Université Grenoble Alpes
ALM MIAGE L3
Examen d'Architecture des ordinateurs - première session
Décembre 2017
Durée : 3h00 en tout (~1h00 Architecture des ordinateurs, ~1h Système, ~1h Réseau).
1 document A4 commun ASR autorisé. Sujet Architecture des ordinateurs sur 1 recto-verso.

I. Erreur ? (barème indicatif : 9 points)

Q1. Poser et réaliser en binaire sur 8 bits, les 3 additions suivantes :

- 0101 1101 + 0101 0110 (addition 1),
- 0110 1010 + 1101 1100 (addition 2) et
- 1010 0101 + 1110 0111 (addition 3).

Il n'est pas demandé de calculer les valeurs en décimal, par contre, pour chaque addition indiquer si le résultat sur 8 bits est juste en supposant que les nombres manipulés sont :

- Soit des entiers non signés en binaire naturel (hypothèse « non-signé »)
- Soit des entiers signés représentés en binaire complément à 2 (hypothèse « signé »).

Pour la suite, vous supposerez que les nombres manipulés sont **des entiers non signés en binaire naturel**.

Parfois en regardant les bits de poids fort des deux termes d'une addition, on peut savoir si le résultat de l'addition sur n bits sera juste ou faux avant de faire le calcul effectif pour tous les bits.

Q2a. Indiquer pour les 3 cas suivants si le résultat sur 8 bits est connu et nécessairement juste ou faux ou s'il est indécis tant que l'on ne connait pas tous les bits xxx xxxx et yyy yyyy :

- 0xxx xxxx + 0yyy yyyy
- 0xxx xxxx + 1yyy yyyy
- 1xxx xxxx + 1yyy yyyy

Dans les cas indécis, il faut regarder les autres chiffres xxx xxxx et yyy yyyy pour déterminer si le résultat des calculs va être juste ou faux en opérant de même (en regardant les bits de poids forts). Ceci donne l'idée d'un circuit permettant de déterminer si le résultat d'une addition va être juste en regardant les opérandes poids forts d'abord (on regarde les bits x_7 et y_7 puis si c'est un cas indécis, on regarde les bits x_6 et y_6 puis si c'est un cas indécis, on regarde les bits x_5 et y_5 en même temps …).

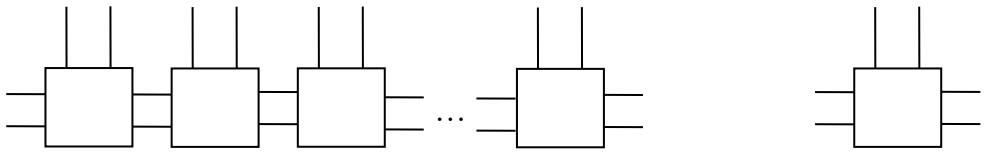
Q2b. Donner la table de vérité d'une cellule de ce circuit ayant en entrée :

- un bit de chaque opérande (x_k et y_k de même rang k),
- un bit « fait » indiquant si l'analyse des bits précédents (de rang > k) de chaque opérande a pu décider si l'addition est juste ou pas (fait=0 : le calcul de la justesse de l'addition ne peut pas encore être fait ; fait=1 : le calcul de la justesse de l'addition a été fait) et
- un bit « juste » indiquant, quand « fait » est vrai si l'addition est juste ou pas (juste =0 : addition fausse, juste =1 : addition juste).

En sortie cette cellule de circuit doit donner deux nouvelles valeurs 'fait', 'juste' : déterminant si le calcul de la justesse de l'addition a été fait et si le résultat de l'addition est effectivement juste ou pas.

Q2c. Dessiner la cellule de circuit associée à la table de vérité en utilisant des portes et/ou/non seulement.

Q3. Le circuit complet final prévu aura la forme suivante :



Circuit complet — Cellule seule

Recopier ce dessin sur votre copie et indiquer pour le circuit complet (ne pas répondre sur le sujet) :

- le sens de transmission des informations (horizontalement, les informations vont-elles de droite à gauche ou de gauche à droite ?)
- où se trouvent x_k, y_k pour k=7..0
- où se trouvent et quelles sont les valeurs initiales de « fait » et « juste »
- où se trouvent les valeurs finales de « fait » et « juste »

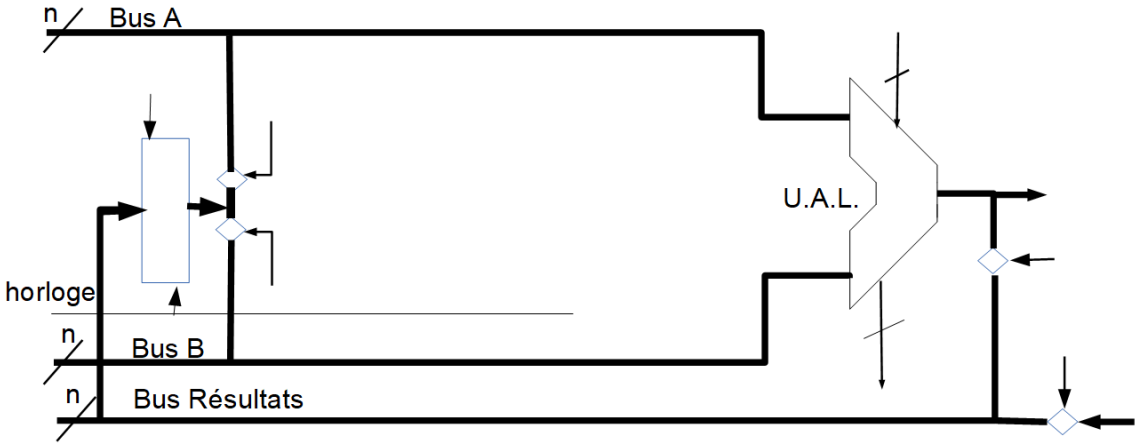
Question bonus : Évaluer la complexité de ce circuit.

II. Division (barème indicatif : 7 points)

Pour réaliser une division entière Q←A/B, l'algorithme suivant doit être implémenté :

```
Début
   Tmp← B, N←0, Q←0, R←A
   Tant que Tmp ≤ A :
   |  Tmp ← Tmp*2, N← N+1
    Fin tant que
   Tant que N > 0 :
   |  Tmp ← Tmp / 2, N←N-1, Q ← Q*2
   |  Si R ≥ Tmp :
   |  |  R ← R – Tmp, Q←Q+1
   |  Fin si
    Fin tant que
Fin
```

Q1. Spécifier et dessiner la partie opérative nécessaire à la réalisation d'un circuit PC/PO faisant le calcul d'une division selon l'algorithme donné. Préciser clairement les entrées et sorties de cette partie opérative ainsi que les valeurs initiales nécessaires. Vous pourrez vous inspirez de l'embryon de schéma suivant et le recopier sur votre copie pour faire votre dessin (ne pas répondre sur le sujet) :



Q2. Dessiner l'automate formel (ou automate d'actions) commandant la partie opérative précédente pour réaliser l'algorithme donné. Dans cette question, les entrées et sorties de l'automate pourront rester symboliques. La question suivante concerne l'implémentation et pourra/devra introduire des représentations binaires plus précises.

Q3. Réaliser une partie représentative du circuit exécutant cet automate :

- Donner les premières lignes des tableaux donnant les représentations binaires des états et signaux (signaux : c'est à dire les entrées et sorties de l'automate).
- Donner les premières lignes de la table de vérité de la fonction de transition et de la fonction de sortie.
- Dessiner une partie représentative du circuit à l'aide de portes et/ou/non et de bascules.
- Relier ce circuit au circuit de **Q1**.

III. Question de temps (barème indicatif : 4 points)

Un même programme a été réalisé sous forme d'un circuit combinatoire (CircComb), d'un circuit séquentiel à flot de données (CircFlotDonnées), d'un circuit séquentiel sous forme PC/PO (PC/PO) et sous forme d'un programme assembleur à exécuter par un ordinateur (ASM).

Q1. Indiquez pour chaque forme de réalisation comment le temps d'exécution du programme pourra être évalué. Donner les éléments caractéristiques, les moyens d'évaluer ces éléments caractéristiques et les calculs nécessaires pour évaluer ce temps global.

Q2. Peut-on dire a priori quelle réalisation sera la plus rapide, la plus lente, quel gain peut être prévu ? Donner des exemples ou contre-exemples en choisissant des programmes parmi ceux que l'on a vu cette année, ou d'autres à votre convenance.

IV. Erreur ?

Q1.

- 0101 1101
  + 0101 0110
  = 1011 0011 correct pour « non-signé », incorrect pour « signé »,
- 0110 1010
  + 1101 1100
  = 0100 0110 incorrect pour « non-signé », correct pour « signé »,
- 1010 0101
  + 1110 0111
  = 1000 1100 incorrect pour « non-signé », correct pour « signé »,

Pour la suite les nombres manipulés sont **des entiers non signés en binaire naturel**.

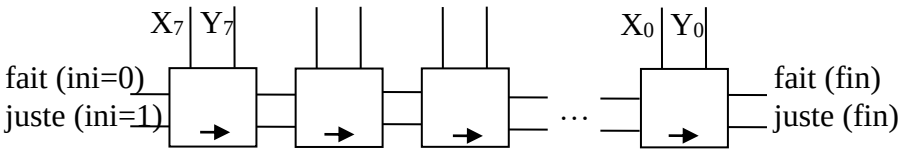
Q2a.

- 0xxx xxxx + 0yyy yyyy : correct pour « non-signé »
- 0xxx xxxx + 1yyy yyyy : indécis pour « non-signé »
- 1xxx xxxx + 1yyy yyyy : incorrect pour « non-signé »

Q2b. Table de vérité pour « non-signé » (fait, juste , x_k, y_k) ➔ (fait, juste)

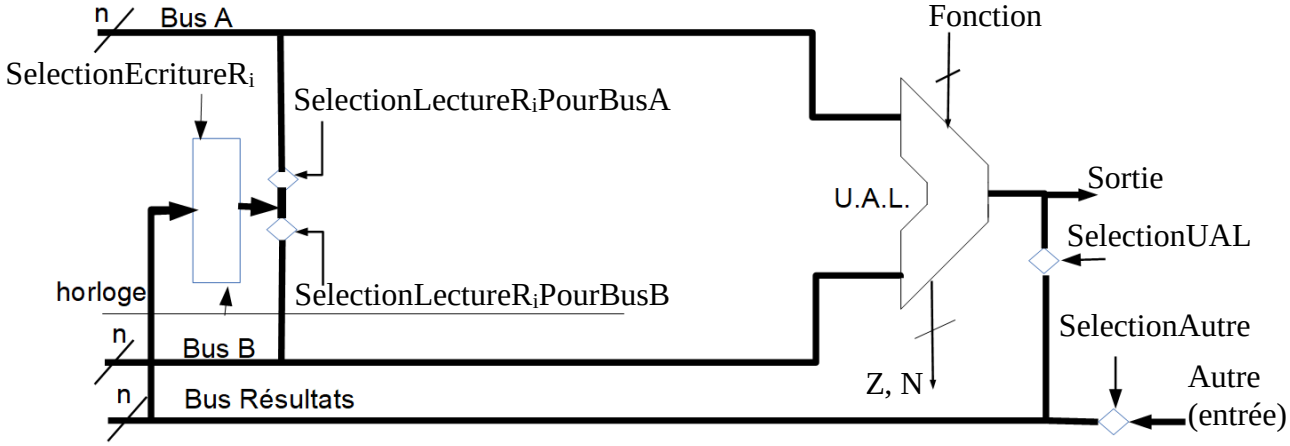
Fait	Juste	Xk	Yk	Fait	Juste
0	*	0	0	1	1
0	*	0	1	0	(1)
0	*	1	0	0	(1)
0	*	1	1	1	0
1	X	*	*	1	X

Q3. Circuit complet final :

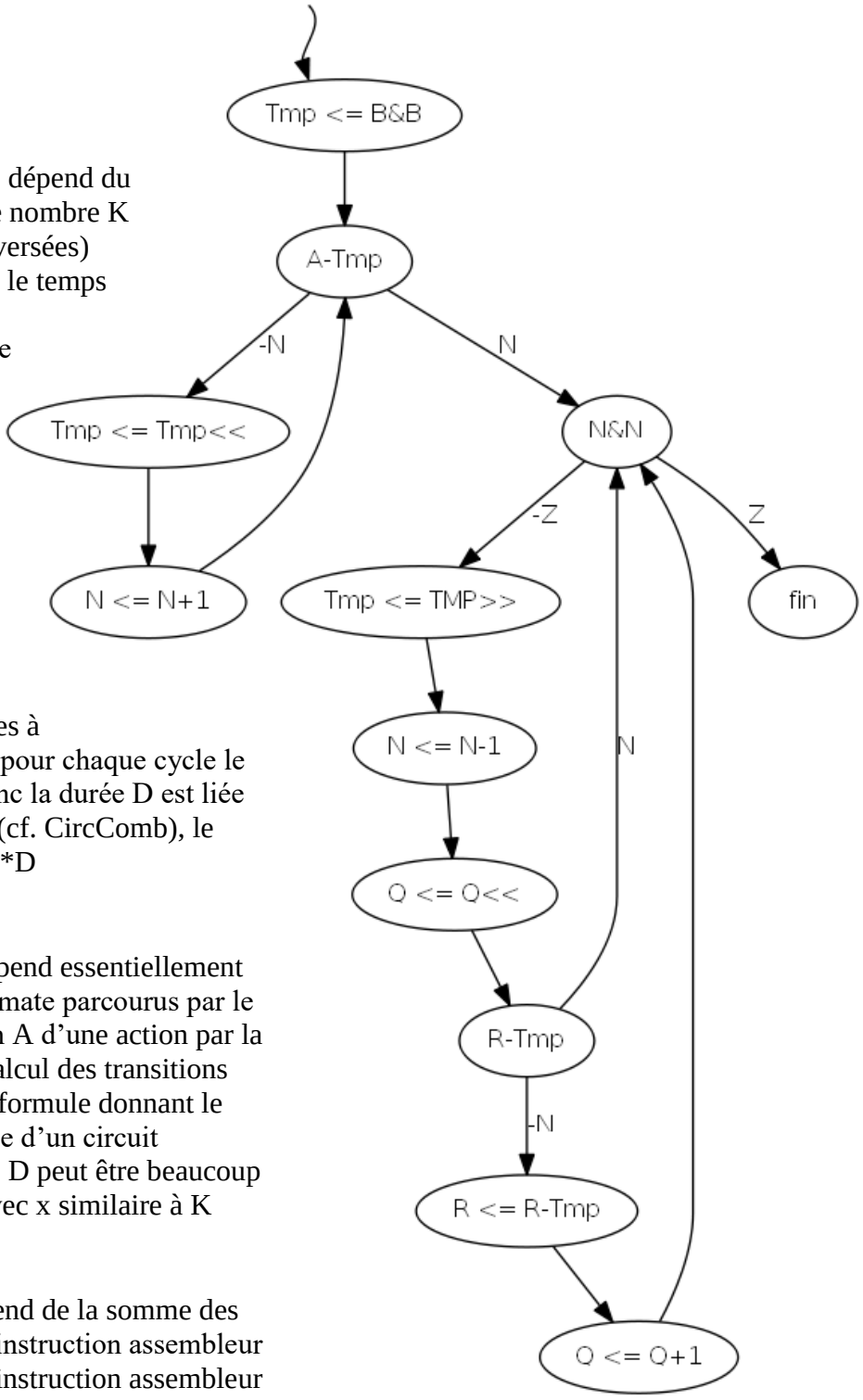


V. Division

Q1. Partie opérative : 6 registres donc 3 initialisés à 0 (R2 pour Q, R3 pour R, R4 pour Tmp), un registre initialisé à 1 (R5) et 2 registres initialisés avec les entrées de l'algorithme (R0 : A et R1 : B) ; 5 fonctions : addition, soustraction, et logique, décalage droit (1 bit), décalage gauche (1 bit)



Q2. Automate formel (ci-contre)



VI. Question de temps

- CircComb : le temps de calcul dépend du plus long chemin (en terme de nombre K de portes logiques de base traversées) reliant une entrée à une sortie, le temps total est le produit de K par le temps de transition d'une porte logique de base T ; au final, le temps est de l'ordre de K*T

- CircFlotDonnées : hypothèse : il s'agit d'un circuit avec une seule boucle (ou cycle) cadencée par une horloge fixe ; le temps de calcul dépend du nombre N de cycles nécessaires à l'exécution du programme, et pour chaque cycle le temps dépend de l'horloge donc la durée D est liée au plus long chemin du cycle (cf. CircComb), le temps total est le produit de N*D

- PC/PO : le temps de calcul dépend essentiellement du nombre N d'états de l'automate parcourus par le calcul et du temps d'exécution A d'une action par la PO, (on néglige le temps de calcul des transitions dans l'automate) ; au final, la formule donnant le temps N*A est similaire à celle d'un circuit CircFlotDonnées (N*D), mais D peut être beaucoup plus grand que A (D = x*A avec x similaire à K comme dans CircComb)

- ASM : le temps de calcul dépend de la somme des temps d'exécution de chaque instruction assembleur exécutée, ce temps (pour une instruction assembleur donnée) dépend du nombre d'états parcourus dans l'automate d'interprétation (PC) du langage machine de l'ordinateur et du temps de calcul d'une action élémentaire de la PO de l'ordinateur. Pour simplifier, si toutes les instructions assembleur étaient interprétées avec le même nombre d'état n de la PC, chaque action de la PO prenant un temps t, le temps total de calcul serait donné en fonction du nombre d'instructions assembleur exécutées i par le produit : i*n*t.