CC-TD-F - LIF6 Architecture matérielle et logicielle

CCF, le vendredi 15 juin 2012, 8h-9h30 en amphi. Thémis 8.

Votre copie doit être anonymisée ; vous pouvez rendre votre sujet complété avec votre copie : dans ce cas, il faut y préciser votre numéro de copie. Aucun document autorisé, durée 1h30, barème donné à titre indicatif. Merci de respecter le plus possible l'ordre des questions.

I Addition séquentielle (10 pts)

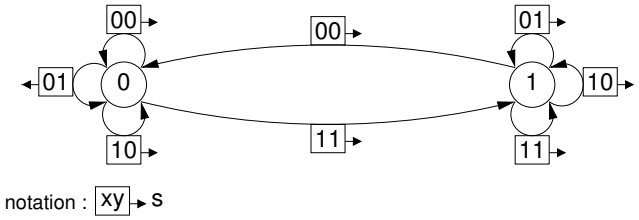
On veut mettre au point un circuit séquentiel pour effectuer une addition en séquentiel ; en n cycles d'horloge le circuit calculera $(s_{n-1} \dots s_1 s_0)_2 = (x_{n-1} \dots x_1 x_0)_2 + (y_{n-1} \dots y_1 y_0)_2$. Au cycle i $(n-1 \geq i \geq 0)$:

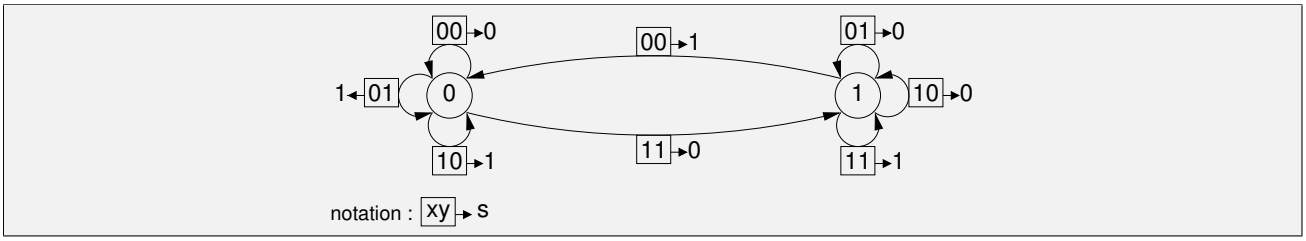
- l'état courant du circuit est donné par la valeur de la retenue c_i (on suppose $c_0 = 0$) ;
- le bit de somme est $s_i = g(x_i, y_i, c_i) = x_i + (y_i + c_i)$ (opérations arithmétiques),
- le nouveau bit de retenue à propager $c_{i+1} = f(x_i, y_i, c_i)$.

La fonction $f : \mathbb{B}^3 \to \mathbb{B}$ est donc la fonction de transition du circuit, et $g : \mathbb{B}^3 \to \mathbb{B}$ la fonction de sortie.

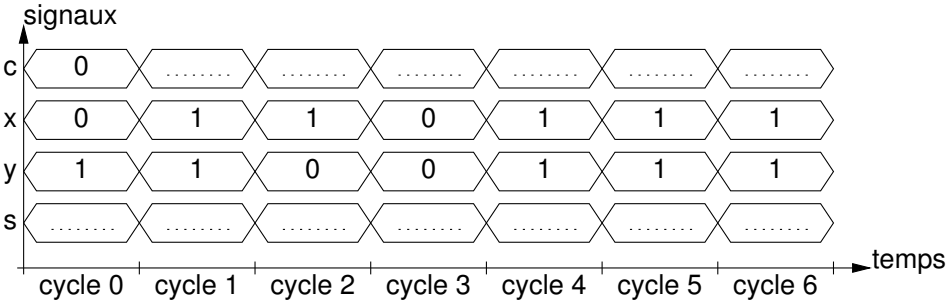
Dans la suite, on s'intéresse à ce qui se passe à un cycle i quelconque, donc on oublie les indices : c est l'état courant du circuit, x et y sont les deux bits en entrée du circuit à additionner, $s = g(x, y, c)$ est le bit de somme au rang i en sortie du circuit, et $c^+ = f(x, y, c)$ donne l'état suivant du circuit.

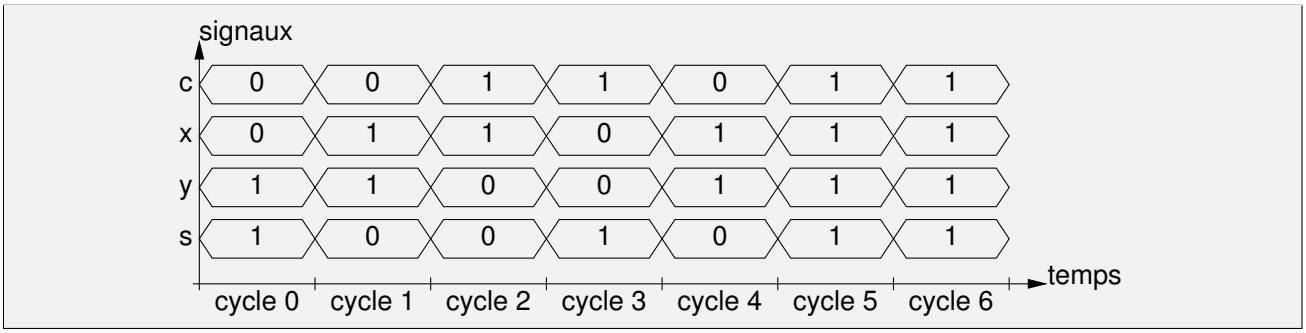
Q.I.1) - (1 pt) Complétez la représentation graphique de l'automate fini séquentiel pour le circuit séquentiel demandé, en indication les valeurs prises par la fonction de sortie.





Q.I.2) - (2 pts) Complétez le chronogramme suivant.





Q.I.3) - (1 pt) Complétez le tableau suivant :

c	x	y	s	c^+
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

c	x	y	s	c^+
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

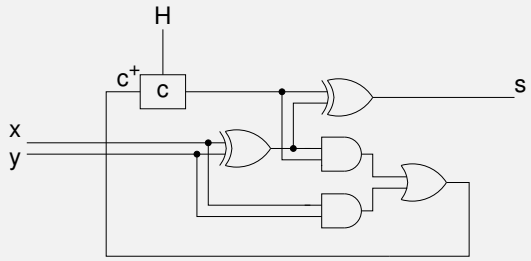
Q.I.4) - (1 pts) Montrez (en justifiant) que $s = x \oplus y \oplus c$.

Pour s, on constate que $s = \overline{c}(x \oplus y) + c(\overline{x \oplus y})$. Comme $a \oplus b = \overline{a}b + a\overline{b}$, on en déduit que $s = c \oplus (x \oplus y)$, et comme le $\oplus$ est commutatif et associatif, on obtient la formule annoncée.

Q.I.5) - (2 pts) En partant de la forme norme disjonctive pour s, montrez (en justifiant) que $c^+ = (x \oplus y)c + xy$.

On a $c^+ = \overline{c}xy + c\overline{x}y + cx\overline{y} + cxy$. On en déduit $c^+ = (\overline{x}y + x\overline{y})c + (\overline{c} + c)xy = (x \oplus y)c + xy$.

Q.I.6) - (3 pts) Déduisez des questions précédentes un circuit implantant un additionneur séquentiel.



II Programmation en assembleur du LC3

Il s'agit de compléter deux routines dans le langage d'assemblage du LC3. Il est important de vous conformer aux commentaires indiqués dans le programme et de commentez également soigneusement votre code. Reportez proprement vos réponses sur votre copie.

II.1 Nombre d'occurrences (5 pts)

Complétez la routine `nboccs` pour qu'elle compte le nombre d'occurrences d'un caractère dans une chaîne de caractères (par exemple, le nombre d'occurrences de `'o'` dans `"Toto fait du vélo"` est 3). Rappel : le code du caractère de fin de chaine, noté généralement `'\0'`, est tout simplement l'entier 0.

```
        .ORIG x3000
; Programme principal
        LEA R0,ch       ; charge l'adresse désignée par ch dans R0
        LD R1,car       ; charge le caractère à l'adresse car dans R1
        JSR nboccs      ; appelle la routine nboccs
        HALT
; Partie dédiée aux données
ch:     .STRINGZ "Toto fait du vélo"
car:    .FILL x006F     ; code ASCII du caractère 'o'
```

```
; Sous-routine pour pour compter le nombre d'occurrences d'un caractère
; dans une chaîne de caractères terminée par un '\0'.
; paramètres d'entrée   : R0 contient l'adresse du début de la chaîne
;                         R1 contient le code ASCII d'un caractère c
; paramètre de sortie   : R2 le nombre d'occurrences du caractère c
; registres temporaires : R3 contiendra l'opposé de R1
;                         R4 servira de pointeur pour parcourir la chaîne
;                         R5 recevra successivement chacun des caractères de la chaîne
nboccs:
### PARTIE A COMPLETER ###
         .END
```

```
; Sous-routine pour pour compter le nombre d'occurrences
; dans une chaîne de caractères terminée par un '\0'.
; paramètres d'entrée   : R0 contient l'adresse du début de la chaîne
;                         R1 contient le code ASCII d'un caractère
; paramètre de sortie   : R2 le nombre d'occurrences du caractère
; registres temporaires : R3 contiendra l'opposé de R1
;                         R4 servira de pointeur pour parcourir la chaîne
;                         R5 recevra successivement chacun des caractères de la chaîne
nboccs:  AND R2,R2,0    ; R2 <- 0, initialisation du compteur
         ADD R4,R0,0    ; R4 <- R0, initialisation du pointeur
         NOT R3,R1
         ADD R3,R3,1    ; R3 <- -R1
loop:    LDR R5,R4,0    ; R5 <- le caractère pointé par R4
         BRz endloop    ; si le caractère est 0, on sort de la boucle
         ADD R5,R5,R3   ; R5 <- R5+R3 = R5-R1
         BRnp diff      ; cas de non-égalité, il ne faut pas incrémenter R2
         ADD R2,R2,1    ; en cas d'égalité, on incrémente le compteur
diff:    ADD R4,R4,1    ; on incrémente le pointeur
         BR loop
endloop: RET            ; retour de la routine
```

II.2 Ajouter 1 aux éléments d'un tableau (5 pts)

Complétez la routine `add` pour qu'elle permette d'ajouter 1 à chacune des cases mémoire dont les adresses sont comprises entre l'adresse contenu dans `R0` et celle contenue dans `R1` lors de l'appel.

```
          .ORIG x3000     ; adresse de début de programme
; Programme principal
          LEA R0,debut    ; charge l'adresse du premier élément dans R0
          LEA R1,fin      ; charge l'adresse du dernier élément dans R1
          JSR add         ; appelle la routine add
          HALT            ; termine le programme
; Partie dédiée aux données
debut:    .FILL #1
          .FILL #3
          .FILL #5
          .FILL #7
fin:      .FILL #9
; Routine add, pour ajouter 1 au contenu des cases dont les adresses sont
; comprises entre l'adresse contenue dans R0 et celle contenue dans R1.
; On suppose que R0 <= R1
; paramètres d'entrée : R0 et R1, adresses de début et de fin du tableau
; registre temporaire : R2 servira de pointeur pour parcourir les cases du tableau
                        R3 registre dans lequel on effectuera des calculs
add:
### PARTIE A COMPLETER ###
         .END
```

```
; Routine add, pour ajouter 1 au contenu des cases dont les adresses sont
; comprises entre l'adresse contenue dans R0 et celle contenue dans R1.
; On suppose que R0 <= R1
; paramètres d'entrée : R0 et R1, adresses de début et de fin du tableau
; registre temporaire : R2 servira de pointeur pour parcourir les cases du tableau
                        R3 registre dans lequel on effectuera des calculs
add:       AND R2,R2,0
           ADD R2,R0,0     ; R2 <- R0
loop:      LDR R3,R2,0     ; R3 <- tab[R2]
           ADD R3,R3,1     ; R3 <- R3+1, on calcule le résultat
           STR R3,R2,0     ; tab[R2] <- R3
           ADD R2,R2,1     ; on incrémente le pointeur
           NOT R3,R1
           ADD R3,R3,1     ; R3 <- -R1
           ADD R3,R3,R2    ; R3 <- R2+R3 = R2-R1
           BRp endloop     ; si R2>R1, on sort de la boucle
           BR loop         ; sinon, on reprend la boucle
endloop:   RET
           .END
```