Examen Architecture des Ordinateurs – 17 Décembre 2018

Tous documents autorisés. Calculatrices interdites. Durée 2h.

Pour toutes les questions, une explication **concise** est nécessaire pour que la réponse soit prise en compte. Le barème est donné à titre indicatif

1. Fonctions booléennes et circuits combinatoires [12 pts]

On cherche ici à construire un circuit permettant la comparaison de deux nombres naturels sur n bits. Pour commencer, on va regarder comment faire des tranches de comparaison de 1bit.

Q1. Soient deux variables sur un bit, X et Y. On veut construire un circuit qui prend en entrée X et Y et en sortie permet d'avoir Z_1Z_0 = 00 si X=Y, 01 si X>Y et 10 si X<Y. Donner la table de vérité de ce circuit.

X	Y	Z_1	Z_0
0	0	0	0
0	1	1	0
1	0	0	1
1	1	0	0

Q2. Implémenter ce circuit à l'aide des portes logiques OR, AND et NOT seulement.
Z_1 = X.Y et Z_1 = X.Y

On souhaite pouvoir faire des comparaisons sur N bits à partir de tranches de 1 bit.
Q3. On considère deux variables de deux bits X_1X_0 et Y_1Y_0. La comparaison entre X_0 et Y_0 vient d'être faite et le résultat est contenu dans Z_1Z_0.

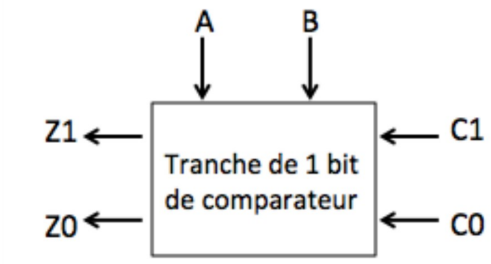
1. Pour quels valeurs de X_1 et Y_1 il est possible de conclure sans avoir besoin de la sortie de la comparaison entre les bits de poids faible (X_0,Y_0).
   Si (X_1,Y_1) = (0,1) ou (1,0) on peut conclure dans le reste
2. Dans ce cas, quels valeurs donner aux variables de sortie (Z_1Z_0) de la comparaison entre X_1 et Y_1 ?
   (Z_1Z_0) = (10) ou (01)
3. Dans le cas ou il est impossible de conclure sans la comparaison des bits de poids faible (X_0,Y_0), que doit-on renvoyer en sortie ?
   On renvoit Z_1Z_0 venant de l'étage précédent.

Q4. Modifier votre circuit 1 bit en rajoutant deux autres entrées C_1C_0 qui vont prendre en compte le potentiel résultat venant des bits de poids plus faible. Écrire la table de vérité du nouveau circuit.

	X	Y	C_1	C_0	Z_1	Z_0
0	0	0	0	0	0	0
1	0	0	0	1	0	1

2	0	0	1	0	1	0
3	0	0	1	1	0	0
4	0	1	0	0	1	0
5	0	1	1	0	1	0
6	0	1	0	1	1	0
7	0	1	1	1	1	0
8	1	0	0	0	0	1
9	1	0	0	1	0	1
10	1	0	1	0	0	1
11	1	0	1	1	0	1
12	1	1	0	0	0	0
13	1	1	0	1	0	1
14	1	1	1	0	1	0
15	1	1	1	1	0	0

Q5. En utilisant les éléments de tranche 1 bit suivants (figure ci-dessous), dessiner le schéma (circuits et connexions) d'un comparateur 4 bits.



On met une première tranche avec C0 et C1 à 0, puis A et B prennent en entrée les bits de poids faibles. Ensuite Z0 et Z1 sont branchés sur C0 et C1 de la tranche suivante ….

Q6. On considère τ, le temps de propagation d'une porte logique. Quel est le temps de propagation d'une tranche de 1 bit ? (temps maximal entre une entrée et une sortie)

$3\,\tau$

Q7. En déduire le temps de propagation du circuit total pour n=4 bits.
$3\,\tau^* 4 = 12\,\tau$

On considère maintenant seulement le besoin de savoir si deux variables sur n bits sont différentes (ou non).
Q7. Écrire la table de vérité de ce circuit pour n=2 bits (donc deux variables de deux bits chacune).
0 pour m0, m6, m10 et m15.

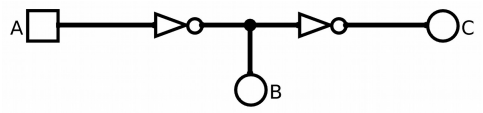
Q8. Donner l'écriture disjonctive normale (vous pouvez utiliser les noms m_i et M_i pour les minterms/maxterms respectivement).

F = m1+m2+m3+m4+m5+m7+m8+m9+m11+m12+m13+m14
il manque m0, m6, m10 et m15

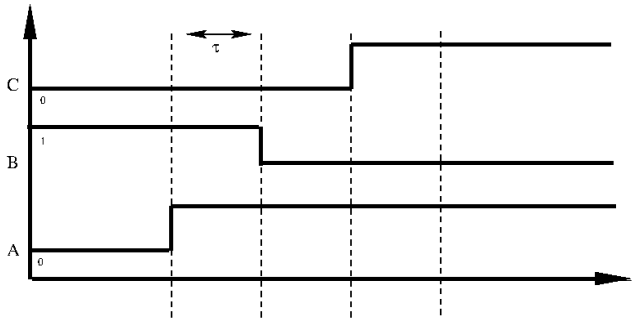
Q9. A l'aide d'un diagramme de Karnaugh, trouver une expression réduite.
$F= Y_0X_0 + Y_0X_0 + Y_1X_1 + Y_1X_1 = X_0$ XOR $Y_0 + X_1$ XOR Y_1
Q10. Implémenter ce circuit à l'aide d'un multiplexeur 1 parmi 16.
Cas inusuel. On branche X0,X1,Y0,Y1 sur les variables de sélections C0-C3 du multiplixeur. Ensuite, pour chaque entrée Ei on met 0 où 1 selon la table de vérité.

2. Circuits séquentiels et Bascules [6 pts]

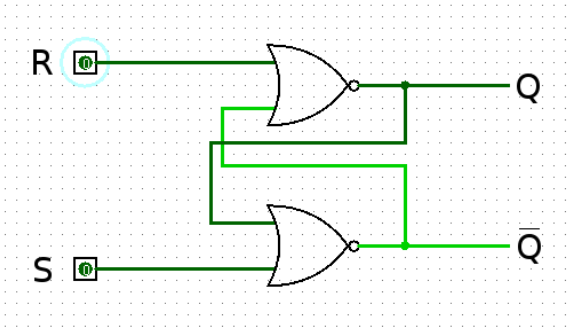
On s'intéresse au chronogramme de circuits séquentiels. On considère que chaque porte logique a un délai de τ : si un signal change d'état à l'entrée d'une porte, il faudra attendre un temps τ avant d'observer un changement sur la sortie. Un exemple est donné sur le diagramme ci-dessous.



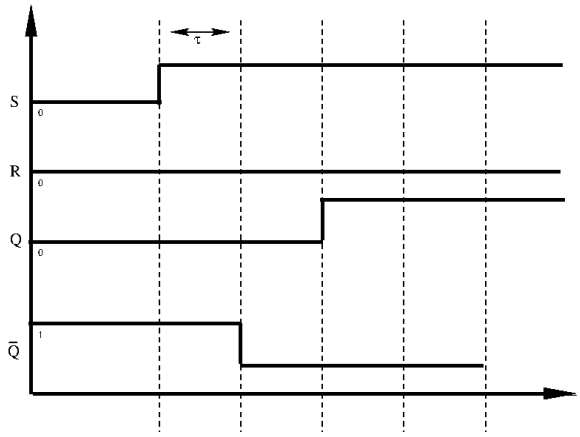
On fixe la valeur de A, le chronogramme de ce circuit pour B et C sera :



Q11. On considère la bascule RS suivante



On suppose que Q=0 (Q=1) et que R=S=0 comme condition initiale. Sur le chronogramme (figure ci-dessous), illustrer le changement dans les valeur de Q et Q. (vous recopierez le chronogramme sur votre copie).



Q12. « Réaliser » un compteur par 4 à l'aide de deux bascules : sur chaque front montant de l'horloge, le compteur suivra les transitions suivantes (seule la table de vérité est demandée).

$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 0 \rightarrow \ldots$

Q1	Q0	D1	D0
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

Q13. On souhaite maintenant réaliser un compteur pair/impair qui effectue les transitions suivantes sur chaque front montant de l'horloge :

$1 \rightarrow 3 \rightarrow 5 \rightarrow 7 \rightarrow 1 \rightarrow 3 \ldots$ ou

$0 \rightarrow 2 \rightarrow 4 \rightarrow 6 \rightarrow 0 \rightarrow 2 \ldots$

Combien de bascules D sont nécessaires ? **3 BASCULES**

Q14. « Réaliser » le compteur (ici, seule la table de transition est demandée).

Q2	Q1	Q0	D2	D1	D0
0	0	0	0	1	0
0	0	1	0	1	1
0	1	0	1	0	0
0	1	1	1	0	1
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	0	0	0
1	1	1	0	0	1

Q15. On souhaite détecter, à l'aide d'une variable sur un bit p, la présence ou non de l'état « 6 » (p=1 si l'état 6 est détecté, 0 sinon). Expliquer comment câbler le bit p au compteur.

P = Q2.Q1.Q0

3. Pipeline [5pts]

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, address	Charge un flottant depuis l'adresse (l'address est au format N(R) avec N un immédiat et R un registre entier)
sf Fsrc, address	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, address	Charge un entier depuis l'adresse
sf Rsrc, address	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	Si Ra ≠ Rb : goto Label
bneq R, 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 le tableau ci-dessous 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 boucle suivante :

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

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. Écrire le code assembleur du cœur de la boucle 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 ?

1	Lf f0,0(r1)
2	Lf f1, 400(r1)
3	Fmul f0,f0,f31
4	Addi r1,r1,4
5	
6	
7	
8	
9	Fadd f0,f0,f1
10	
11	
12	
13	Sf f0,796(r1)
14	BNE …

On obtient 14 cycles par itération.

Q16. Effectuer un « déroulage » d'ordre 2 et d'ordre 4, toujours en optimisant le nombre de cycles et en spécifiant le cycle sur lequel démarrera chaque instruction. Combien de cycles par itération (de la boucle C initiale) obtient-on ?

Déroulage d'O(2) :

1	Lf f0,0(r1)
2	Lf f1, 400(r1)
3	Fmul f0,f0,f31
4	Lf f2, 4(r1)
5	Lf f3, 404(r1)
6	Fmul f2, f2, f31
7	Addi r1, r1, 4

8	---
9	Fadd f0,f0,f1
10	---
11	---
12	Fadd f2, f2, f3
13	Sf, f0, 792(r1)
14	---
15	---
16	Sf f2, 796(r1)
17	BNE …

On obtient donc : 17/2 = 8.25 cycles/ité

Pour O(4) on trouve 23 cycles, et donc 23/4 = 5.75 cycles/ité