**L2 Info - 2014-2015**
**Architecture Matérielle et Logicielle**
L. Gonnord & N. Louvet
http://laure.gonnord.org/pro/teaching/archiL2.html

# DM archi 2014 - corrigé

## 1 Automate reconnaisseur (cf tp4)

Il est demandé de terminer l'exercice du TP :

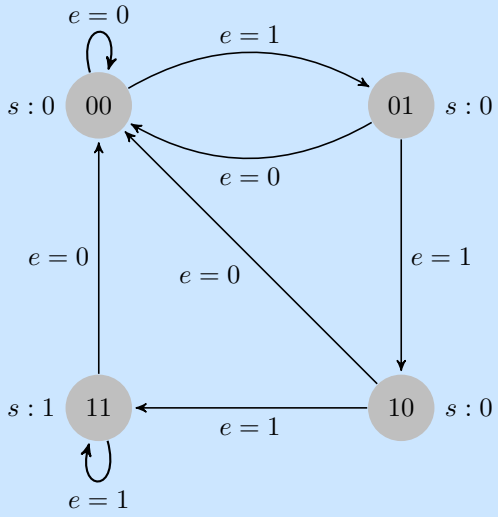
Exercice 1 (Un automate reconnaisseur). Construire en Logisim un automate séquentiel reconnaissant le motif `111` (en séquence, c'est à dire que l'automate sort 1 dès qu'il a lu trois '1' de suite, et continue à sortir 1 tant que les 3 dernières entrées lues sont des '1', ...). On utilisera la méthodologie suivante :

- Décrire la machine voulue en langage courant. *En particulier, que veut dire "reconnaître" ?*
- Quelles sont les entrées et les sorties de l'automate à construire ?
- Dessiner un automate équivalent au circuit à construire.
- Construire la table de vérité du circuit. *L'état suivant et le/les sorties sont calculées à partir de l'état courant et de/des entrées*
- Dessiner sur papier le circuit.
- Dessiner avec Logisim et tester (ce n'est pas si simple !)

Solution. La fonctionalité du circuit est :

```
00101101110101101001110111110 <-- INPUT
00000000010000000000100011110 <-- OUTPUT
```

Il y a donc une unique entrée sur 1 bit et une unique sortie sur 1 bits. Sans surprise, les états de notre automate vont coder "j'ai lu N" fois "1" de suite avec $0 \leq N \leq 3$, donc 4 états (donc 2 bits).




La table de vérité calcule la sortie $s$ en fonction de l'état courant (current state, codé sur 2 bits $q_1 q_0$) et de l'entrée ($e$, 1 bit). Elle calcule aussi le nouvel état ($q_1^+ q_0^+$). Sans suspense on obtient immédiatement :

| $q_1$ | $q_0$ | $e$ | $q_1^+$ | $q_0^+$ | $s$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 |

La fonction de sortie est facile à calculer et implémenter :

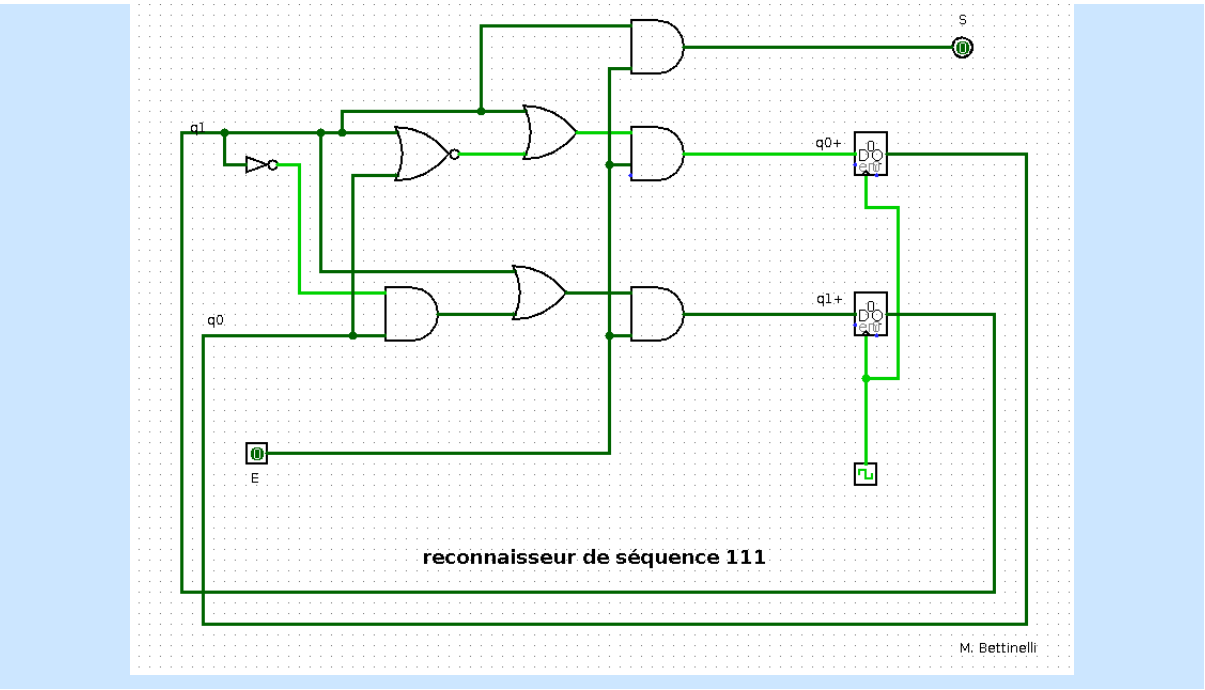
$$s = q_1.e$$

Pour les fonctions de calcul de nouvel état, les formules sont un peu plus compliquées :

$$q_0^+ = e.(\bar{q_1}\bar{q_0} + \bar{q_0}q_1 + q_0 q_1) = e.(\bar{q_0}\bar{q_1} + q_1)$$

$$q_1^+ = e.(\bar{q_1}q_0 + q_1\bar{q_0} + q_1 q_0) = e.(q_1 + q_0\bar{q_1})$$

Ce qui donne, sous logisim (noter l'utilisation du "non-ou" au lieu du "et" avec deux négations), grâce à M. Bettinelli :




## 2 Instructions LC3

En annexe vous trouverez un programme écrit en binaire LC3. Il est demandé de :

1. décoder chaque instruction hexa vers les instructions en binaire 16 bits
2. décoder chaque instruction binaire selon l'ISA du LC3, selon le code fourni par exemple dans l'ISA du LS3.
3. raconter ce que fait le programme, en écrivant bien comme il faut comme sur la première ligne ce que fait l'instruction. Lorsqu'il s'agit de load et store ou branchement, on pourra ajouter des labels au programme pour bien montrer quelles sont les données chargées (ou les sauts effectués).
4. expliquer ce que fait le programme, avec du pseudo code

Avec l'aimable permission des auteurs William Slough et Andrew Mertz, univ Estern Illinois.

| Address | Content | Content (binary) | Instruction details |
|---|---|---|---|
| x3000 | x5020 | 0101 000 000 1 0 0000 | AND<br>DR = R0 SR = R0<br>imm = 1 imm5 = x00<br>R0 = R0 AND SEXT(x00) setcc() |
| x3001 | x1221 | | |
| x3002 | xE404 | | |
| x3003 | x6681 | | |
| x3004 | x1262 | | |
| x3005 | x16FF | | |
| x3006 | x03FD | | |
| x3007 | xF025 | 1111 0000 0010 0101 | TRAP<br>trapvect8 = x25<br>halt execution |
| x3008 | x0006 | | data word |

Figure 2: A sequence of machine instructions for the LC-3 computer.

Avec l'aimable permission des auteurs William Slough et Andrew Mertz, univ Estern Illinois.

| | Address | Content | Content (binary) | Instruction details | |
|---|---|---|---|---|---|
| | x3000 | x5020 | 0101 \| 000 \| 000 \| 1 \| 0 0000 | AND<br>DR = R0 SR = R0<br>imm = 1 imm5 = x00<br>R0 = R0 AND SEXT(x00) setcc() | R0<-0 |
| | x3001 | x1221 | 0001 001 000 1 00001 | ADD<br>DR=R0 SR=R0 imm5=1<br>ADD R1 R0 1 | R1<-1 |
| | x3002 | xE404 | 1110 010 0 0000 0100 | LEA : DR = R2, offset9=x04<br>R2 = PC+5 | R2 <- x3007 (@fin) |
| | x3003 | x6681 | 0110 011 010 00 0001 | LDR : DR=R3 baseR=R2 offset6=x01<br>R3 <- mem[x3008] | R3 <- x6 |
| loop: | x3004 | x1262 | 0001 001 001 10 0010 | ADD<br>DR=R1 SR=R1 imm5=x02<br>ADD R1 R1 2 | R1 <- R1+2 |
| | x3005 | x16FF | 0001 011 011 111111 | ADD<br>DR=R3 SR=R3 imm5 = -1 !!<br>ADD R3 R3 -1 | R3<R3-1 |
| | x3006 | x03FD | 0000 001 1 1111 1101 | BRp offset= - 3 !<br>PC <- PC -3 | si R3>0 goto loop |
| fin: | x3007 | xF025 | 1111 \| 0000 \| 0010 0101 | TRAP<br>trapvect8 = x25<br>halt execution | HALT |
| | x3008 | x0006 | | data word | |

donnée

Figure 1 A sequence of machine instructions for the LC-3 computer.

TP5 Archi -