Examen de Conception de Circuits

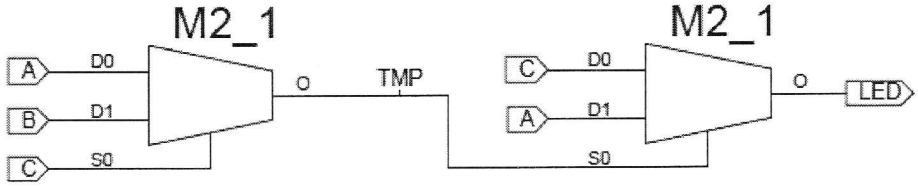
E. Mesnard

18 avril 2013

Documents autorisés : **feuille A4 manuscrite Recto/Verso**
Durée : **2 heures**

Exercice 1 (3 points) Combinatoire : Multiplexeur 2 vers 1

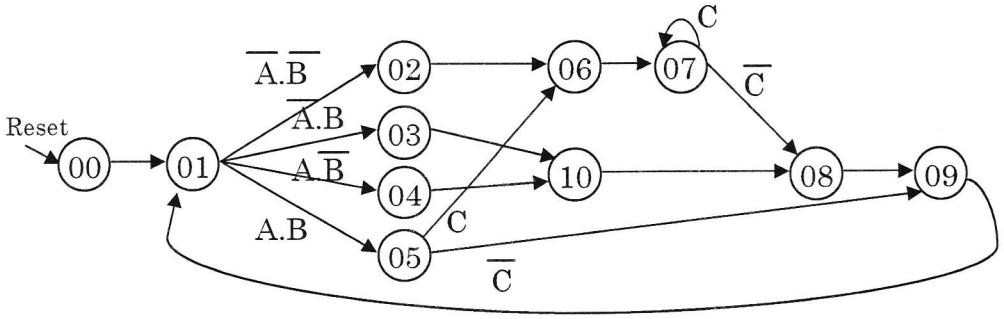
Soit le circuit suivant, constitué principalement de deux M2_1 (multiplexeurs 2 vers 1) :



Indiquer comment se comporte la *LED* (éteinte ou allumée) en fonction des entrées *A*, *B* et *C*. Présenter le résultat sous la forme **minimisée** (équation **et** schéma logique).

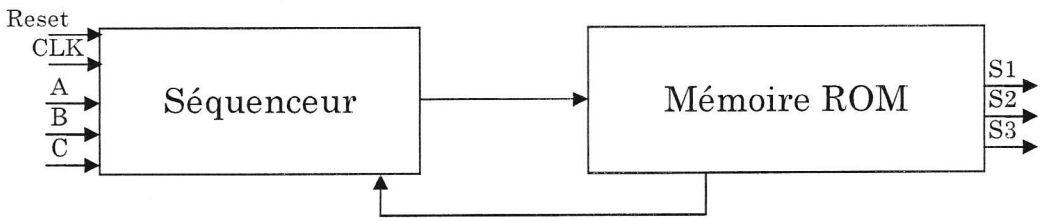
Exercice 2 (6 points) Séquentiel : Synthèse d'automate

Soit un automate dont le diagramme d'états (logique de transition τ) et le tableau de sortie (logique de sortie σ) sont fournis ci-dessous :



Etat	Sortie
0,1,3,8	S1
2,6,7	S2
3,7,8,10	S3

L'objectif général est d'appliquer la **synthèse micro-programmée** pour transformer cet automate en un circuit. Il est rappelé que cette méthode de synthèse conduit à l'élaboration d'un « séquenceur » (cadencé à la fréquence d'une horloge CLK) couplé à une « mémoire morte ROM ».



1) Dans un premier temps, appliquer la synthèse basée sur l'exploitation des deux micro-instructions « classiques » *BRCH(Condition, Adresse)* et *ECLT(CodeEclt)*. Ecrire le programme (à l'aide de ces 2 instructions), puis donner le schéma du séquenceur et le contenu bit à bit de la mémoire ROM.

2) Dans un second temps, refaire cette conception de circuit en appliquant une autre méthode micro-programmée. Dans cette variante, les micro-instructions disponibles sont :

   SEQ : passage en séquence
   SAUT(Adresse) : branchement inconditionnel à l'adresse indiquée
   BNC(Adresse) : branchement équivalent à *si C=0 alors SAUT(Adresse) sinon SEQ finsi*
   BC(Adresse) : branchement équivalent à *si C=1 alors SAUT(Adresse) sinon SEQ finsi*
   AIG_AB(Adresse_Base) : aiguillage équivalent à *SAUT(Adresse_Base + Valeur(AB))*
   Exemple : AIG_AB(5) fera un saut en 7 si A=1 et B=0, car Valeur(AB) = 0b10 = 2

   Ré-écrire le programme (à l'aide de ces nouvelles instructions), puis préciser les modifications engendrées sur le séquenceur et dans la mémoire ROM.

Problème (11 points) **Longueur d'un mot binaire**

Le but de ce problème est de concevoir **intégralement**, donc jusqu'au dessin de l'ensemble des schémas logiques, un circuit (en fait, deux versions !) qui calcule la longueur d'un mot binaire. Il est rappelé que cette longueur est le nombre de bits significatifs. Par exemple, la longueur du nombre binaire 6 (0b00000110) est 3, et celle de 93 (0b01011101) est 7.

1) Circuit version 1 : Synthèse UC/UT
   L'algorithme qu'il faut implémenter, donné ci-dessous en langage C, est optimisé pour **ne pas** faire appel à la bibliothèque mathématique (pas de logarithme, ni de multiplication ou division). Le principe est simple : le nombre est « divisé par 2 » (décalage binaire) jusqu'à ce qu'il soit nul. L'interface avec l'utilisateur est minimaliste : un bouton BTN0 pour valider et un bouton BTN1 pour acquitter, les 8 switches SW(7:0) pour la saisie du nombre N (mot binaire entier naturel, non signé, codé sur 8 bits), et les LEDs pour l'affichage du résultat.

```
unsigned char Calcul_Longueur(unsigned char N) {
unsigned char L;  // Longueur calculee sur le mot N
  L = 0;
  while (N!=0) {  // Traitement tant que le nombre n'est pas nul
    N = N >> 1;   // Decalage pour realiser une division entiere par 2
    L++;          // Incrementation de L
  }
  return(L);      // La longueur est le nombre de fois qu'il a ete
}                 // possible de diviser le nombre par 2
```

2) Circuit version 2 : Synthèse combinatoire pure
   Concevoir une seconde version de ce circuit, en la basant uniquement sur les opérateurs combinatoires. Du fait qu'il n'y ait pas de système de contrôle (pas de CLK, ni Reset, ...), l'interface est encore plus minimaliste : les LEDs présentent en temps réel la longueur du mot donné par la position des 8 switches SW(7:0).
   Expliquer le principe retenu, puis fournir les équations logiques liant les LEDs aux SW.
   Commenter, principalement en termes d'efficacité.