Nom :
Prénom :
N° de place :

ENSICAEN
$1^{ière}$ année
électronique

Examen de circuits logiques 2015-2016

durée : 90 minutes, aucun document autorisé, calculatrice autorisée
Les réponses seront données sur ces feuilles à l'intérieur des espaces prévus à cet usage.

1- Conversion numérique (5 points)

◆ Complétez le tableau ci-dessous

Base 2 (12 bits)*	Base 10	Démarche, erreur de représentation
	89,98	
	-110,34	

* les nombres binaires seront représentés en complément à deux sur 12 bits en codage virgule fixe $Q_{8,4}$.

Rappel représentation $Q_{m,k}$ sur N bits: $b_{m+k-1}b_{m+k-2}.....b_k,b_{k-1}......b_2b_1b_0$; N=m+k

◆ Codez la valeur suivante en virgule flottante suivant la norme IEEE 754, présentez vos résultats intermédiaires.

A = -0,0045

Rappel : représentation en virgule flottante suivant la norme IEEE 754.

La valeur X est représentée suivant la forme : $X = (-1)^S \cdot 2^{E-127} \cdot 1,F$

X s'écrit alors en binaire virgule flottante : $\underbrace{S}_{\text{signe}}\underbrace{e_7e_6\ldots e_1e_0}_{E}\underbrace{f_{22}f_{21}\ldots f_2f_1f_0}_{F}$;

E et F sont codés en binaire non signé.

2- Synthèse combinatoire (5 points)

L'objectif de l'exercice est la synthèse d'un des deux blocs élémentaires constituant un multiplieur hardware de type Booth.

L'algorithme de Booth est largement utilisé pour les multiplications, car il permet de réduire de moitié le nombre de sommes partielles. Dans le cas d'une implémentation hardware, il conduit à une structure qui reste régulière et facilement extensible.

Les lignes ci-dessous décrivent le principe de l'algorithme de Booth d'ordre 2.

Algorithme de Booth :

soit A, en complément à 2, l'opérande de multiplication :

$$A = -a_{N-1}\,2^{N-1} + a_{N-2}\,2^{N-2} + \ldots + a_2\,2^2 + a_1\,2^1 + a_0\,2^0$$

en utilisant les relations : $2^i - 2^{i-1} = 2^{i-1}$ et $2^i = 2^{i+1} - 2.2^{i-1}$

on peut écrire :

$$A = (-2.a_{N-1}+a_{N-2}+a_{N-3})2^{N-2} + (-2.a_{N-3}+a_{N-4}+a_{N-5})2^{N-4}+\ldots+ (-2.a_3+a_2+a_1)2^2 + (-2.a_1+a_0+0)2^0$$

Il s'agit d'une décomposition de **Booth d'ordre 2**. Les termes du type $(-2.a_{i+1}+a_i+a_{i-1})$ ne peuvent prendre que 5 valeurs : -2, -1, 0, 1 et 2.

Quand un nombre B est multiplié par A : $B\times A = B\times(-2.a_{N-1}+a_{N-2}+a_{N-3})2^{N-2} + \ldots + B\times(-2.a_1+a_0+0)2^0$

les produits partiels obtenus sont du type : **B×(-2); B×(-1); B×0; B×1; B×2**

Le calcul du produit final résulte de la conjugaison de 3 types de commandes :

- complémentation à 2 (*-1),
- décalage à gauche (*2),
- annulation (*0).

Le multiplieur Booth est constitué d'un décodeur Booth et de cellules élémentaires. Le décodeur génère les commandes de complémentation, décalage et annulation et les cellules élémentaires exécutent ces 3 types de commande.

Une structure pour un format de données de 4 bits est illustrée ci-dessous

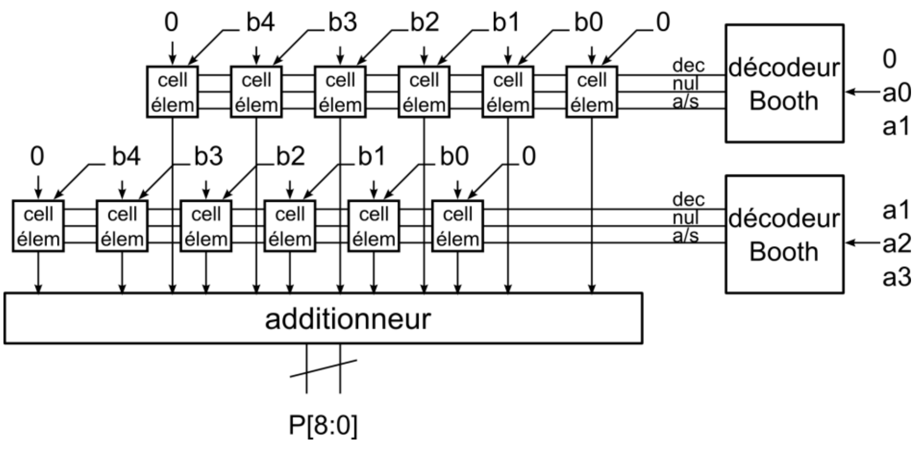

Figure 1: Structure de multiplieur Booth 4 bits

Q2.1. Compléter la table de vérité du décodeur Booth ci-après.

a_{i+1}	a_i	a_{i-1}	produit partiel	déc. (dec)	ann. (nul)	compl. (a/s)
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Q2.2. Synthétiser un circuit logique décodeur Booth à trois entrée, a_{i+1}, a_i, a_{i-1}, correspondant à la table précédente.

3- Implémentation d’une multiplication série de type Booth (10 points)

L’objectif de cet exercice est la conception d’une machine à états finis contrôlant les étapes d'une multiplication de type **Booth d’ordre 1** en série sur une unité de traitement donnée.

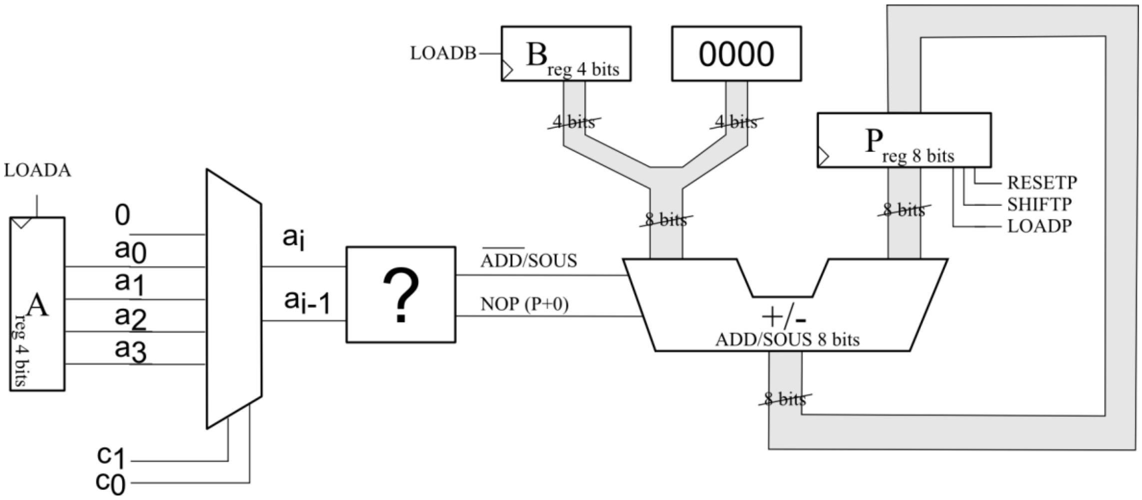

Figure 2: Architecture de l'unité de traitement pour l'implémentation de l'algorithme de Booth pour la multiplication B×A. Avec le multiplieur contenu dans le registre A et le multiplicande dans le registre B. Les résultats intermédiaires sont stockés dans le registre de travail P. Le résultat de multiplication se trouvera dans le registre P à la fin de l'algorithme.

L’additionneur/soustracteur 4bits permet 3 opérations : P+B, P-B et P+0.

Décomposition Booth d'ordre 1

soit A, en complément à 2, l’opérande de multiplication :

$$A = -a_{N-1}\, 2^{N-1} + a_{N-2}\, 2^{N-2} + \ldots. + a_2\, 2^2 + a_1\, 2^1 + a_0\, 2^0$$

en remarquant que : $2^{i-1} = 2^i - 2^{i-1}$

on peut le réécrire :

$$A = (-a_{N-1}+a_{N-2})2^{N-1}+(-a_{N-2}+a_{N-3})2^{N-2}+(-a_{N-3}+a_{N-4})2^{N-3}+\ldots.+(-a_2+a_1)2^2+(-a_1+a_0)2^1+(-a_0+0)2^0$$

Le calcul du produit du multiplicande B par le multiplieur A peut alors s'écrire :

$B\times A = B\times(-a_{N-1}+a_{N-2})2^{N-1}+B\times(-a_{N-2}+a_{N-3})2^{N-2}+\ldots.+B\times(-a_2+a_1)2^2+B\times(-a_1+a_0)2^1+B\times(-a_0+0)2^0$

où l'on identifie des produits partiels $B\times(-a_i+a_{i-1})$ pouvant prendre 3 valeurs en fonction des a_i :

soit -B, soit B, soit 0.

La multiplication peut alors être abordée comme une somme de produits partiels décalés.

Exemple sur 4 bits : A=6 ; B=3

> *$A = 0110\ (0\times2^3+1\times2^2+1\times2^1+0\times2^0)$; $B=0011\ (0\times2^2+0\times2^2+1\times2^1+1\times2^0)$*
>
> *Décomposition Booth d'ordre 1 de A :*
>
> *$A = (-0+1)\times2^2+(-1+1)\times2^2+(-1+0)\times2^1+(-0+0)\times2^0$*
>
> *La multiplication devient alors :*
>
> *$B\times A = 3\times(-0+1)\times2^3+3\times(-1+1)\times2^2+3\times(-1+0)\times2^1+3\times(-0+0)\times2^0= 3\times2^3-3\times2^1=24-6=18$*

La table ci-après reprend les différents cas possibles.

a_i	a_{i-1}	$-a_i+a_{i-1}$	**opération**
0	0	0	Pas d'opération, ajout de 0 au résultat partiel ($\times 0$)
1	0	-1	Soustraire le multiplicande B au résultat partiel ($\times -1$)
1	1	0	Pas d'opération, ajout de 0 au résultat partiel ($\times 0$)
0	1	1	Ajouter le multiplicande B au résultat partiel ($\times 1$)

<u>Algorithme pour la multiplication série de type Booth d'ordre 1</u>

- En fonction des combinaisons des a_i, a_{i-1}, on additionne, on soustrait ou on ne fait rien.
- On décale à droite après chaque opération (addition, soustraction, rien).

Opération élémentaire		**P** $P_7P_6P_5P_4P_3P_3P_1P_0$	**A** $a_3a_2a_1a_0$	**Commentaires**
Chargement des registres		0000 0000	0110 0	Initialisation des registres. Chargement du multiplieur (A) et du multiplicande (B) et mise à 0 du résultat partiel (P).
		0000 0000		
Pas d'opération (P+0)	+	0000 0000	011<u>0 0</u>	-a0+0=0, pas d'opération
Chargement résultat, LOADP		0000 0000		
Décalage à droite	→	0000 0000		Décalage arithmétique à droite
		0000 0000		
Soustraction de (+(-B))	+	**1101** 0000	01<u>10</u> 0	-a1+a0=-1, soustraction de B
Chargement résultat, LOADP		1101 0000		
Décalage à droite	→	1110 1000		Décalage arithmétique à droite
Pas d'opération (P+0)		1110 1000		
	+	0000 0000	0<u>11</u>0 0	-a2+a1=0, pas d'opération
Chargement résultat, LOADP		1110 1000		
Décalage à droite	→	1111 0100		Décalage arithmétique à droite
		1111 0100		
Addition de B (+B)	+	**0011** 0000	<u>01</u>10 0	-a3+a2=1, addition de B0
Chargement résultat, LOADP		0010 0100		
Décalage à droite	→	**0001 0010**	**produit**	Décalage arithmétique à droite

Tableau 1: Algorithme de multiplication Booth. Exemple de la multiplication de 3 (0011) dans le registre B par 6 (0110) dans le registre A. Remarque : la soustraction de 3 est représentée dans le tableau par l'addition de -3 (1101).

Les commandes ADD/SOUS et NOP(P+0) dépendent des a_i ; a_{i-1}.

Q3.1. Exprimez ADD/SOUS et NOP en fonction de a_i et a_{i-1}.

Q3.2. Concevez la machine à états finis contrôlant les étapes de l'algorithme. La conception devra conduire à l'expression des tables de vérité de la machine à états finis. Les équations logiques et le dessin au niveau portes ne sont pas demandés. Vous pouvez vous appuyer sur le tableau décrivant l'évolution des signaux de commande de l'unité de traitement.

Le signal MULTstart démarre l'algorithme. Le registre P conserve le résultat de multiplication à la fin de l'algorithme. Une sortie MULTend indique que le résultat est disponible.

Les commandes LOAD effectuent le chargement des valeurs présentes en entrée des registres (A, B et P). La commande SHIFTP effectue le décalage arithmétique du contenu du registre P d'un bit à droite. La commande RESETP met à 0000 0000 le contenu du registre P.

Le multiplexeur sélection 2 bits consécutifs en fonction des entrées de commandes C_1 et C_0, selon la table de vérité ci-contre.

On considère que la période de l'horloge de cadencement, clk, de la machine d'état est supérieure au temps de calcul de l'additionneur/soustracteur.

Entrées de commande		Sorties du multiplexeur	
C_1	C_0		
0	0	a_0	0
0	1	a_1	a_0
1	0	a_2	a_1
1	1	a_3	a_2

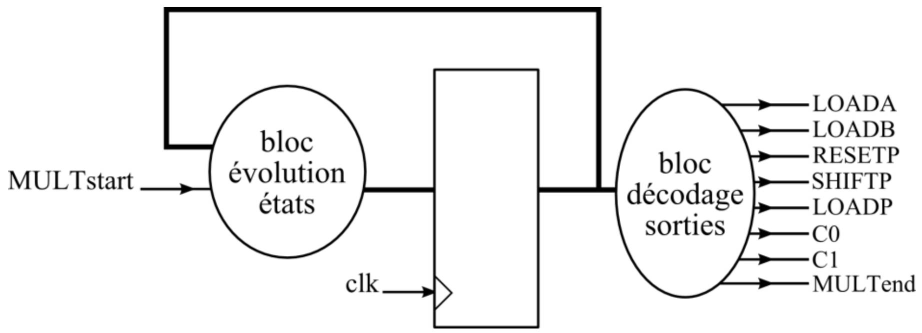

Figure 3: Schéma bloc de la machine à états finis contrôlant l'algorithme de division

description	état/cycle	LOADB	RESETP	SHIFTP	LOADP	LOADA	MULTend	C_1	C_0
multiplication terminée, attente MULTstart	0	0	0	0	0	0	1	0	0
début multiplication, initialisation registres	1	1	1	0	0	1	0	0	0