Examen Architecture des Ordinateurs – 6 Janvier 2020

Tous documents autorisés. Calculatrices interdites. Durée 2h

Pour toutes les questions, une explication **concise** est nécessaire pour que la réponse soit prise en compte.

1. Fonctions booléennes et circuits combinatoires [10 pts]

On cherche à construire un circuit effectuant la multiplication de deux nombres entiers positifs codés sur 2 bits.

Q1. Regardons d'abord le cas simple de deux nombres codés sur 1 bit chacun.
a) Sur combien de bit(s) il faut coder le résultat afin de ne pas avoir de dépassement
b) Donner l'expression algébrique du résultat

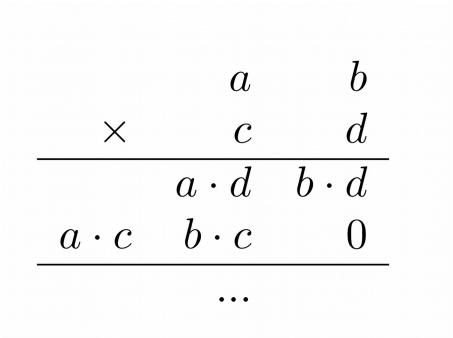
a) 1 bit (1 * 1 =1)
b) res = a0 . b0

Q2. Regardons maintenant le cas de deux nombres codés sur 2 bits chacun. Justifier pourquoi il faut 4 bits pour contenir le résultat afin de ne pas avoir d'overflow (impossibilité de représenter le résultat)
2 bits : [0,3], Res \in [0,9] donc 4 bits

Q3. Écrire la table de vérité, les deux nombres sur deux bits en entrée seront notés a1a0 et b1b0 ; et les bits de sortie c3c2c1c0.

a1	a0	b1	b0	c3	c2	c1	c0	p
0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	1
0	0	1	0	0	0	0	0	1
0	0	1	1	0	0	0	0	1
0	1	0	0	0	0	0	0	1
0	1	0	1	0	0	0	1	0
0	1	1	0	0	0	1	0	1
0	1	1	1	0	0	1	1	0
1	0	0	0	0	0	0	0	1
1	0	0	1	0	0	1	0	1
1	0	1	0	0	1	0	0	1
1	0	1	1	0	1	1	0	1
1	1	0	0	0	0	0	0	1
1	1	0	1	0	0	1	1	0
1	1	1	0	0	1	1	0	1
1	1	1	1	1	0	0	1	0

Pour calculer les expressions algébriques des différents bits de résultat, on utilisera la façon « classique » de faire une multiplication :



Q4. Donner une expression pour c0 en vous aidant éventuellement de la question **Q1**.

c0 = a0.b0

Q5. Ensuite, calculer l'expression de c1 à partir de la somme (ou logique) de deux termes (produit de variables d'entrée).

c1 = a1.b0 XOR a0.b1

Q6. Calculer c2 et c3 en faisant attention à bien prendre en compte les retenues éventuelles.

Soit **r1 = a1.b0.a0.b1** la retenue du calcul précédent.
c2 = a1.b1 XOR r1
Soit **r2 = a1.b1.r1** la retenue de c2
c3 = r2 = a1.b1.a1.b0.a0.b1 = a1.b0.a0.b1

Q7. En supposant que le temps de propagation d'une porte logique (ET,OR,XOR,AND, NOT) soit de τ (quelque-soit le nombre d'entrée de la porte). Quel est le temps de propagation de la multiplication entre les entrées et les sorties.

c0 : 1 τ
c1 : 2 τ
c2 : 2 τ
c3 : 1 τ
Tps de propagation 2 τ

On veut construire un indicateur p nous indiquant si le résultat de la multiplication va être pair ou non en fonction des variables d'entrée a1,a0,b1,b0.

Q8. Donner la forme conjonctive normale (FCN) de p.

p = (a1 + $\overline{a0}$ + b1 + $\overline{b0}$).(a1 + $\overline{a0}$ + $\overline{b1}$ + $\overline{b0}$).($\overline{a1}$ + $\overline{a0}$ + b1 + $\overline{b0}$).($\overline{a1}$ + $\overline{a0}$ + $\overline{b1}$ + $\overline{b0}$) = M5.M7.M13.M15

Q9. A l'aide d'une table de Karnaugh, donner une expression réduite de cet indicateur p (la table de Karnaugh est demandée).

	b0		$\overline{b0}$		
b1	1	0	1	1	$\overline{a1}$
	1	0	1	1	a1
$\overline{b1}$	1	0	1	1	
	1	0	1	1	$\overline{a1}$
	$\overline{a0}$	a0		$\overline{a0}$	

P = $\overline{c0}$ = $\overline{a0}$ + $\overline{b0}$

Q10. En utilisant un multiplexeur 8 vers 1, implémenter la fonction correspondant à l'indicateur p. Vous indiquerez comment les différentes entrées du multiplexeur (les entrées, et les variables de sélection) sont câblées à l'aide des entrées a1,a0,b1,b0.

Variables de contrôle : c0 = b1, c1 = a0, c2 = a1
Entrées :
* e0 = 1
* e1 = 1
* e2 = $\overline{b0}$
* e3 = $\overline{b0}$
* e4 = 1
* e5 = 1
* e6 = $\overline{b0}$
* e7 = $\overline{b0}$

2. Circuits séquentiels et Bascules [6 pts]

On souhaite réaliser un compteur-décompteur automatique par 8. Il part de 0, compte jusqu'à 7 et décompte jusqu'à 0 et puis recommence :

$0 \rightarrow 1 \rightarrow 2 \rightarrow \ldots \rightarrow 7 \rightarrow 6 \rightarrow \ldots \rightarrow 0 \rightarrow \ldots \rightarrow 7 \rightarrow \ldots$

Q11. On commencer par établir le fonctionnement d'un compteur-décompteur par 8 à l'aide d'une variable E. On sera en mode compteur quand E=0, et en mode décompteur quand E=1. Donner le nombre de bascules nécessaire et donner la table de vérité.

3 bascules (8 états)

Etat	E	Q2	Q1	Q0	D2	D1	D0
0	0	0	0	0	0	0	1
1	0	0	0	1	0	1	0
2	0	0	1	0	0	1	1
3	0	0	1	1	1	0	0
4	0	1	0	0	1	0	1
5	0	1	0	1	1	1	0
6	0	1	1	0	1	1	1
7	0	1	1	1	0	0	0

0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
2	1	0	1	0	0	0	1
3	1	0	1	1	0	1	0
4	1	1	0	0	0	1	1
5	1	1	0	1	1	0	0
6	1	1	1	0	1	0	1
7	1	1	1	1	1	1	0

On cherche maintenant à faire changer l'état de E lorsque l'on **arrive** sur l'état 0 ou l'état 7 à l'aide d'une quatrième bascule Qs.

Q12. On veut définir une variable A, qui sera à 1 lorsque l'on se trouve dans l'état 6 en mode compteur ou dans l'état 1 en mode décompteur (sinon A sera à 0). Donner l'expression de A en fonction de Q2, Q1, Q0 et E.

$A = \overline{E}.Q2.Q1.\overline{Q0} + E.\overline{Q2}.\overline{Q1}.Q0$

Q13. On veut maintenant trouver le câblage Ds de la quatrième bascule afin que la bascule change d'état ($0 \rightarrow 1$ ou $1 \rightarrow 0$) lorsque l'on passe de l'état 6 à l'état 7 en mode compteur ou lorsque l'on passe de l'état 1 à l'état 0 en mode décompteur. Donner l'expression de Ds en fonction de Qs et de A afin de finaliser le circuit final.

$Ds = A.\overline{Qs} + \overline{A}.Qs$

Q14. Comment câbler la variable E afin de réaliser la bascule ?

Il faut que la variable E soit câblée sur Qs.

3. Pipeline [4 pts]

Soit le jeu d'instructions suivant : (on dispose des registres entiers R0 à R31 et des registres flottants F0 à F31).

Instructions sur les flottans :	
lf Fdest, adresse	Charge un flottant depuis l'adresse (l'adresse est au format N(R) avec N un immédiat et R un registre entier
sf Fsrc, adresse	Stocker un flottant dans l'adresse
fadd Fdest, Fsrc1, Fsrc2	Fdest ← Fsrc1 + Fsrc2 (addition flottante simple précision)
fsub Fdest, Fsrc1, Fsrc2	Fdest ← Fsrc1 - Fsrc2 (soustraction flottante simple précision)
fmul Fdest, Fsrc1, Fsrc2	Fdest ← Fsrc1 * Fsrc2 (multiplication flottante simple précision)
fdiv Fdest, Fsrc1, Fsrc2	Fdest ← Fsrc1 / Fsrc2 (division flottante simple précision)
Instructions sur les entiers :	
lw Rdest, adresse	Charge un entier depuis l'adresse
sf Rsrc, adresse	Stocke un entier dans l'adresse
add Rdest, Rsrc1, Rsrc2	Rdest ← Rsrc1 + Rsrc2 (addition avec des entiers)
addi Rdest, Rsrc1, Imm	Rdest ← Rsrc1 + Imm (addition avec des entiers)
Branchements :	

bne Ra, Rb, Label bneq R, label	Si Ra ≠ Rb : goto Label Si R ≠ 0 : goto Label

Les additions, soustractions et multiplications flottantes sont pipelinées. Une nouvelle instruction peut démarrer à chaque cycle. Les latences sont de 2 cycles pour lf et fournies par les résultats de l'exercice 1 pour les multiplications et additions flottantes. La division a une latence de 15 cycles. Elle n'est pas pipelinée : une nouvelle division ne peut commencer que lorsque la division précédente est terminée. On utilisera les latences suivantes :

Instructions	*Latence*	*Pipelinée ?*
lf	2	oui
fadd, fsub	4	oui
fmul	6	oui
fdiv	15	non

On considère la code suivante :

```
int N=100
float X[N], Y[N], a;
for (int i=0 ; i<N ; i++)
        Y[i]= X[i] * a;
```

On supposera que la variable 'a' est déjà initialisée dans le registre F31. Le tableau X est implémenté à l'adresse 0x10000000 et Y à 0x10000400. La variable R1 est initialisée à 0x10000000.

Q15. Ecrire le code assembleur du coeur de la boucle sans aucun déroulage en optimisant le nombre de cycles. Vous ferez attention à spécifier pour chaque instruction sur quel cycle elle démarre. Combien de cycles par itérations obtient-on ?

Cycle	Instructions
1	lf f1, 0(r1)
3	fmul f2, f1, f31
4	addi r1, r1, 4
9	sf f2, 396(r1)
10	BNE

10 cycles / itération.

Q16. Effectuer un déroulage d'ordre 2, toujours en optimisant le nombre de cycles et en spécifiant le cycle sur lequel démarra chaque instruction. Combien de cycles par itérations obtient-on ?

Cycle	Instructions
1	lf f1, 0(r1)
2	lf f2, 4(r1)
3	fmul f1, f1, f31
4	fmul f2, f2, f31
5	addi r1, r1, 8
9	sf f1, 392(r1)
10	sf f2, 396(r1)
11	BNE

11/2=5.5 cycles/ite

Q17. Combien de cycles par itération obtient-on si au lieu d'une multiplication on a une division ? Que faire pour améliorer la situation ?

La division n'est pas pipelinable : la première commence cycle 3→18→33+derniersf+BNE =35 cycles. Remplacer par la multiplication par 1/a.