez le renverser horizontalement ou verticalement. Assignez le fichier ouvert au projet, compilez le fichier, et créez un symbole.

4. Entrez votre circuit de logique et de décalage de 4 bits dans un fichier que vous appellerez « logiccircuit4bits.bdf ». Assignez le fichier ouvert au projet, compilez le fichier, et créez un symbole. Quel pourcentage des ressources de la puce utilisez-vous pour ce fichier ?

5. Entrez l'additionneur complet suivant dans un fichier que vous appellerez « fulladder1bit.bdf » Répétez les étapes requises jusqu'à la création d'un symbole.

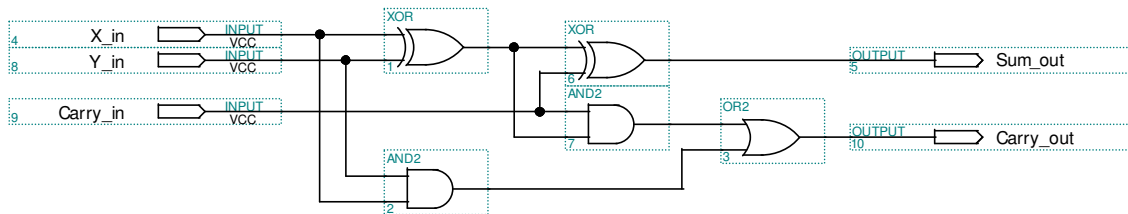


Figure 7: Schéma d'un additionneur complet

6. Entrez votre circuit arithmétique de 4 bits dans un fichier que vous appellerez « arithcircuit4bits.bdf ». Répétez les étapes requises jusqu'à la création d'un symbole. Quel pourcentage des ressources de la puce utilisez-vous pour ce fichier ? Le circuit arithmétique est-il plus complexe que le circuit de logique et de décalage ?

7. Entrez votre circuit d'état, que vous appellerez « state4bits.bdf ». Répétez les étapes requises jusqu'à la création d'un symbole.

8. Combinez vos registres, votre circuit arithmétique et logique avec un multiplexeur 74257 (**ou lpm_mux**) dans un fichier de haut niveau similaire à celui présenté sur la Figure 8 (voir à la dernière page de ce document). Il est tout à fait possible que votre diagramme diffère légèrement, suivant les conventions et noms que vous avez utilisés pour votre conception, et les symboles que vous avez créés.

9. Assignez le fichier ouvert au projet. Assignez la pin 83 à l'horloge, et la pin 1 à CLRn, mais laissez tous les autres signaux sans assignations. Compilez le fichier : le moteur de routage et de placement va assigner toutes les autres pins pour vous.

10. Créez un fichier de formes d'ondes (.vwf).

a. Choisissez **Edit >> Grid Size...** et entrez 50ns. Notez que votre circuit devrait fonctionner de façon satisfaisante avec une taille de grille de 20ns et une période d'horloge de 40ns, pourvu que vous l'ayez bien conçu. Cette valeur est très conservatrice, et même les circuits les plus inefficaces devraient pouvoir avoir assez de temps entre chaque impulsion d'horloge !

b. Choisissez **Edit >> End Time...** et entrez 2.0us.

c. Choisissez **Edit >> Insert Node or Bus...** afin d'importer toutes les entrées et toutes les sorties dans le fichier .vwf.


- d. Mettez en place les signaux d'horloge et CLRN (notez que CLRN est actif au niveau bas, de telle manière que l'ALU ne sortira pas de son état de repos sauf si CLRN est mis au niveau haut).
- e. Mettez en place le reste des signaux d'entrée afin d'implémenter la série d'instructions RTL contenues dans la Table 5. Il est très fortement recommandé de grouper les signaux pour simplifier l'entrée de données et la lecture. Pour grouper A3 – A0, en supposant que A3 est le MSB et A0 est le LSB, assurez-vous que A3 se trouve en haut, et A0 en bas. Choisissez A3, puis appuyez sur la touche shift et cliquez sur A0. Ensuite, faites un clic droit et choisissez **Group...**, donnez un nom à ce groupe, choisissez le format d'affichage en binaire, et sélectionnez **OK**. Pour entrer une valeur pour le groupe, utilisez la souris pour sélectionner un signal sur la plage de temps voulue, puis cliquez sur l'icône  à droite de l'écran. Une petite boîte va s'ouvrir et vous demander une nouvelle valeur.

Table 5: Séquence des micro-opérations à simuler

Cycle d'horloge	Micro-opérations RTL
1	$A \leftarrow "1010", B \leftarrow "0011", C \leftarrow A \wedge \bar{B}$
2	$A \leftarrow "0110", C \leftarrow \text{ashl } A$
3	$A \leftarrow "0011", B \leftarrow "0101", C \leftarrow A + B$
4	$A \leftarrow "1100", C \leftarrow A + 1$
5	$A \leftarrow "0011", B \leftarrow "0101", C \leftarrow A \oplus B$
6	$A \leftarrow "1010", C \leftarrow \bar{A}$
7	$C \leftarrow "0000"$
8	$A \leftarrow "0101", B \leftarrow "0011", C \leftarrow A + \bar{B} + 1$
9	$A \leftarrow "1110", C \leftarrow A$
10	$A \leftarrow "0110", C \leftarrow \bar{A} + 1$
11	$A \leftarrow "0101", B \leftarrow "0011", C \leftarrow A \wedge B$
12	$A \leftarrow "0001", B \leftarrow "0010", C \leftarrow A + B + 1$
13	$A \leftarrow "1101", C \leftarrow \text{ashr } A$
14	$A \leftarrow "0110", B \leftarrow "0101", C \leftarrow A + \bar{B}$
15	$C \leftarrow "1111"$
16	$A \leftarrow "1100", B \leftarrow "1010", C \leftarrow A \vee B$

11. Démarrez la simulation, et vérifiez que toutes les composantes de votre ALU fonctionnent correctement. **Notez qu'il y a un délai à la sortie à cause des registres d'entrées/sorties!** Si des éléments de votre circuit ne fonctionnent pas correctement, il est impératif que vous les corrigiez à cette étape, jusqu'à ce que tout fonctionne correctement. Faites finalement une capture d'écran de la simulation pour votre rapport de laboratoire.
12. Utilisez **Assignments >> Pins...** pour assigner le reste des pins comme indiqué sur la Table 6.

Table 6: Assignations de pins pour le circuit final

Signal	Numéro de pin
S3	48
S2	49
S1	50
S0	51
A3	28
A2	29
A1	30
A0	31
B3	33
B2	34
B1	35

B0	36
C3	70
C2	69
C1	68
C0	67
V	65
Z	71
S	73
C	74

13. Puisque les DELs s'allument lorsqu'on leur présente un niveau bas, il est préférable d'inverser les signaux de sortie. Cependant, les inverser corrompt la lecture de la simulation. Pour éviter cela, utilisez la commande « File → Save As... » pour sauvegarder une deuxième copie de votre fichier au niveau le plus haut. Pour inverser C3 – C0, faites un clic droit sur le registre de sortie de 4 bits, et sélectionnez **Propriétés**, ensuite choisissez le tab de **Ports**. Un par un, choisissez les signaux Y3 – Y0 et cliquez sur **All** dans la boîte **Inversion**. Faites la même chose pour les sorties du circuit d'état. Assignez le fichier ouvert au projet et compilez-le. Lisez le fichier « .rpt » pour vous assurer que vos assignations de pins se sont bien propagées dans le nouveau fichier.

14. Connectez les pins d'entrée aux interrupteurs DIP, et les pins de sortie aux DELs. Programmez le EPM7128 et vérifiez que votre ALU fonctionne (utilisez votre simulation pour vérifier les valeurs de sortie ; n'oubliez pas le délai sur les sorties de la simulation). Un interrupteur DIP enfoncé équivaut à une entrée de 0V. Relevé, l'interrupteur fournit 5V. À cause du fait que certaines cartes UP2 ont souffert d'une utilisation intensive, il serait prudent de tester le fonctionnement de chaque interrupteur avec un multimètre avant de tout connecter.

Remise du rapport

Ce laboratoire doit être remis deux semaines après la première session de votre lab. Les parties suivantes doivent être incluses dans votre rapport :

1. diagrammes/schémas de vos circuits;
2. une discussion brève sur la façon dont vous êtes arrivés à concevoir votre circuit ; et
3. une image de votre fichier de simulation (.vwf). Vous pouvez séparer le tout en plusieurs images si nécessaire.

Le TA doit vérifier que votre circuit fonctionne en simulation **ET** sur la carte UP2 **AVANT** que vous ne quittiez le laboratoire. Sinon, il sera considéré que vous n'avez pas fait de démonstration, et vous serez notés en conséquence.

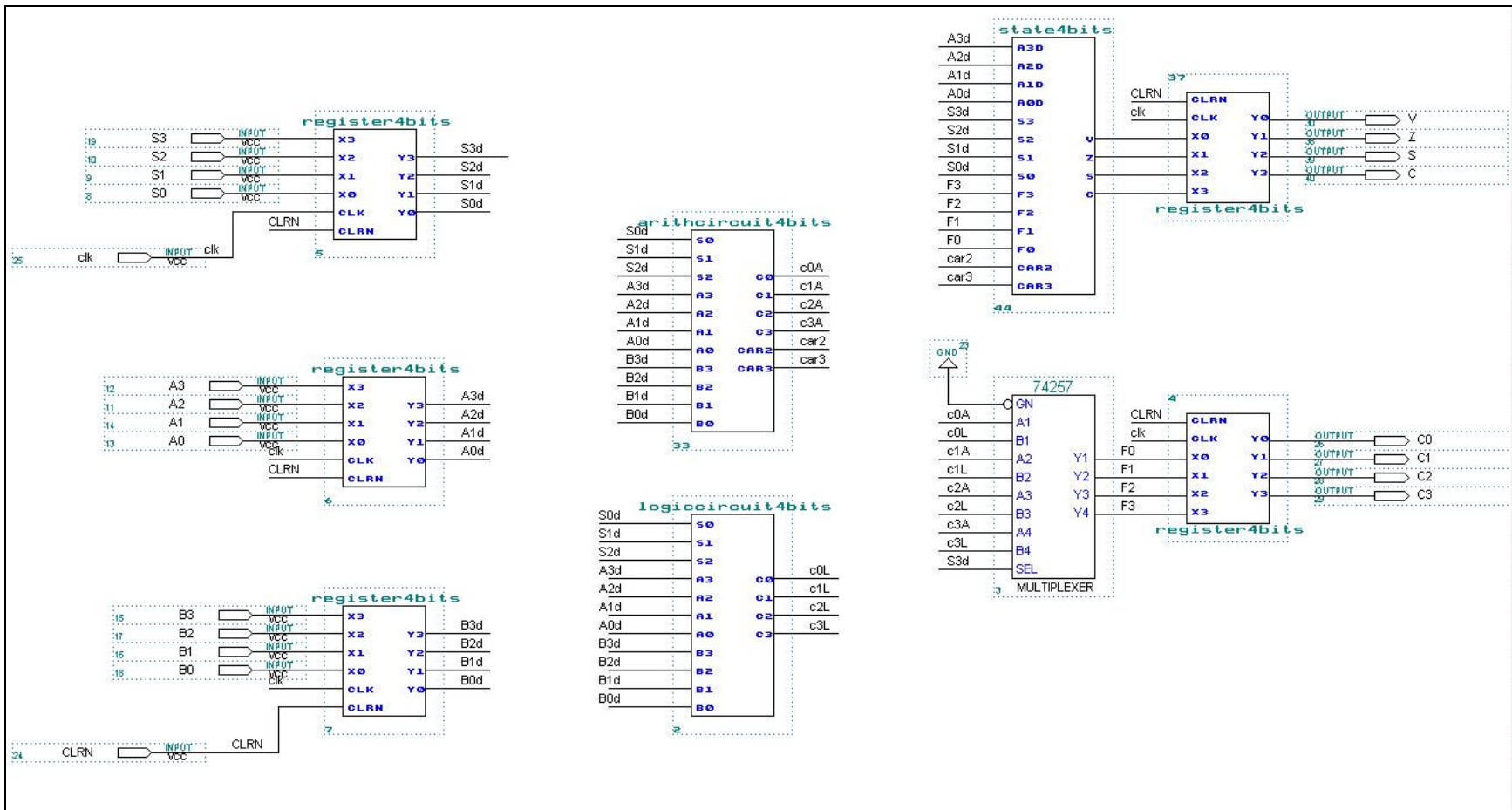


Figure 8: Schéma de l'ALU complet

Annexes

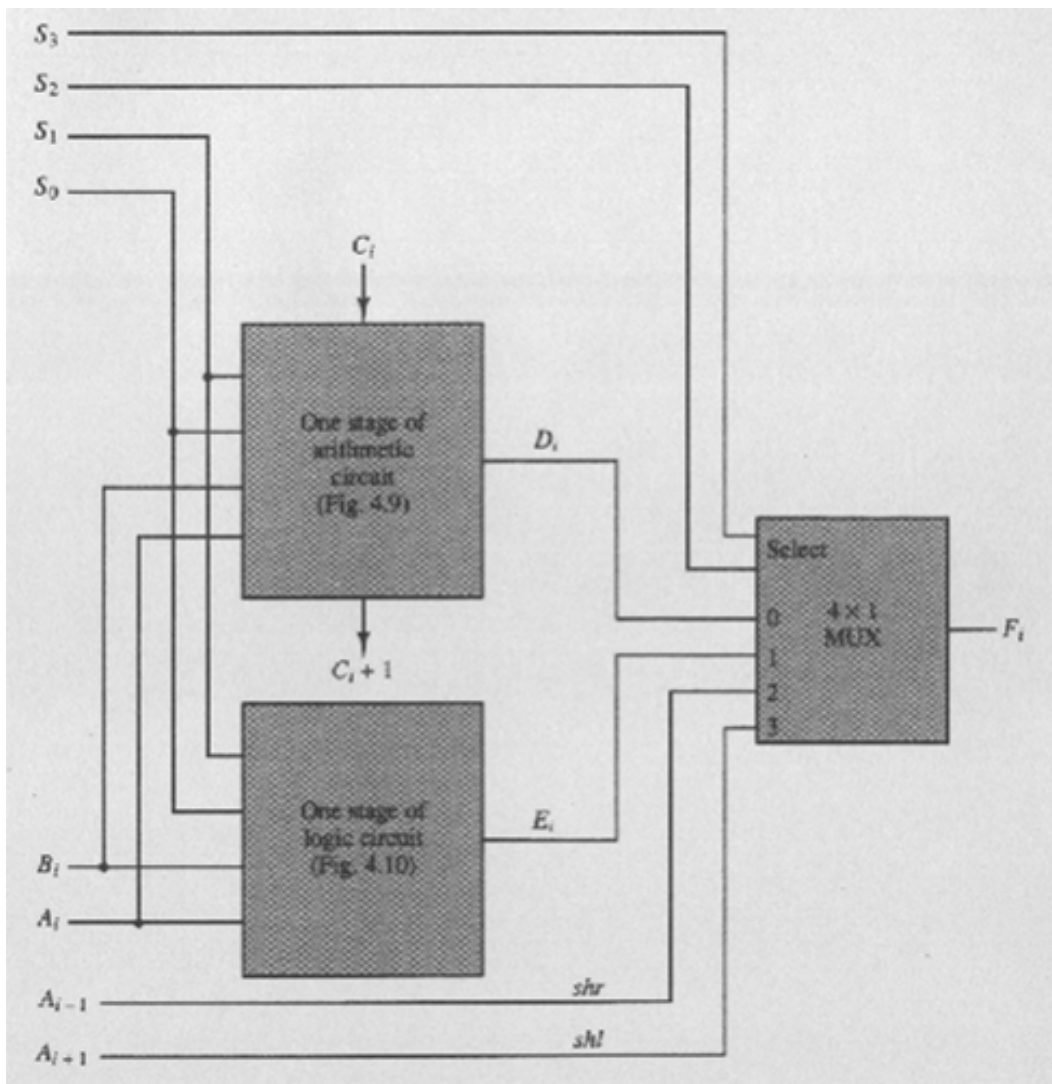


Figure 9: Unité arithmétique et logique à un étage

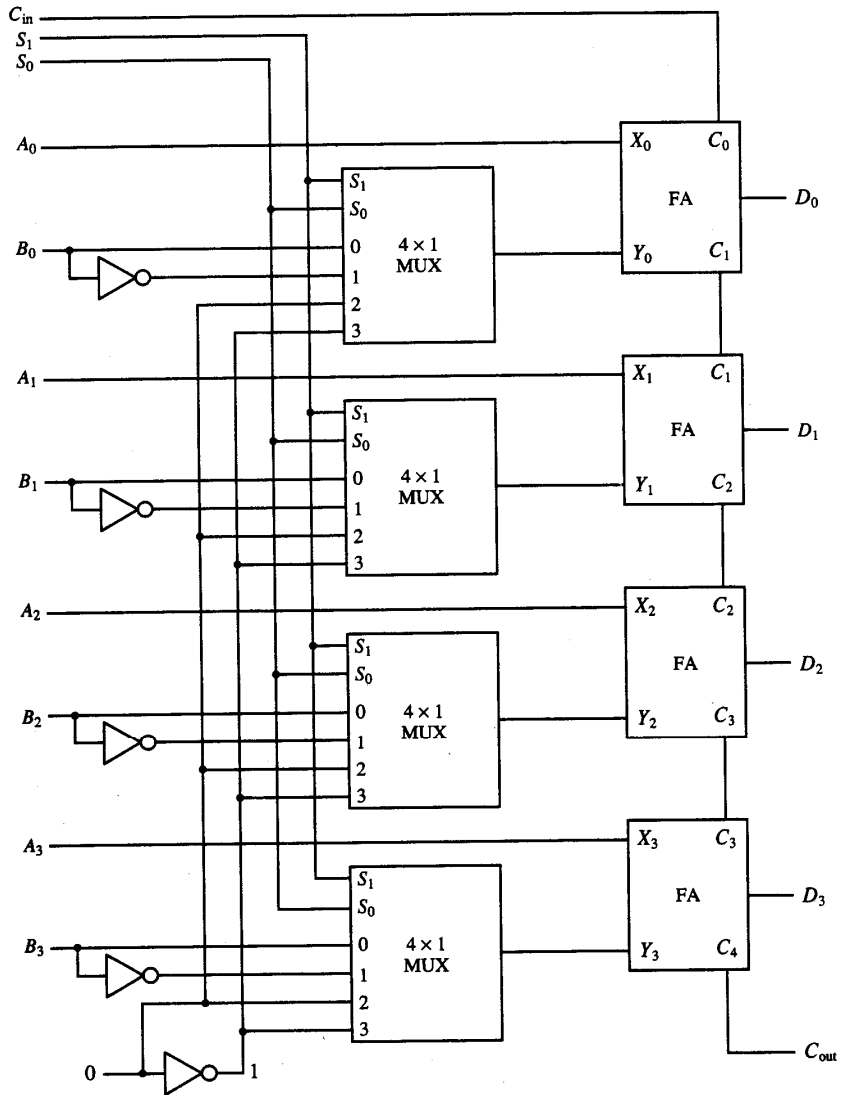


Figure 10: Circuit arithmétique de 4-bit

Table 7: Table de fonction d'une unité arithmétique et logique et de décalage

Operation select					Operation	Function
S_3	S_2	S_1	S_0	C_{in}		
0	0	0	0	0	$F = A$	Transfer A
0	0	0	0	1	$F = A + 1$	Increment A
0	0	0	1	0	$F = A + B$	Addition
0	0	0	1	1	$F = A + B + 1$	Add with carry
0	0	1	0	0	$F = A + \bar{B}$	Subtract with borrow
0	0	1	0	1	$F = A + \bar{B} + 1$	Subtraction
0	0	1	1	0	$F = A - 1$	Decrement A
0	0	1	1	1	$F = A$	Transfer A
0	1	0	0	×	$F = A \wedge B$	AND
0	1	0	1	×	$F = A \vee B$	OR
0	1	1	0	×	$F = A \oplus B$	XOR
0	1	1	1	×	$F = \bar{A}$	Complement A
1	0	×	×	×	$F = \text{shr } A$	Shift right A into F
1	1	×	×	×	$F = \text{shl } A$	Shift left A into F

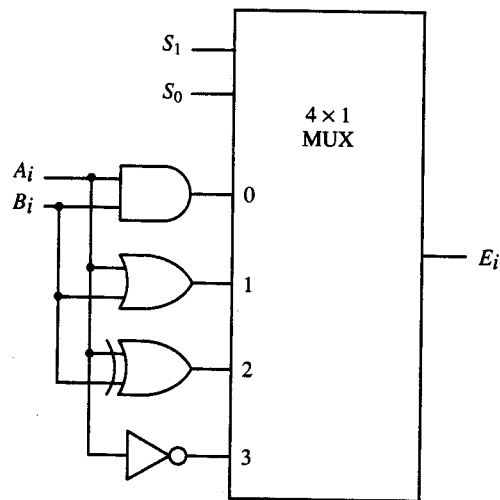


Figure 11: Unité logique à un étage

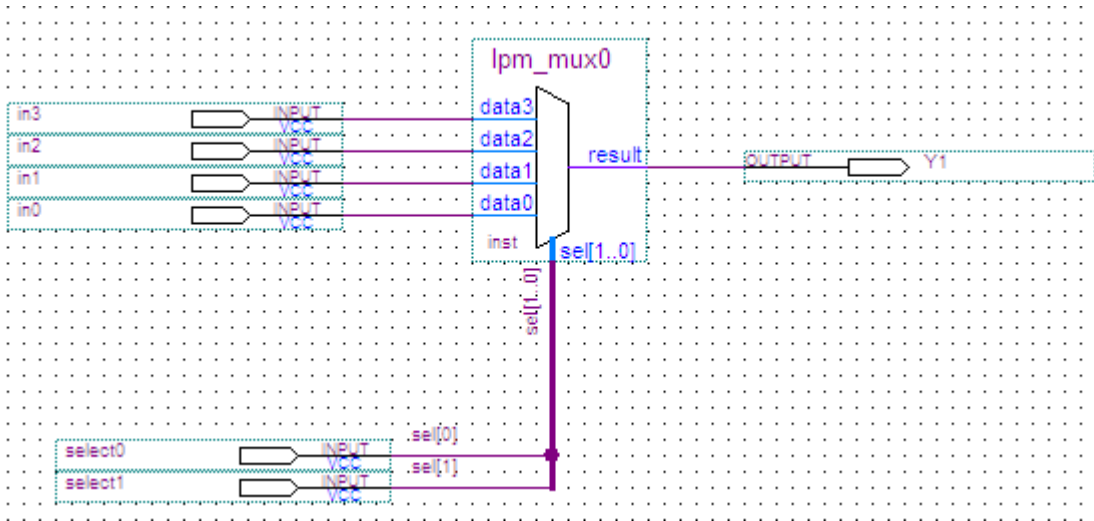


Figure 12 : Exemple d'un multiplexeur lpm-mux